



AppWorks™ Platform Advanced Development Guide

Release 16.6

This documentation has been created for software version 16.6.

It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at <https://knowledge.opentext.com>.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111

Toll Free Canada/USA: 1-800-499-6544 | International: +800-4996-5440

Fax: +1-519-888-0677

Support: <https://support.opentext.com>

For more information, visit <http://www.opentext.com>

Copyright © 2019 Open Text. All Rights Reserved.

Trademarks owned by Open Text.

One or more patents may cover this product. For more information, please visit
<https://www.opentext.com/patents>

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Last updated: 4/30/2019

Table of Contents

Chapter 1 Getting started with Explorer	27
AppWorks Platform Welcome page	27
Creating a shortcut using a URL	28
Understanding tasks	29
Using the AppWorks Platform Welcome page	31
Organizations menu	32
Shortcut bar	32
Context-menu options	32
Part I WORKING WITH BUSINESS MODELS	35
Chapter 2 Business process models	36
Capabilities of AppWorks Platform business modeling	36
Benefits of using AppWorks Platform business modeling	37
Chapter 3 Modeling business processes	38
Flow connectors and message maps	38
Swimlanes and modeling events	39
Tabs on business process model editor	39
Creating a business process model	40
Designing a business process model	41
Working in grid view	42
Menu options in Model Editors	44
Setting properties of a business process model	45
Using logging service in a business process	46
Associating business identifiers to a business process model	48
Creating a KPI on a business process model	49
Modifying the business process management service properties	51
Business process management service properties interface	52

Configuring BPMN constructs	60
Setting the properties for Start event	60
Setting the properties of a Decision construct	61
Setting the properties of an End event	62
Setting the properties of an Activity	67
Setting the properties of Receive Message Event	67
Setting the properties of a Compensate event	68
Setting the properties of a Delay event	69
Setting the properties of a Catch Exception event	69
Setting the properties of an independent subprocess	70
Setting the properties of a Send Message event	70
Setting the properties of Time-out	71
Setting the properties of a Swimlane	71
Setting the properties of an Annotation	72
Setting the properties of While construct	72
Setting the properties of Until	73
Setting the properties of For Each	74
Setting the properties of a Transaction	74
Setting the properties of an Embedded Sub-process	75
Using message maps	75
Creating a message map	77
Using the consolidated view	89
Configuring message filters	91
Message filter interface	92
Setting the process execution mode	93
Attaching web services to a business process model	94
Attaching a user interface to a business process model	96
Invoking a decision table from a business process model	97
Enabling the rule engine for decision table	97
Enabling reliable messaging in a business process	98
Documenting business process models	99
Configuring the document generation properties	100
Validating a business process model	103

Debugging a business process model	103
Configurable toolbar items in business process and case modelers	108
Executing a process model	109
Instantiating a business process model	110
Viewing process instances	111
Viewing a process instance message map	112
Viewing and editing the XML structure of a WS-AppServer model	113
XPDL support	114
Importing an XPDL file	115
Exporting a business process model to an XPDL format	116
Printing a business process model	117
Importing VCMDATA or Zip files	117
Creating a business identifier	119
Switching to the improved business process modeling environment	120
Switching a business process	120
Reverting a switched business process	121
Behavioral changes of business process models in AppWorks Platform versions	122
Transactional improvements in short-lived business processes	131
Defining runtime security	131
Chapter 4 Working with business activity monitoring	134
Configuring a new database for BAM	135
Creating a tablespace	138
Creating a service group for BAM	139
Using features of BAM	141
Defining business event responses	141
Creating a business event response	142
Publishing a business event response	153
Unpublishing a business event response	154
Editing a business event response	154
Deleting a business event response	155
Modeling process monitoring objects	156
Guidelines for developing the Process Monitoring object	156
Creating a process monitoring object	158

Publishing a process monitoring object	163
Unpublishing a process monitoring object	164
Editing a process monitoring object	165
Deleting a process monitoring object	165
Building business measures	166
Creating a business measure	167
Publishing a business measure	181
Unpublishing a Business Measure	181
Editing a Business Measure	182
Deleting a business measure	182
Defining a KPI	183
Creating a KPI	184
Publishing a KPI	195
Unpublishing a KPI	196
Editing a KPI	197
Deleting a KPI	197
Composing dashboards	198
Creating dashboards	199
Viewing dashboards using standard views	204
Using business activity monitoring	235
Chapter 5 Modeling cases	238
Creating a case model	238
Designing a case model	239
Setting the properties of a case model	242
Setting the properties of a case activity	249
Setting the properties of a state	256
Setting the properties of a transition	256
Examples of various case models	257
Executing a case model	262
Case instance specific properties	262
XML structure of instance-specific properties	262
Elements in the XML structure of instance-specific properties	263
Attaching files to a case	266

Chapter 6 Working with a business calendar	269
Creating a business calendar	270
Defining a business week pattern	270
Creating an exception list to the business calendar	271
Importing calendar exceptions	271
Defining exceptions to the business calendar	272
iCal import wizard calendar exceptions interface	273
Linking calendar exceptions to the business calendar	273
Setting time zone to a business calendar	274
Publishing a business calendar to an organization	275
Using a business calendar	275
Planning in hours and minutes	275
Planning in business days	276
Using a business calendar in a business process model	277
Chapter 7 Modeling data	279
Creating an XML schema	279
Creating XML schema fragments	281
XML schema created from an external URL	283
Creating an object template	284
Modeling relational data	285
Using WS-AppServer package to model relational data	286
Modeling data transformation	291
Creating a data transformation model	292
Transforming data using XSLT functions	294
Testing a data transformation model	296
Creating a content map	297
Chapter 8 Modeling business rules	299
Rules	300
Rule groups	300
Rule types	300
Creating a business rule	301
Properties of a rule	302
Setting effective and expiry dates for rules	304

Testing rules using rule test tool	304
Creating a business rule template	305
Building a decision table	306
Properties of a decision table	307
Accessing the business object in decision table editor	309
Working with a condition template	311
Building a condition expression	312
Types of parameters	313
Working with an action template	314
Creating a rule group	316
Chapter 9 Modeling schedules	318
Creating a schedule	319
Deploying a schedule	321
Chapter 10 Modeling human interactions	323
Creating a delivery model	324
Inbox model datatypes	326
Working with dispatch algorithm	328
Creating a Java archive definition	331
Working with Inbox	332
Attaching files to a task	335
Creating an Inbox model	336
Customizing the Inbox view	338
Personalizing the Inbox view	339
Setting Inbox preferences	341
Searching notifications	342
Searching tasks in the Inbox	343
Forwarding notifications	350
Forwarding tasks	350
Working with tasks in the work list	352
Working with cases in the Inbox	372
Creating an e-mail model	376
Create a static E-mail model	376
Create an E-mail model based on an XML schema document	376

Create an E-mail Model on a user interface document	377
Working with the Email model editor	377
Adding a workflow library to user interface applications	378
Adding a workflow library to HTML applications	379
Chapter 11 Modeling roles	381
Creating roles	382
Editing or deleting a role	382
Assigning menus and toolbars to users	383
Assigning user interfaces to roles	384
Assigning users to roles	386
Assigning multiple roles to a user in a team	387
Assigning sub-roles (Managing roles)	387
Assigning roles from multiple teams to a user	388
Configured task definition	389
Overview of task status	391
Chapter 12 Modeling master data	393
Preparing master data repository	394
Creating an MDM model	396
Subscription properties interface	398
Creating the components of an MDM model	400
Creating a data store	401
Creating an MDM subscription handler	403
Designing a state model	403
Creating a data quality plan	412
Creating a data quality service	413
Using business identifiers	413
Chapter 13 Creating a business process category	415
Business process category properties interface	415
Part II DEVELOPING APPLICATIONS	417
Chapter 14 Developing applications	418
Chapter 15 Collaborative workspace - Team development setups	419
Separate developer stations that are configured to a Source Control Management (SCM) system	420

Separate workspaces, where a central development server is configured to an SCM system	420
Multiple development servers that are configured to an SCM system	421
A single Workspace with a central development server	422
Setting up the CWS development environment	423
Creating a workspace	423
Working with workspace documents	425
Creating a solution	426
Creating a project	428
Creating a document	429
Using the Quick Access Menu	434
Chapter 16 Working with web services	435
Generating web service operations on databases	435
Generating standard web service operations on a database	436
Generating custom web service operations on database details	437
Generating application and business logic on relational data	439
Setting access control on tables and views of a database	458
Setting access control on database procedures	460
Generating a web service operation on a business process model	460
Generating a web service operation on a data transformation model	462
Generating web service operations on Java classes	463
Generating a web service operation on an object template	465
Generating a web service operation on a decision table	466
Generating web service operations on custom logic	468
Generate Web Service Operations on Custom Logic - Wizard interface	468
Generating web service operations on a WS-AppServer package	469
Generate Web Service Operations on WS-AppServer Package - Wizard interface	470
Generate Web Service Interface on WS-AppServer Models - Wizard interface ...	471
Executing a web service operation using web services explorer	472
Generating Web Service operations on database metadata	473
Generate Web Service Operations on Database Metadata interface	473
Working with external web services	475
Searching for external web services	476

Generating AppWorks Platform web service operations for external web services	477
Configuring access to external web services	480
Attaching service group to a web service interface	481
Publishing web services to an external UDDI registry	481
Personal external access	486
Viewing the WSDL of a web service operation	487
Updating web services	487
Testing web service operations	488
Setting access control on web service interface and web service operations	489
Generating a web service operation on an email model	490
Viewing inline schema of web services	491
Chapter 17 Working with web applications	492
Working with XForms	492
Creating a user interface using a web page URL	493
Creating XForms	494
Customizing controls	499
Customizing XForms	528
Working with data	543
Working with composite controls	555
Integrating an XForm with WS-AppServer	587
Using the undo and redo features	588
Interceptor support	589
Managing XForms	590
Customizing interface styles	592
Publishing XForms to runtime	595
Managing translation	595
Working with JavaScript message bundles	595
Translating XForms	596
Translating validation and UFO messages	597
Reconstructing missing translations	598
Translating text to a target language	600
Enabling multi-browser support for web content	602
Application or form changes for multi-browser support	604

Script, style, and XML data island changes for multi-browser support	605
Type library changes for multi-browser support	605
HTML and JavaScript changes for multi-browser support	606
XML changes for multi-browser support	610
Event model changes for multi-browser support	614
Regular expression changes for multi-browser support	617
Style changes for multi-browser support	617
Accessing XForms and translations using URLs	625
Accessing XForms	625
Accessing translations	626
Viewing translations of XForms based on browser settings	627
Accessing AppWorks Platform with a specific locale	628
Working with bi-directional support in AppWorks Platform user interfaces	628
Customizing CSS for RTL support	629
Customizing HTML for RTL support	630
Developing user interfaces in AppWorks Platform	631
Language codes	632
Supporting controls	632
Working with HTML5 SDK	633
Overview	633
Getting started with the SDK	638
Script loader using require.js	650
Organization level styling and BASE relative URLs	654
Organization-specific web content	654
From absolute to BASE relative URLs	654
BASE URL	654
BASE element and application.js	655
Referring to other Web content	655
Examples	656
Special care with URLs from CSS	656
Upgrading an existing Web content	657
Managing AppWorks Platform using web task manager	657
Overview of task	658

Chapter 18 Working with application connectors	660
Creating and configuring a custom connector	661
Application connectors	663
Auditing	663
BAM connector	663
Business Process Management	663
CoBOC service	663
AppWorks Platform (<instance name>) service	664
Other - Custom application connector	664
Collaborative workspace connector	664
Data transformation service	664
Document store	664
E-mail connector	664
Event service	665
File transfer protocol (FTP) connector	665
LDAP connector	665
MDM publisher	665
MDM service	665
Notification service	665
Rule repository service connector	666
Scheduler service connector	666
Security administration	666
Single Sign-On	666
Universal description, discovery, and integration (UDDI) connector	666
User management	667
WS-AppServer connector	667
XForms connector	667
HTTP connector	667
JMS connector	668
Repository	668
Part III HOW TOs	669
Chapter 19 Business process modeling (How Tos)	670
Invoking a web service in rules	671

Invoking a web service through schedules	672
Testing web service operations using service test tool	673
Triggering a business process from rules	674
Triggering a business process from schedules	675
Triggering a business process model from a WS-AppServer application	677
Triggering a notification request from an XForm	678
Triggering rules from a business process model	680
Triggering a business process model from an XForm	684
Using data transformation web service in a business process model	687
Using a web service in a data transformation model	691
Using a web service in a business process model	693
Using an external web service in a business process model	696
Using an Inbox model in a business process model	699
Using an Email model in a business process model	702
Attaching files to an email	705
Using human interaction in a business process model	706
Using human interaction in a case model	711
Using a business rule in an XForm	713
Using WS-AppServer business logic in XForms	715
Using an automatic follow-up connector in a case model	720
Using free follow-up in a case model	721
Using a document received event in a case model	722
Using an intermediate follow-up connector in a case model	724
Using the applicability service in a case model	725
Step 1: Create the case data schema fragment	728
Step 2: Create the decision table schema fragment	728
Step 3: Create the Applicability Service using contract	728
Step 4: Implement the Web service using a BPM	729
Step 5: Create the decision table	729
Step 6: Update the BPM	730
Step 7: Build the Message Map assignments	730
Step 8: Configure the Applicability Service in the case model	732
Step 9: Execute the case model	732

Triggering a case model from an XForm	733
Configuring attachments in BPM	736
Publishing business identifier values	738
Using a manual follow-up connector in a case model	741
Enabling debugging on previous business process models	742
Chapter 20 Application development (How Tos)	744
Developing Web services using the HTTP connector	744
Using REST with HTTP connector	744
Creating a Web service using the HTTP connector	746
HTTP Get request without parameters	747
HTTP Get request with parameters service URI	750
Transforming request and response using XSLT	752
Handling JSON as input and output parameters of REST service	754
Using authentication	755
Using HTTPS	756
Adding custom HTTP headers	757
Organization awareness	758
Adding HTTP header content dynamically	758
Common errors	761
Accessing applications within applications using iframecontainer	762
Calling a web service from a WS-AppServer application	763
Creating an XForm using web service operation generated on data source	764
Attaching custom scripts to an XForm	765
Downloading a file from a service	765
Exporting data from an XGrid control to MS Excel sheet	767
Handling globalization aspects in web application development	769
Reordering rows in a table	772
Uploading a file to a service	773
Using Inbox as a composite control	775
Uploading a file using the Upload control	776
Using WS-AppServer custom class as an alternative to object template	777
Using WS-AppServer custom class in the Transformation application	778
Guidelines to remodel Object Templates in the existing applications to WS-AppServer custom class	780

Extending a connection in a WS-AppServer application	780
Proxy user script	782
Developing applications with HTML5 SDK	783
Accessing the REST Web service	784
Authenticating users with HTML5 SDK	784
Building a Custom Approval page	785
Building read-only mobile applications	792
Building transactional mobile applications	795
Displaying a KPI	798
Displaying a geolocation on a mobile device	801
Displaying graphs on a mobile device	806
Formatting a page per user preferences	810
Implementing pagination	815
Integrating with Google Maps using HTML5 SDK	819
Integrating with WS-AppServer	821
Retrieving the data	828
Translating a page into a specific language	829
Validating a page using jQuery validation plug-in	834
Viewing Inbox tasks	839
Working with the generic approval page	840
Invoking AppWorks Platform REST APIs	843
Using ProcessInstances as a composite control	845
Using ProcessInstanceGraphicalView as a composite control	847
Using ProcessInstanceActivities as a composite control	850
Setting up continuous integration for AppWorks Platform applications	851
Building an application	851
Deploying an application	852
Testing an application	853
Using Brava to view documents	854
Part IV BUSINESS MODELING REFERENCES	856
Chapter 21 Business modeling references	857
Easy way to draw constructs in design time	857
Business process modeling reference	858

Business processing interface	858
Constructs used in a business process model	923
Business modeling roles	943
Operation types	944
Parameter types	951
Interprocess communication in business process models	955
Process execution modes	956
Modeling a transaction	957
Data modeling reference	961
Rule elements	961
Rule actions	962
Rule engine function library	968
XPath and XSLT	980
Chapter 22 Using the custom XSLT function	1007
Scenario 1: Creating repeated target elements based on multiple source elements without linking from Source	1008
Scenario 2: Creating an element under repeated target node linking from Source1010	1010
Scenario 3: Creating repeated target elements when Source element is selected and an Absolute XPath is provided	1012
Scenario 4: Creating repeated target elements when the Source element is not selected but a direct link is provided for other element	1014
Scenario 5: Creating repeated target elements when the Source element is selected and a relative XPath is provided	1015
Scenario 6 : Creating a Target element based on the Source elements value without the Source element selected directly	1017
Chapter 23 Using the Repeat Target instruction	1023
Scenario 1: When a single Source element is mapped to Repeat Target instruction	1025
Scenario 2: When multiple source elements of the same parent are mapped to the Repeat Target instruction	1026
Scenario 3 : When multiple Source elements of different parents are mapped to the Repeat Target instruction	1027
Constructs used in a case model	1044
State properties interface	1047
Transition properties interface	1048
Permitted activities in a transaction	1050

Service container properties interface	1051
Auditing service connections parameters interface	1054
Advanced properties	1055
Service group configuration interface	1057
Connection point configuration interface	1058
Service container configuration interface	1058
Clone service container configuration interface	1062
Working with process event listeners	1062
Creating process event listeners	1063
Configuring process event listeners	1066
Business object instance interface	1069
Business object instance state history interface	1070
Mapping of BPML to BPMN constructs	1070
Modeling atomic and open-ended transactions	1074
Model-View-Controller	1076
Chapter 24 Collaborative Workspace	1077
Validating a project	1077
Adding existing projects to a solution	1078
Validating a document	1079
Inserting related documents and exploring workspace contents	1079
Locating an existing document in the workspace	1080
Managing an application package	1081
Modifying the contents of an installed application package	1082
Publishing a modified application to run time	1083
Managing dependencies between documents	1084
Performing miscellaneous common tasks on application content	1086
Publishing a project to an organization	1087
Publishing a document to an organization	1088
Using the qualified name	1088
Reverting changes made in the workspace	1092
Setting access control on XML store definition	1093
Sharing application content of a workspace with other developers	1094
Switching between workspaces	1095

Synchronizing workspace and file system	1096
Resolving synchronization conflicts overwrites the workspace content with the content in the file system	1097
Updating the workspace with content modified by other developers	1099
Uploading a document to the project	1100
Using localization bundles in an application	1100
Viewing and modifying properties of a workspace	1102
Viewing and modifying properties of a solution	1103
Viewing and modifying properties of a project	1104
Viewing the change history of a workspace and its contents	1104
Reloading runtime references	1105
Using the CWS command-line tools	1107
Generic CWS command with subcommands	1108
Specific CWS commands	1109
CWS sub command: CreateSVNWorkspace	1110
CWS sub command: PruneObsoleteAdministration	1111
CWS sub command: RemovePSL	1112
CWS command-line tool: CWSBuild	1113
CWS command-line tool: CWSDeploy	1114
CWS command-line tool: CWSOverview	1115
CWS command-line tool: CWSPackage	1116
CWS command-line tool: CWSSynchronizer	1116
CWS command-line tool: CWSWorkspaceRemovalTool	1117
CWS command-line tool: MoveSynchronizerLocation	1118
CWS command-line tool: ResetSynchronizer	1119
CWS command-line tool: UpgradeSVNWorkingCopy	1120
CWS sub command: validate	1122
CWS sub command: package	1122
CWS sub command: synchronize	1123
CWS sub command: removeworkspaces	1124
CWS sub command: publish	1125
CWS sub command: movesynchronizelocation	1125
CWS sub command: reloadruntimereferences	1126
CWS sub command: resetsynchronizer	1127

CWS sub command: upgradesvnworkingcopy	1128
CWS sub command: removeworkspace	1129
Defining a custom schedule for object pruning	1130
Document locking	1131
Using models of other applications	1133
Model contracts	1134
Model packages	1139
Runtime references	1146
Analyzing package dependencies among application packages	1149
Correcting absolute and relative paths in web artifacts	1150
Correcting absolute paths	1151
Correcting invalid relative paths	1152
Preperation for dynamic base URL and future changes	1152
Creating and downloading application packages	1153
Deploying application artifacts to the AppWorks Platform_installdir	1154
Deploying artifacts to the AppWorks Platform web server	1154
Creating a web library definition	1155
Impact analysis across development workspaces	1156
Steps to enable the generation of package dependency files	1156
Understanding the contents of the package dependency file	1157
Using the information from the package dependency files	1160
Qualified names versus artifact IDs	1161
Setting properties of an application package	1161
Setting the SVN working copy format of team development workspaces	1164
Upgrading application development workspaces	1165
Upgrading stand-alone development workspaces	1165
Upgrading team development workspaces	1169
Viewing locks on the SCM repository	1173
Collaborative Workspace Service connection interface	1174
Chapter 25 Application development (Reference)	1176
Operator precedence and associativity	1176
XForms references	1177
AppWorks Platform XForms overview	1177

AppWorks Platform XForms architecture	1178
Overview of AppWorks Platform AJAX toolkit	1179
Model-View-Controller concept in XForms	1180
XForms designer	1182
Script editor	1183
XML editor	1184
Interface design overview	1185
Model properties dialog box	1195
Property sheet of controls, regions, and AppPalletes	1200
Controls available for creating property sheets	1205
Predefined XForms	1206
Predefined and specific formats for data types	1208
Script editor toolbar	1211
Selectors in cascading style sheets	1212
Keyboard shortcuts (XForms)	1225
DOM explorer	1226
WS-AppServer reference	1228
Customizing WS-AppServer SOAP faults	1228
Event listeners in WS-AppServer	1231
Handling database-generated fields	1236
JMX functionality in WS-AppServer	1240
Localizing WS-AppServer messages	1242
Mapping of BusObjects in WS-AppServer	1245
Managing transactions using WS-AppServer	1253
Retrieving objects participating in a WS-AppServer transaction	1258
Adding a new property	1262
Adding dynamic filters	1262
Transient attributes	1263
Attribute properties interface	1265
Using cursors in WS-AppServer	1267
Using Eclipse to implement event listeners	1268
Embedding WS-AppServer functionality in applications	1270
Working with SOAP requests and responses	1272

WS-AppServer JAR dependencies	1290
Memory management in WS-AppServer	1291
Running WS-AppServer in multitenant mode	1291
Setting access control	1294
Setting access control at the server level	1295
Setting access control at the client level	1295
ACL parameters	1296
ACL definitions	1297
A quick reference to access control	1301
ACL explorer interface	1304
Single sign-on reference	1305
Client-side single sign-on library	1305
Server-side single sign-on API	1309
Web application development	1310
Working with alert system	1310
Generating alerts	1311
Generating log messages from repository	1312
Defining AppWorks Platform alerts	1313
Publishing alerts	1313
Working with logging framework	1314
Using logging framework	1315
Generating log messages	1317
Best practices for developers using logging framework	1317
Working with problem registry	1319
Problem registry for database connectors	1320
Using problem registry	1321
Reading WSDL from other locations	1323
Invoking AppWorks Platform web services externally	1323
Exploring web service and its child entities	1325
Event service	1327
Impact of event service failure	1327
Overview of DOM	1328
XML search patterns	1331

Additional features for querying a database	1332
Application status categories	1335
Application verification scenarios	1336
Binding template interface	1336
Cell object property	1337
Configuration parameters for custom application connectors	1337
Configuring advanced search parameters	1339
Configuring application connectors	1340
Configuring applications for anonymous usage	1340
Using APIs for database access	1341
Conversion methods	1346
Currency conversion methods	1346
Parameters	1347
Return Value	1347
Other conversion methods	1347
Parameters	1347
Return value	1347
Creating a custom desktop	1348
Creating a Java class metadata	1349
Creating an XML	1349
Creating an XML store definition	1350
Creating AppWorks Platform toolbar	1351
Custom connector configuration interface	1351
Customizing the AppWorks Platform toolbar	1352
Customizing validation messages	1352
Customizing form level messages	1352
Customizing messages related to models	1352
Customizing control level messages	1353
Debugging processes at runtime	1353
External messages	1353
Fields for configuring MDM service groups	1354
File synchronization	1356
Completing search details	1357

Flow of control while loading welcome page	1359
Free-form controls	1359
Locales	1360
Manually synchronize interface	1365
Mapping variables	1365
Mapping single variables	1365
Mapping group variables	1365
Matched instance details interface	1366
Matched object instance interface	1367
Matched object instance state history interface	1368
Modifying a task	1368
Modifying the system attributes of a task	1369
Modifying an existing document	1370
Mutex and override rules	1372
Operators to build conditions	1373
Process monitoring	1373
Process instance	1375
Process instance manager interface	1375
Restart process instances interface	1376
Process options menu	1377
AppWorks Platform AJAX toolkit architecture	1377
Projects	1378
Properties of an activity within a transaction	1379
Properties of the attribute element	1380
Ranges	1382
References	1383
Crash recovery	1383
Reliable messaging	1384
Regions	1384
Defining regions inline in the HTML structure of an application	1385
Defining regions in the application definition	1387
Request and response messages	1388
Resizing a swimlane	1388

Retrieving the template with installation inputs	1389
Starting a batch of service containers	1389
Run-time documents	1390
Runtime Translation Repository for XForms	1392
Sample HTML	1392
Sample Java code snippets	1393
ApplicationConnector Class	1393
ApplicationTransaction Class	1394
Scheduler Service Interface	1395
Schema editor	1396
Select X-Axis field	1398
Select Y-Axis field	1399
Sending email	1399
Setting additional properties of a task	1400
Setting special attribute properties	1400
Setting the properties of a sub-context model	1401
Simple client interface	1402
Simple object access protocol	1402
Single sign-on connection parameters interface	1403
Single sign-on for developers	1403
Sorting notifications	1403
State machine	1404
State model notation using SCXML	1406
Synchronization settings interface	1407
Tabular interface of a deployed MDM model	1408
Task definition	1408
Task properties interface	1410
Taskbar context menu	1412
The anchoring feature	1413
The find feature	1414
The type library	1414
The validation feature	1415
Validation types	1415

Execution order of validations	1416
Validation messages	1416
Time zones	1417
TimeFrame composite control properties interface	1430
Toolbar context menu	1431
Transactional and non-transactional web service	1432
Triggering a new Process Instance	1433
Trigonometric functions of rule engine	1434
type	1435
Types of queries	1437
Uploading documents	1437
Usage of standard attributes	1438
Usage report	1439
Using a web service in a case model	1439
Using Intellisense	1441
Using the execute condition	1442
Versioning in XMLStore	1445
Web service definition set options	1445
XForms - A W3C standard	1446

Chapter 1

Getting started with Explorer

After installing and configuring AppWorks Platform, you are ready to access your applications. The displayed applications are the ones that are associated with your role or assigned to you. With the appropriate role, you can configure and administer AppWorks Platform, access various applications, and develop new processes and composite applications.

When you open AppWorks Platform in your web browser, you might be prompted for an user name and password. On successful verification of the user name and password, the Explorer opens and displays the new AppWorks Platform desktop. Your user name is displayed.

Note: While working with AppWorks Platform in a browser, ensure that the pop-up blocker is disabled. Doing so enables an alert to be displayed if you close the browser without saving your changes to the AppWorks Platform application running in the browser.

By default, the Welcome page displays a Shortcut Bar, a Task bar, and App Palettes that display your artifacts, such as applications and documents. While the Shortcut Bar and Task bar are common across all roles, the artifacts displayed in the App Palettes are role-specific and depend on the permissions granted to a role. Therefore, depending on how a role is designed, the Welcome page interface can vary in numerous ways. See [AppWorks Platform Welcome page](#) and [Using the AppWorks Platform Welcome page](#).

AppWorks Platform Welcome page

When you log in to AppWorks Platform through a web browser, the Explorer opens and you see the Welcome page. Use this page to navigate and work with the AppWorks Platform application.

The applications and documents available on the Welcome page are known as artifacts. The Welcome page provides access to all artifacts available to your user role. The Welcome page interface responds to your preferences and usage patterns, allowing for live categorization of your work artifacts. This means that depending on your usage preferences, work history, and the roles and access permissions assigned to you, the Welcome page interface is continuously updated each time you log in to the application. The Welcome page is updated depending on the artifacts that you most frequently use and those that you last accessed before logging out.

By default, the Welcome page displays a Shortcut Bar and a Task bar that are common across roles. It uses App Palettes to classify and present artifacts, such as applications and documents. The artifacts displayed in the App Palettes are role-specific and depend on the permissions granted to a role.

You can pin the **App Palette** to collapse it and maximize your working area. For a larger working area, you can also keep the Shortcut Bar in the floating mode.

When you start working with the AppWorks Platform, you can view your most recently accessed applications and documents for easy navigation. You can expand it to view all applications available to you.

You can customize your Welcome page. For example, as an administrator, you can modify the App Palette for a role, by changing the default Welcome page for users. Additionally, you can group and categorize artifacts according to your preferences.

The Shortcut Bar on the Welcome page provides access to the preferences by default. You can add the applications that you commonly use to the Shortcut Bar. The Task bar at the bottom displays all open App Palettes and, on right-click, provides an option to **Minimize all applications**. You can use the **Show open applications** option to display the applications minimized using the **Minimize all applications** option.

For information on how to use the various options on this page, see [Using the AppWorks Platform Welcome page](#).

Creating a shortcut using a URL

Shortcuts enable you to access applications directly, without having to navigate to them. The shortcut bar on the Welcome page provides default shortcuts to applications, using which you can set your user preferences or view the product documentation.

In AppWorks Platform, you can create and add your own shortcuts to the shortcut bar.

To create a shortcut for an application or artifact:

- Drag the application or artifact from the App Palette to the shortcut bar.
- Place the mouse pointer over the application or artifact, select , and select **Copy to Shortcut bar**.
- Use the URL of the application to create a shortcut.

To create a shortcut using the URL of an application:

1. In the App Palette, place the mouse pointer over the application for which the shortcut must be created, select , and select **Copy URL**.
The URL of the selected application is displayed in the Clipboard dialog box.
2. In the Clipboard dialog box, select and copy the URL.
3. In the Shortcut bar, click  and select **Create Shortcut**.
The Create Shortcut App Palette opens.

4. Make the following updates:
 - Paste the URL in **URL of the Shortcut**. This is a mandatory field.
 - Enter the text to be displayed as a tooltip in **Tooltip for the Shortcut**.
 - Enter a unique identifier in **ID**. This is a mandatory field.
 - Enter the URL of the icon to be used for the new shortcut in **Icon URL**.
 - Enter values in **Left** and **Top** to specify the position at which the application must open from the shortcut bar.
 - Enter values in **Width** and **Height** to specify the size of the application from the shortcut bar.
 - Enter the XML data structure used for passing data to applications accessed through the shortcut, in the **Data** pane.

5. Select **Create**.

The new shortcut is created in the shortcut bar.

Note: To delete a shortcut from the Shortcut bar, right-click and select **Remove from this list**.

Understanding tasks

This topic describes various tasks and their functions in AppWorks Platform.

Task name	Description
LDAP Explorer	Access LDAP objects and modify attributes of these LDAP objects.
Schedule Manager	Activate and administer schedules.
XMLStore Explorer	Add, modify, delete items and folders from the XML Store in the XMLStore Explorer.
External Services Configuration	Configure user credentials to access external Web services.
FTP Configurations Manager	Create a configuration profile for the FTP server.
Database Metadata Manager	Create and manage access control for database metadata.
CoBOC Browser	Create and manage business objects in CoBOC.
User Manager	Create, edit and delete users and roles. Also, assign design-time elements such as tasks.

Task name	Description
UDDI Registry Manager	Create, edit, validate, and delete UDDI registries.
Publish Services	Edit the web services that are published to an external registry.
Process Archival Manager	Import archived business processes and view a list of loaded archives.
Process Instance Manager	Manage business process instances by restarting aborted process instances, suspending process instances which are in waiting or running state, resuming suspended and debug ready process instances, terminating incomplete process instances, and refreshing the view.
Case Instance Manager	Manage case instances by viewing case activity details, viewing case instances based on filter criteria and by monitoring them such as suspending, terminating, and closing a case instance.
Organization Manager	Manage organizations by creating and deleting them.
License Manager	Manage AppWorks Platform Licenses through usage reports and license keys.
Security Administration	Manage security through certificates for signing applications and service groups, and authenticators.
System Resource Manager	Manage system resources such as Services, Service Groups, Connection points, method sets, and database configurations.
Translation Manager	Manage the translation languages, translation contexts, and labels and their translations, specified for various applications.
Web Task Manager	Manage web libraries and running applications.
MDM Model Browser	Publish or unpublish MDM models and monitor MDM models.
Applications Manager	Modify installed applications before publishing them to runtime.

Task name	Description
CoBOC Transaction Monitor	Monitor and view the details of CoBOC transactions.
My Inbox	Receive and work with various tasks, cases, and notifications sent to this inbox.
Restore Process Instances	Restore archived process instances.
Service Test Tool	Send and receive SOAP messages.
Rule Test Tool	Simulate the execution of rules on a business object.
Notification Preferences	Specify display name and email ID to send email notifications and tasks.
Synchronize Service Containers	Synchronize data across the service containers so that all the service containers of a group have up-to-date information.
Test Web Gateway	Test the Web gateway for performance.
Component Audit Log	View and configure audit data for AppWorks Platform artifacts.
SOAP Debugger	View and debug SOAP requests and responses.
Manage Deployed Processes	View and manage deployed business process models by performing various operations such as viewing instances by process, process monitoring options, viewing audit information, deleting a business process model, and refreshing the view.
Log Viewer	View logs.
Web Gateway Monitor	View the statistics associated with the Web gateway.

Using the AppWorks Platform Welcome page

The Welcome page is an interactive and flexible interface that is automatically customized according to the artifacts that you most frequently use and those that you last accessed before logging out. You can also use the various options on the Welcome page to work with AppWorks Platform effectively.

Organizations menu

AppWorks Platform can be configured and used in an intra-enterprise scenario. You can use AppWorks Platform to model logical business units as organizations. You can be part of one or more organizations.

The **Organizations** menu lists the organizations that are available in the AppWorks Platform application. It displays the current organization by default.

To switch to another organization, select an option from the **Organizations** list. The Welcome page is reloaded with artifacts from the selected organization.

Shortcut bar

You can use the Shortcut bar to create shortcuts to the applications or artifacts you use frequently. It scrolls into view when the mouse pointer is placed on it; it scrolls out of view on removing the mouse pointer. To remove the scrolling behavior, click .

The Shortcut bar contains the Welcome page preferences shortcut by default. You can use it to set your preferences. Additionally, you can create shortcuts to artifacts by dragging an artifact from an app palette to the Shortcut bar. You can also use a URL to create a shortcut. For details, see [Creating a shortcut using a URL](#). To remove an artifact from the Shortcut bar, right-click the artifact and select **Remove from this list**.

Context-menu options

The menus available for an app palette or artifacts in an app palette provide you with various options to work with them.

Menu options for artifacts

The menu available for artifacts may differ depending on whether the artifact you select is an application or a document. When you place the mouse pointer on an artifact in an app palette, the  icon appears. Click  in the app palette to display the following options:

Menu option	Description
Open	Opens the artifact.
Copy URL	Copies the URL of the artifact to the clipboard.
Copy to Shortcut bar	Creates a shortcut of the artifact and displays it on the Shortcut Bar.
Tag	Assigns a user-supplied tag to the artifact.
Remove from this list	Removes an artifact from the list of most-

Menu option	Description
	recently accessed artifacts. This option is available only for the recently accessed artifacts that are displayed in the collapsed view of the app palette.

Menu options for app palettes

These are the options that change the manner in which the artifacts are displayed. To see these options, right-click an app palette that contains artifacts. The **Refresh** and **Scroll by** menus are displayed.

The options available for the **Scroll by** menu are:

- **Roller**: Use it to scroll through artifacts.
- **Scrollbar**: Use it to scroll through artifacts.

Search field

You can search for artifacts in the app palette using name or associated tags.

To use the search function, expand the title bar of the app palette. Similar search fields are available for other app palettes.

- To search for artifacts by name, click  in the search field and select  (Quick Find). This is the default search mode.
- To search for artifacts by tag names, click  in the search field and select  (Tag Search). To narrow down the search, enter another tag in the search results view.

Keyboard shortcuts in an app palette

In addition to the various context-menu options, you can use the following keyboard shortcuts on the Welcome page:

- Press arrow keys to navigate among tags in a Tag Cloud or artifacts in an app palette.
- Press Ctrl+Alt+Z or Ctrl+Alt+X to tab through the open applications on the Welcome page. Each of these tabs display a revolving view of all open applications. Ctrl+Alt+Z displays a clockwise rotating view, while Ctrl+Alt+X shows an anti-clockwise rotating view.

Part I

WORKING WITH BUSINESS MODELS

Chapter 2

Business process models

Business Process Modeling is the activity of representing both the current "as is" and future "to be" processes of an enterprise, so that the current process may be analyzed and improved. A process model is an anticipation of what the business process will look like – one that you can fully or partially automate as a formal and structured process.

Using AppWorks Platform business process modeler, you can model your business processes based on their taxonomy, which means you can reuse the same model for the development of many applications. The business process models are graphical depictions of processes that become part of the application and govern how the business process performs when you run the application.

Capabilities of AppWorks Platform business modeling

Businesses thrive in the environment that surrounds them. The interaction between your business and its environment is dynamic. There are ever-changing customer needs, market situations, innovations, and economic implications, which drive you to reexamine your current processes.

The AppWorks Platform business process modeler enables you to:

- Align your processes to business needs: Cater exactly to customer needs as you understand and align your processes to meet the customers' changing needs
- Reduce research and development costs: Automating your processes saves time and money by reducing work cycles and function redundancy across the enterprise
- Acquire greater control over the market situation: Business process modeler allows you model new processes before implementation. This experimentation helps you find better ways of conducting business, effectively communicating the value of your products and services to your customers and rethinking the economic implications of your investments.
- Sustain in existing business and explore new businesses

Benefits of using AppWorks Platform business modeling

When you model your business processes using AppWorks Platform, you derive these benefits:

- Improve management of business processes to align products and services with market demand
- You are empowered with better decision-making ability
- Business processes are better streamlined, and it is easy for teams within your organization to relate to their work and deliver better quality work on time
- Identify the flip side of your business processes and determine their causes
- Helps you to understand the strength and flexibility of your business processes
- Helps you to think about implications of your processes which can, in turn, be a source of innovation
- You can create and design business models to define and address your company's business needs using AppWorks Platform. With your own models, you can increase the capacity to manage continuous change and constantly adapt to evolving business scenarios by injecting new ideas into your business models.

Using AppWorks Platform, you can model:

- [Creating a business process model](#)
- [Creating a case model](#)

Chapter 3

Modeling business processes

The AppWorks Platform business process modeling environment offers an intuitive modeling environment in which you can model your business processes using the [BPMN constructs](#). The BPMN constructs help you to model easily business processes. It also facilitates modeling complex business processes and mapping to business process execution languages.

To model a business process flow using AppWorks Platform, you can begin with modeling the events that occur to start a process, the processes that are performed, and the results of the process flow. Business decisions and branching of flows are modeled using decision cases and flow connectors. Also, a process in the flow can contain sub-processes which are graphically depicted by another business process diagram through a hyperlink to a process symbol.

You can design business models in AppWorks Platform using the business modeling environment. A business user models a business process in the business process modeling environment. A business process model visually illustrates a company's business processes. You can create a business process diagram comprising various BPM components in the business process modeling environment. This is achieved by making use of a variety of BPMN constructs such as Start event, Activity, Sub-process, End event, and so on, depending upon the need to model a business process. In addition, you can carry out the following tasks:

- Link document types to a BPMN construct
- Link the business process model components such as AppWorks Platform application services, Generic application services, email services, roles, and so on, to an activity in the business process diagram
- Link documents to an activity

A business process model is an executable model. In the business process modeling environment, you can validate a business process model, publish it to runtime, and execute instances of the same.

Flow connectors and message maps

At the core of business process modeling are the tasks themselves:

- Process
- Sub-process
- Tasks

Each of these tasks is graphically depicted using symbols to reflect the hierarchical relationships between them. To show the order of execution of tasks, connect them with flow connectors. Decisions, merges, forks, and joins in the process flow are modeled using decision cases and the flow connectors. A decision is a question that is asked at a point in the process flow.

Processes transform data in your organization. You can model how data is transformed during process flow using message maps.

Swimlanes and modeling events

Modeling business processes involves a large number of people and associated events that are responsible for the working of the processes. The AppWorks Platform business process model editor enables you to specify who does what by placing the events and processes into shaded areas called pools that denote who is performing a process. You can sectionalize these pools into swimlanes. A pool typically represents an organization, and a swimlane typically represents a department within that organization (a swimlane can also represent other things such as functions, applications, and systems). A pool is drawn as a rectangular region horizontally across or vertically down the diagram. A swimlane is a sub-partition within a pool and extends the entire length of the pool. By taking processes and placing them in pools or lanes, you specify who does what; for events, you specify where they occur; and for decisions, you specify where decisions are made and who makes them.

Often, events occur while a particular process is being performed, causing an interruption to the process and triggering a new process to be performed. You can model these intermediate events directly on the process activity to which it is associated.

In an end-to-end value chain, organizations and individuals want to pick best-of-breed components that provide value chain the best value. You can accomplish this using AppWorks Platform to ensure that applications and Web services work together in harmony through orchestrating services. AppWorks Platform provides a powerful augmentation to modeling techniques such as application and system design with the support of BPMN and BPML.

Tabs on business process model editor

The business process model editor in AppWorks Platform provides the following tab views.

Model	The Model tab appears by default when you open the business process model editor. Alternatively, click this tab to model a business process and perform all your design time activities.
--------------	--

Message Map	Use the Message Map tab to create and view the message map for your business process models or the message map for any activity residing within a business process model.
Business Logging	Use the Business Logging tab to enable logging for desired activities within the modeled business process. An activity can have multiple entries of nature information, critical or error. Define them at the level of various events associated with the model.
Message Filter	When using BAM to monitor a process and its activities based only on specific messages, click the Message Filter tab to configure message filters. You can configure message filters only for process specific messages and activities such as Task, Web service, Independent Subprocess, Decision Table and Receive Message, if they exist for the business process.

Some of the high-level tasks that you can perform using the AppWorks Platform business process model editor are:

- [Creating a business process model](#)
- [Designing a business process model](#)
- [Attaching web services to a business process model](#)
- [Attaching a user interface to a business process model](#)
- [Using message maps](#)
- [Setting properties of a business process model](#)
- [Validating a business process model](#)

Creating a business process model

A business process model allows you to model a company's business processes such as sales, purchasing, production, or invoicing.

To create a business process model:

1. [Select a starting point](#) and select  (Business Process Model) to create a business process model.
The Untitled Business Process Model - Business Process Model opens. You can start [Designing a business process model](#) and save it later or save it first and then design.
2. Click **Save**.
The Save Document dialog box opens.
3. Enter the following details and click **Save**.

Name	The name of the document. The name must contain values satisfying the following conditions:
------	---

	<ul style="list-style-type: none"> ■ Begins with a letter or an underscore (_) ■ Contains a letter, number, period (.), or underscore (_) ■ Contains at least one letter or number if the code begins with an underscore <p>Avoid white spaces in the name.</p>
Description	The description of the document. Contains only letters, numbers, and special characters. The special characters permitted are periods (.), commas (,), parentheses (), single quotation marks ('), hyphens (-), and underscores (_).
Save in Folder	Select a folder or project to save your document.

4. [Define runtime security](#).
5. [Publish](#) the business process model.

A business process model is created and added to the project content tree in Workspace Documents (Explorer) view.

Designing a business process model

You can design a business process model using the various BPMN constructs. An icon that is available on the business process modeler toolbox represents each construct. The following steps describe the procedure to add the constructs while designing a business process model.

To design your business process model:

1. Select a starting point and select  (Business Process Model) to create a business process model.
The Untitled Business Process Model - Business Process Model opens. If you have the business process model already opened, perform Step 2.
By default, the grid view is turned on in the business process modeler.
For more information on using the grid view, see [Working in Grid View](#).
2. Drag  (Start event) from the toolbox to the business process modeler.
3. Drag  (Activity) from the toolbox to the business process modeler.
4. To view the toolbox after it is closed, click **Options** and select **Toolbox**.
5. Drag  (Decision) from the toolbox on to the business process modeler if you have any decision branching.
You can add other constructs required for your process flow by dragging them on to the business process modeler. See [Constructs Used in a Business Process Model](#).
6. Link the constructs using the  (connector) from the toolbar.
7. To align the connector text to the connector direction, right-click it and select **Align Text**.

8. To end the process, drag  (End event) from the toolbox on to the business process modeler.

You have successfully modeled a business process.

To accelerate the modeling process and quickly draw constructs:

- See [Easy Ways to Draw Constructs in Design Time](#).

To quickly associate an activity with another document such as a Web Service, User Interface, Business Process Model, Case Model or a Role:

- Click the **Insert** tab and select the targeted document. Alternatively, click the **Workspace Explorer** tab in the modeler and drag-drop the required document on to a construct in the business process model.

To group a set of activities:

1. Use the  construct when you need to group a set of activities.
2. Click on the plus sign of the embedded sub-process to expand and view all the activities that are grouped within.
3. Click on the minus sign of the embedded sub-process to collapse the view.
4. Click **Undo** (CTRL+Z) when you want to cancel the current operation and revert to the previous state.
5. Click **Redo** (CTRL+Shift+Z) when you want to re-incorporate the canceled operation.

After you complete this task:

1. Set the properties of the business process model, attach artifacts (such as Web services, tasks, add roles, assign worklists, create message maps, and set properties for the activities), build and publish the business process model to make it executable.
2. Make sure you have the Process Developer role to publish the business process model and to execute it.
3. [Publish](#) the business process model.

To open the Translation Editor where languages and translations are provided:

Activity description is translatable. Auto-suggest menu is available for the description field for selecting the existing text identifiers or for creating new identifiers.

- Right-click the **Business process model** and click the **Translate** option.

The translated activity name is visible in Process instance manager and Graphical instance view of the Business Process.

Working in grid view

The grid view helps you to give a neat appearance and makes your model more appealing. It helps you to avoid the cluttered appearance and provides greater clarity for viewing.

- The grid view feature is available for Business Process Models, OSD (Organization Structure Diagram) and Case Management Models as well. The grid features of the model editor are turned on by default.
- Distribute, align and resize are implemented parallel to the grid, but are not necessarily related to it, because they also work with the grid disabled.

To work in grid view:

1. Right-click in the business process modeling environment and select **Show Grid**.
The grid view of the business process modeling environment appears.
2. Right-click in the business process modeling environment and select **Snap to Grid**.
The existing business process model or the business process model that you created neatly aligns to the grid. If you do not select **Snap to Grid**, the BPMN constructs/shapes do not align to the grid.
3. Right-click in the business process modeling environment and select **Align** to align the constructs/shapes.
Constructs/shapes are aligned based upon the selected option. For details on selection options, refer to Align Options in the topic [Menu Options in Model Editors](#).
4. Hold the Shift key and select the (target) shape whose size you want to change. Then, select the base shape whose size is to be applied to the (target) shape, right-click the base shape and select Resize. Constructs/shapes are resized based upon the selected option.
For details on selection options, see Resize Options in [Menu Options in Model Editors](#).

For example, suppose Shape A is at the top of the business process model workflow and Shape B is just below Shape A. Shape A is 80 pixels in size and Shape B is 50 pixels.

To resize Shape B to the same size as that of Shape A:

1. Hold the Shift key and select Shape B.
2. Select shape A, right-click on it and select **Resize > Make Same Size**. Shape B is resized to the same size as that of Shape A.

To continue working in grid view:

3. On the keyboard, hold Shift key to select the required shapes in the modeling environment right-click and select **Distribute**.
The space between selected shapes is distributed based upon the selected option.
For details on selection option, refer to Distribute Options in the topic [Menu Options in Model Editors](#).
4. Select a shape and use the mouse to move/drag it to required area/direction. The shape is moved to the required area/direction. Alternatively, hold Shift key and select a shape to move it in required direction one pixel at a time.
5. On the keyboard, hold the CTRL key and select any two shapes in the modeling environment to connect them in the desired direction.
The selected shapes are joined by a connector. For example, to connect from top to

bottom first, select the shape on top followed by the shape at the bottom.

All connectors can have description and properties.

Menu options in Model Editors

Align options

Left	Align the selected shapes using the left border of the shape.
Center	Align the selected shapes using the center of the shapes.
Right	Horizontally align the selected shapes using the right border of the shape.
Top	Vertically align the selected shapes using the top border of the shape.
Middle	Vertically align the selected shapes using the center of the shape.
Bottom	Vertically align the selected shapes using the bottom border of the shape.

Resize options

Left	Horizontally align the selected shapes using the left border of the shape.
Center	Horizontally align the selected shapes using the center of the shapes.
Right	Horizontally align the selected shapes using the right border of the shape.
Top	Vertically align the selected shapes using the top border of the shape.
Middle	Vertically align the selected shapes using the center of the shape.
Bottom	Vertically align the selected shapes using the bottom border of the shape.

Distribute options

Vertical Space	Distribute the vertical space between the shape borders uniformly across all selected shape.
Vertical Center	Distribute the vertical space between the shape centers uniformly across all selected shape.
Increase Vertical Space	Uniformly increase the vertical space between all selected shapes.
Decrease Vertical Space	Uniformly decrease vertical space between all selected shapes.
Horizontal Space	Distribute the horizontal space between the shape borders uniformly across all selected shapes.
Horizontal Center	Distribute the horizontal space between the shape centers uniformly across all selected shapes.

Increase Horizontal Space	Uniformly increase the horizontal space between all selected shapes.
Decrease Horizontal Space	Uniformly decrease the horizontal space between all selected shapes.

Setting properties of a business process model

When designing a business process model, set its properties in line with your business needs. This helps you to achieve intended orchestration at the time of executing your business process.

To set the properties for your business process model:

1. Click  (Business Process Model) in the Workspace Documents (Explorer) view <Project>.
2. Right-click the business process model whose properties you want to set and select **Properties**. Alternatively, double-click in the business process modeler or press the keyboard shortcut F8 to view the properties.
The Properties - Business Process Model - <Description> pane appears.
3. Click the **General** tab to configure execution priority, execution mode, business calendar, enable crash recovery, or create a contract.
4. Click the **Business Identifiers** tab to add business identifiers for the business process model.
Adding business identifiers helps you to identify the process in the Process Instance Manager pages during runtime. This is especially useful when multiple processes are running simultaneously, processes with similar names and when there are multiple processes running in a single project.
5. Click the **Monitoring** tab to configure monitoring settings for the process.
Doing this gives you the advantage of monitoring your business process during runtime and helps to quickly track the status of the business process and identify the problem areas.
6. Click the **Attachments** tab to define the type of documents that can be handled by the process.
7. Click the **KPI** tab to add KPIs to the business process models.
KPIs allow users to monitor process related information and trigger business actions and outbound events as specified.
8. Click the **Namespaces** tab to check the various prefixes and their namespaces defined in the business process model.
Namespaces are a set of rules that determine how network resources are named and identified.
9. Click the **Annotation** tab to provide additional or specific notes or comments on the business process model that you would like to use long after you have designed the business process.

The properties of the business process model are set to achieve intended orchestration.

Using logging service in a business process

When you want to monitor a business process activity and its associated data during orchestration, you need to model multiple log entries per event for an activity using the Business Logging tab available in the business process model editor. You can log entries on activities such as a Web service call, human task and an independent sub-process call or a complex activity such as For Each, Until, While, and Transaction. In addition to using the logging service you may also monitor your business process instances from the Process Instance Manager.

Each log entry is associated with a life cycle event of an activity (For example - onStart, onComplete, onWaiting, onResume). See the *AppWorks Platform Administration Guide*, topic Composite Application Logging for more information.

Following events are possible for activities:

- On Start - Before starting to execute the activity.
- On Complete - On successful completion of an activity.
- On Error - When a SOAP fault occurs.
- On Timeout - When the activity times out.
- On Waiting - When the activity goes into waiting.
- On Resume - When the activity is resumed after waiting or after an abort.
- On TransactionStart - When a WS-AppServer transaction starts.
- On TransactionCommit - When a WS-AppServer transaction is committed to database.
- On TransactionAbort - When a WS-AppServer transaction is aborted.

Notes:

- On Start, On Complete, On Error, On Waiting, and On Resume events are specific only to activities and also to groups such as For Each, While, Until, and Transaction (but not for the Context group).
- On Timeout event is specific only to the time-out activity.
- The On TransactionStart, On TransactionCommit and On TransactionAbort events are specific only to the WS-AppServer transactions.

To define log messages for an activity:

1. Select a starting point and click  (Business Process) to open a business process model.
The business process model appears in the business process modeler.
2. Click the **Business Logging** tab.
The Log Message Builder appears with a consolidated view of all activities.
3. Select **<specific activity>** from the list and expand the **<specific activity>** groupbox in the consolidated view.
The activity view of events appears for the selected activity.

4. Click  in the <specific activity> groupbox.
An empty grid view with Event, Type, Mode, and Message fields appear.
5. To define the log message, do all of the following:
 - a. From the Event list, choose the required event for the activity.
 - b. From the Type list, choose the required message severity type for the activity. The message severity type can be Debug, Information, Warning, Error, and Fatal. The properties at the Business Process Management Service container takes precedence over the message severity defined here. So, only the level that is given in the Business Process Management Service is used for logging specified in its properties in the Log Settings tab. However, if you want a specific level at this place, do change the log levels at the Business Process Management Service also.
 - c. From the Mode list, choose the required mode for the activity. Mode can be either static or dynamic. If you choose mode as static, you need to provide a log message if you know the exact parameters of the mode and the same is logged. If you do not know the exact parameter of the mode, it is recommended that you select the mode as Read from Message. When you choose the mode as Read from Message, you must provide an XPath expression and the value of that XPath expression is then logged.
 - d. Provide required message for the activity in the Message field. It helps the user when a meaningful message is specified based upon the severity type and the mode that you selected.
6. Click **Save**.
7. Publish and execute the business process. The log messages are published to the default central log database.

You have successfully defined log messages for an activity. You can now view the logged messages at the default central log database.

Defining log messages for an activity

To define log messages for an activity:

1. Select a starting point and click  to open a business process model.
The business process model appears in the business process modeler.
2. Click the **Business Logging** tab.
The Log Message Builder appears with a consolidated view of all activities.
3. Choose <specific activity> from the list or expand <specific activity> Groupbox in the consolidated view.
The activity view of events appears for the selected activity.
4. In the <specific activity> Groupbox, click .
An empty grid view with Event, Type, Mode, and Message fields appears.

Event	Choose an event for the activity.
Type	Choose the message severity type: Debug , Information , Warning , Error , and Fatal . The properties in the Business Process Management Service container take precedence over the message severity defined here. Only the level that is given in the Business Process Management Service is used for logging specified in its properties in the Log Settings tab. However, if you want a specific level at this place, change the log levels in the Business Process Management Service service.
Mode	Choose a mode for the activity. Mode can be static or dynamic. If you select static , provide a log message if you know the exact parameters of the mode and the same is logged. If you do not know the exact parameter of the mode, select the mode as Read from Message . You must provide an XPath expression and the value of that XPath expression is then logged.
Message	Provide a message for the activity. Provide a meaningful message that relates to the severity type and mode.

5. Click .
6. Publish and execute the business process.
The log messages are published to the default central log database.

You have successfully defined log messages for an activity. You can now view the logged messages at the default central log database.

Associating business identifiers to a business process model

You can associate business identifiers that are available across the current workspace. Through the option of [Runtime references](#), you can also reuse business identifiers that are deployed as part of other applications.

A single business process model can have multiple business identifiers associated. Of the associated identifiers, only four identifiers are shown as separate columns in Process Instance Manager. It is possible for users to select this sub-set of identifiers. Open Text recommends that you select the identifiers that are most frequently needed for direct viewing of the values when instances are displayed. Before you begin, create and design a business process model and Business Identifiers.

To associate business identifiers to a business process model:

1. In Workspace Documents (Explorer)> <Project>, click  (Business Process Model) to open the required business process model.
The model appears in the business process modeler.

2. Right-click in the business process modeler and select **Properties**. Alternatively, double-click in the business process modeler or press the keyboard shortcut F8 to view the properties.

The Properties - Business Process Model - <Description> pane appears.

3. Click the **Business Identifiers** tab to add business identifiers for the business process model.

All the available business identifiers in the workspace can be associated to the business process model. You can also create new business identifiers by clicking New on the toolbar and associate them to the process or associate the existing business identifiers per your need.

All the associated business identifiers are shown in the message map, just as the process specific messages. You can assign values to business identifiers. Business identifiers cannot be used as source elements in the message map.

You have successfully associated the business identifier to a business process model and used the message map to assign the values. For each instance that is created from this model, instances of the business identifiers too will be created to enable the filtering of instances in the Process Instance Manager.

After you complete:

- You can [search for instances in Process Instance Manager](#) using Business Identifiers.

Creating a KPI on a business process model

In AppWorks Platform, you can create and configure the Key Performance Indicators (KPI) on a Business Process Model in any of the following ways:

- Create a KPI using Business Measure.
- Create a KPI directly on a Business process model

Before you begin:

- [Creating a business process model](#) and [Designing a business process model](#) a business process model.
- Ensure that Business Activity Monitor service container is started in your current organization.

To create a KPI:

1. In the Workspace Documents > <project>, right-click the required <business process model> and select **Model Properties**.
2. Select the KPI tab and click  icon on the tool bar of the table to launch KPI Wizard.
3. Specify the name, description, and the goal that describes the purpose and objective of the KPI in the Name, Description, and Goal fields, respectively.
4. In the Define Objectives area:

Unit of Measure	Choose the unit of measure
Target Value	Enter a target value
Range limits	<p>Click  to add range limits.</p> <p>For each range limit, specify a name and enter the lower and upper limits in the respective columns.</p>

5. Click **Next**.
6. In the **Build Metric** area, build the metric by choosing the aggregate function on an attribute that was defined using a Message filter. Based on the parameters selected and filters applied, the KPI Wizard automatically generates the business measure. Do the following:

Define Aggregation

Aggregation type	Select the Aggregation type: Sum, Count, Average, Min, Max
Attribute	Select the required Attribute. These attributes include the standard attributes as well as the Message Filter elements defined in the business process model.
Alias	Type an Alias name for the selected attribute.

Define Filters - Create a filter that is based on the parameters configured in the Aggregate definition. You can define multiple filters.

- | | |
|--------------|---|
| Add a filter | <ol style="list-style-type: none"> 1. Click  to add a filter. 2. Select the required Attributes. 3. Select operators: =, <>, >, <, >=, and <= 4. Provide values for the attributes and click Next |
|--------------|---|

Configure Time frame

	<ol style="list-style-type: none"> 1. Select a date-time attribute 2. Select Time-frame window type: Static or Rolling.
From	Select From date.
To	Select To date.

Expression

- | | |
|--|---|
| | <ol style="list-style-type: none"> 1. Build the necessary expression. 2. If the KPI should be triggered on a periodical basis to build trend analysis, select the Define Schedule check box and click Next. |
|--|---|

Schedule - Configure the frequency of interval at which the KPI should be invoked

Auto Deploy	Automatically deploy the schedule to run-time when the KPI is published.
Schedule Type	Choose a frequency and configure the duration.
Target Document Type	<ol style="list-style-type: none"> 1. Choose the required Target from the Business Process Model and Web service document types. 2. For Custom XML document type, provide the text for Request XML.
Define Actions	<ol style="list-style-type: none"> 1. Select Define Actions. Additionally, you can also select Model Email that can be triggered as an action check box if you want to model an e-mail template and send e-mail as part of the action 2. Click Next.
Email Model	Select Email Model to configure an email model on the KPI.
Conditions and Actions	Select Conditions and Actions to configure the required conditions and actions. When on a scheduled interval, if the specified conditions are met, the appropriate actions (Triggering Business process, Web Service, or sending Notifications) are triggered.

You have successfully created KPI on a Business Process Model.

The KPIs created are displayed in the explorer view, as a child node of the process or the Activity on which the KPI is created. However, if a KPI is created on constructs such as Send Message, Receive Message, Start, End, Timeout, and so on, which are not shown in the explorer view, then the KPIs created on these constructs are not displayed in the explorer view. Therefore, if you have to use these KPIs, select the Insert > Composite Control and other option on the User Interface toolbar and select the relevant KPI.

Modifying the business process management service properties

Modify the properties of the Business Process Management Service container if you want to enable business logging, create a connector for the admin database, process engine database, data transformation, scheduler, and to query process instance data.

Before you begin:

- Create the Business Process Management Service Container.

To modify the business process management service properties:

1. On the Welcome page > My Applications, click  (System Resource Manager). The System Resource Manager window opens with the available Service Containers in

the Service Containers App Palette.

2. On the **Properties - Business Process Management AppPalette**, click the **Business Process Management** tab.
The default view of the Admin Database tab appears. The Process Engine and Scheduler and Query Process Instance Data tabs also appear.
3. Modify details as necessary in the following tab views:
 - Admin Database tab view
 - Process Engine tab view
 - DataTransformation tab view
 - Scheduler tab view
 - Rule Repository tab view
 - Binary Parser tab view
4. Click .
5. Right-click the **Business Process Management** service and select **Restart**.
The [Business process management service properties interface](#) is restarted.

You have successfully modified the properties of the Business Process Management Service.

Business process management service properties interface

The Business Process Management Service properties interface helps you to create a connector for the admin database, process engine database, CoBOC repository, and scheduler settings for the Business Process Management Service container. See [Service container configuration interface](#) for information on the settings of the respective tabs in the service properties interface. This topic describes various fields in the Business Process Management tab.

To specify Log4J settings for business logging feature:

1. In the **Log Settings** tab view, clear the **Use System Policy** check box and select the **Enable Logging** check box.
This enables the log category related fields in **Logger Severities**.
2. In the **Category** column of the tabular view of logger severities, select **AppWorks Platform BPM Business Logging** category and select Debug, Information, Warning, Error and or Fatal severity check boxes as required.
This ensures that only the specified type of severity is picked from the business logging tab irrespective of the severity type that you select at the time of business logging.
Logger severity indicates the impact of an error on the execution of the business process model through the log message.
3. At times, during the execution of a business process that has `ForEach`, `While`, or `Until` loops in it, upon reaching the maximum iteration limit, the iterations are not suspended and continue into an endless loop. To avoid this situation, it is necessary to suspend iteration when the maximum iteration limit is reached, and send an alert message to the

Administrator. To accomplish this, pass the property `cordys.bpm.runtime.maxiterationcount` as a JVM argument. In the **JRE Configuration** tab of the Business Process Management Service properties, expand **Set the Parameters** Groupbox, click and pass the property `cordys.bpm.runtime.maxiterationcount` as a JVM argument in the JVM Property field, and click .

If the Administrator resumes the process, the process will continue with its iterations until it executes the loop for maximum iterations specified in the JVM argument.

Through the **Query Process Instance Data** tab you can create an application connector for accessing process instance data. This tab displays the database connection details that were used during the creation of the application connector.

To create an application connector, you must have the role of Process Administrator. The advantages are:

- When the Process Administrator operates on process instance data, it does not interfere with operations of the BPM SOAP processor and thus improves performance.
- The end user can have multiple BPM SOAP processors for different users or organizations, but there can be only one Query Process Instance Data Processor that meets the requirements of the Process Administrator.
- When either of the SOAP processors shuts down or fails to start, it does not affect the other processor.
- The end user can create an application connector with read-only credentials to the PIM data.

Process Instance Data tab

Database Configuration Description	A description for the database configuration.
DB User	A user name to access the database. Provide a user name. This authentication will work only with SQL server and OLEDB configuration. If you want to connect to the SQL server with Windows authentication, leave the User and Password fields blank.
Support Multibyte	If this option is selected, multibyte characters, such as Chinese alphabet, are supported.
Support special characters in XML	If this check box is selected, special characters are supported in XML. The encoding or decoding of special characters in XML takes some time and slows down the performance of the machine. Therefore, users should clear this option if the database does not have any table or fields with special characters.
Set precedence to NULL	Select this check box to set the database value to null in the following conditions:

	<ul style="list-style-type: none"> ■ When you use templates for Web services that have an update request in a business process model. ■ When you set null as true in the update request and also data for the same field(s). ■ When there is a need for the database value to remain null If this field is not selected, the data will be stored in the field.
Default Database	The default database that will be accessed. Specify a database.
Select Database Configuration	The location where the database is configured.
Driver	The name of the database driver used.
Provider	The name of the database provider.
Database Vendor	The name of the database vendor.
Organization	The name of the organization - system.
Server/Instance Name	The name of the server or the instance where the database is available.
Size	The number of cursors to be cached.
Read	<p>The number of minimum and maximum read connections dedicated for this service.</p> <ul style="list-style-type: none"> ■ Minimum: The minimum value is 1. ■ Maximum: The maximum value is 10.
Update	<p>The number of minimum and maximum update connections dedicated for this service.</p> <ul style="list-style-type: none"> ■ Minimum: The minimum value is 1. ■ Maximum: The maximum value is 10.
Size	The number of queries to be cached.
Refresh Interval	The refresh rate for the cursor cache.
Query Cache	
Cursor Cache	
Database Connections	

The Business Process Management Service can share a single Operating System (OS) process with the following services, provided all processors sharing this OS Process point to the same CoBOC database.

- Admin Database
- CoBOC Service

- Rule Repository Service
- Scheduler Service

The Rule Repository tab appears only when you select the Enable Rule Engine check box. The Binary Parser tab appears only when you select the Enable Binary Parser check box in the Process Engine tab.

Note: When the Business Process Management service is started, the active schedules are loaded asynchronously by default. However, if you want the active schedules in the Business Process Management service to be loaded synchronously, you must set the WCP property `com.cordys.scheduler.engine.load.async` to `false`.

Admin database, Data Transformation, Scheduler and Rule Repository tabs

Select Database Configuration	When you select an existing database configuration, the admin database is pointed to it otherwise, if you select a new database configuration, the New Database Configuration window appears. Point the admin database to the new database configuration. Select an existing database configuration from the list or select a new database configuration.
Database Configuration Description	Describes the database configuration for running AppWorks Platform. This is a read-only field.
Driver	Based up the database configuration you selected, this field displays the database driver name. For more information, see JDBC Interface Details in the AppWorks Platform Administration Guide.

Advanced Properties

Set precedence to NULL	When there are Web services with an update request in a business process model, templates are often used by the user. In such a scenario, select this option to set the database value to null. In an update request, if the NULL attribute is set to true and data is specified, the database value will still be null.
Support special characters in XML	If you want to encode or decode special characters in XML, select this check box to enable support for special characters in XML. The encoding or decoding of special characters in XML takes some time and slows down the performance of the machine. Therefore, you must clear this option if the database does not have any table or fields with special characters.
Support Multibyte	Based up your business requirement, you may want to support other languages such as the Chinese alphabet. Select this option to support multibyte characters.
Share Connection Pool	Select this option to enable applications using AppWorks Platform to share same connection pool for multiple components.

Query Cache

Size	Displays the number of queries to be cached.
Refresh Interval	Displays the refresh rate for the query cache.
Cursor Cache	
Size	Displays the number of cursors to be cached.
Refresh Interval	Displays the refresh rate for the cursor cache.

Database Connections

Read	<p>The number of read database connections dedicated for this service. Displays the Minimum and Maximum number of database connections.</p> <ul style="list-style-type: none"> ■ Minimum: The minimum value is 1. ■ Maximum: The maximum value is 10.
Update	<p>The number of read database connections dedicated for this service. Displays the Minimum and Maximum number of database connections.</p> <ul style="list-style-type: none"> ■ Minimum: The minimum value is 1. ■ Maximum: The maximum value is 10.

Runtime Properties - Visible only for the Rule Repository tab

Max. Value for dispatcher queue size	<p>Maximum number of asynchronous rule actions that can be queued in the dispatcher queue. The default size is 50000.</p> <p>Setting the value to 0 enables an infinite number of rule actions in the queue and may result in Out of Memory error. Depending on the application's requirements, change the size of the dispatcher queue. Higher value of the parameter allows for higher memory and lower value limits memory usage It is exposed to JMX as a cold setting.</p>
Rule Action Threads	<p>Number of threads that are assigned for a rule action. If the number of transactions performed by the rule engine involving external actions is large, increase the size of the rule action threads. The default size is 5.</p>
Trace Rule Execution	<p>When you select this option, tracing of rule execution is enabled. The status of the rules can be monitored from the CoBOC Transaction Monitor.</p>

Advanced options

This tab appears only in the Data Transformation tab view.

Hashbins	A memory store to hold business objects. Type the number of the stores available for maintaining the cache.
----------	---

	<ul style="list-style-type: none"> ■ Minimum value is 1. ■ Maximum value is 10. ■ Default value is 10.
Initial Capacity	<p>The minimum capacity of each hashbin (memory store) to store business objects. Type the initial capacity of each hashbin in terms of the number of business objects it can hold.</p> <ul style="list-style-type: none"> ■ Minimum value is 100. ■ Maximum value is 1000.
Object Idle Time (in Mins)	<p>The maximum time (in minutes) the cache instances can remain idle in the cache.</p> <ul style="list-style-type: none"> ■ Minimum value is 1. ■ Maximum value is 60.

Process Engine tab

Thread Pool for Long Lived Processes	Specify the number of business processes that can run simultaneously. An increase in the number of threads increases the number of processes running in parallel, but at the cost of consuming greater memory. The default value is 5.
Process Model Cache Size	Specify the number of business process models that can be cached. The default value is 200.
Client Connection Group	Displays the current connection point group. Click to select a connection point group and click OK in the Connection Point Groups window to send SOAP requests to Service Groups (SOAP nodes).
Enable Process Monitoring	Enable process monitoring at the SOAP Processor level. This overrides all settings made at the process and activity levels.
Enable Crash Recovery	Resume processes from the point where they had stopped (for example, due to Service Containers failures and so on).
Activate Debug Points	<p>Debug points are usually defined on activities at the time of designing a business process model. This is done to identify and suitably address any bottlenecks that the process may encounter at the time of execution. When the debug points are activated at the Business Process Management Service Container level, it becomes possible to identify the debug points during runtime.</p> <p>When a debug point is encountered, the business process goes into a Debug Ready state in PIM. You may then choose to either debug the process instance activity by activity to identify and address the bottleneck or resume the process instance. Activate Debug Points remains enabled by default. However, you may choose not to enable this feature, which prevents you from identifying the debug points even if they are enabled or selected at activity level in the</p>

	business process model. Select this option to activate debug points defined in the process model.
Enable Binary Parser	Initialize the Binary Parser in an embedded mode. When selected, a Binary Parser tab appears in the Business Process Management window, where you can configure the Binary Transformation Channel.
Enable Priorities	Execute business process models based on priority. The Type of algorithm list appears when you select Enable Priorities. Priorities can be set while modeling a process (See Business Process Model Properties pane), or at runtime (See ExecuteProcess API in the AppWorks Platform API Reference Guide).
Auto recover process instances	Resume processes in case of a database connection failure. When set to true, the processes in running or queued state will resume.
Enable Rule Engine	Enable support for decision tables in business process models. The Rule Repository tab view appears where you can modify the database configuration details suitably.
Enable Data Transformation	Provide the CoBOC configuration with which the Data Transformation is configured. The Data Transformation tab appears. Provide the CoBOC configuration details. These details are used by the engine in an embedded mode, to execute the transformations if the XSLtransform Web service is used in the business process model.
Type of Algorithm	<p>This list is available when you select Enable Priorities. From this list, select the priority algorithm that you want to apply for executing processes, and specify the required information.</p> <p>The following list describes the types of algorithms that you can use to execute processes.</p> <ul style="list-style-type: none"> ■ Aging Algorithm: Processes with the highest priority are executed first, followed by the ones with lower priority. If you: <ul style="list-style-type: none"> • Want to change the priority of process instances periodically to a higher priority so that all processes in the priority queue can be eventually executed, select the Allow time based promotions check box. Specify the frequency (in seconds) at which the priority of the process instance will be assigned a higher priority in the Aging Time box. • Do not want to change priorities of process instances in the priority queue, clear the Allow time based promotions check box. ■ Pseudo time slicing: A set of processes are executed from each priority in the priority queue until all the processes have finished executing.

	<ul style="list-style-type: none"> • Stage 1: Five processes of high priority will execute. • Stage 2: Four processes of above normal priority will execute • Stage 3: Three processes of normal priority will execute. • Stage 4: Two processes of below normal priority will execute. • Stage 5: One process of low priority will execute. • Stage 6: Stage 1 to Stage 5 is repeated until all the processes execute. <p>■ Custom: In Implementation Class, type the fully qualified Java class name that will handle the execution of priority-based processes.</p>
--	--

Statuses of a process instance

The Process Engine is responsible for instantiating a process model. During its life cycle, the process instance passes through several statuses.

The following table describes the statuses of an individual process instance.

Status	Description
Aborted	Process instance that is aborted. A process instance can be aborted due to some error, for example, failure to invoke a Web service.
Complete	Process instance that completed its cycle. A process instance is successfully completed when it reaches the end of the process.
Debug	Process instance that is executed in debug mode through the Process Debugger.
Debug Ready	Process instance that has reached a breakpoint. It can be continued through the Process Debugger or by selecting the Resume option.
Running	Process instance that is currently running. For example, when the process instance is waiting for the response from a Web service activity.
Replaced	Process instance that is resumed after replacing the sub process. When an activity in the main process instance is reset, only the activity can be resumed.
Restart	Process instance that is restarted after it was aborted. For more information, see Restarting a Process Instance from Graphical Mode in the <i>AppWorks Platform Administrator's Guide</i> .
Reset	Process Instance that is executed again after it was suspended. For more information, see Resetting a Process Instance from Graphical Mode in the <i>AppWorks Platform Administrator's Guide</i> .

Status	Description
Suspended	Process instance that is suspended. Activities in a process instance cannot be executed until the process instance is resumed.
Terminated	Process instance that is terminated by the process administrator.
Waiting	Process instance that is waiting for a response from a manual activity or an incoming message.

Configuring BPMN constructs

BPMN constructs are used to model business processes. Each construct has corresponding properties, which can be configured.

The BPMN constructs in the AppWorks Platform business process modeling environment are as follows:

- [Setting the properties for Start event](#)
- [Setting the properties of a Decision construct](#)
- [Setting the properties of an End event](#)
- [Setting the properties of an Activity](#)
- [Setting the properties of Receive Message Event](#)
- [Setting the properties of a Compensate event](#)
- [Setting the properties of a Delay event](#)
- [Setting the properties of a Catch Exception event](#)
- [Setting the properties of an independent subprocess](#)
- [Setting the properties of Time-out](#)
- [Setting the properties of a Swimlane](#)
- [Setting the properties of an Annotation](#)
- [Setting the properties of While construct](#)
- [Setting the properties of Until](#)
- [Setting the properties of For Each](#)
- [Setting the properties of a Transaction](#)
- [Setting the properties of an Embedded Sub-process](#)

Setting the properties for Start event

A business process model should have only one Start event to indicate the start of the process. A Start event has at least one outgoing connector and does not have any incoming connectors.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Start event in the business process model. Alternatively, right-click the Start event and select **Properties**.
The Start Event Properties - Start pane appears. See [Start Event Properties Interface](#).
3. Click the **General** tab and provide a **Description** for the Start event.
4. Select the **Trigger Type** for the Start event.

Message	Drag an activity input message or Process Specific Messages from Workspace Documents (Explorer View) > <Solution> > <Project> to the Start event.
Timer	Create a scheduler before or after selecting this trigger type. <ul style="list-style-type: none"> ■ Select Timer. ■ From the Workspace Documents (Explorer View) > <Solution> > <Project>, drag the business process model on to a scheduler to initiate the business process flow.
No Message or Timer	No input message or time is required to start the process.

5. To enable or disable monitoring and store recovery point for the activity, click the **Monitoring** tab.
This tab is visible only if the selected Trigger Type is Message.
6. Click the **Annotation** tab and type additional comments or notes on the Start event, if any.

You have successfully set the properties for the Start event.

Setting the properties of a Decision construct

Decisions are BPMN constructs. They function as gateways in a business process, where the flow control can take one or more alternative paths. A decision has at least one incoming connector and two or more outgoing connectors. Decisions can be broadly categorized into data-based decisions and event-based decisions.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Decision construct in the business process model. Alternatively, right-click the **Decision construct** and select **Properties**, or press F8 on the keyboard. The <Decision Name> - Decision Properties pane appears. It displays the **Conditions**, **Monitoring**, and **Annotation** tabs.
For information about the options available in the <Decision Name> - Decision Properties pane, see [Decision Construct Properties Interface](#).
3. Click the **General** tab, type a **Description**, and select the Font Size for the decision construct.
4. To enable the exclusive OR (XOR) feature in the decision construct, select **Use Exclusive Gateway**.
5. To specify the condition name, type, and XPath value of the decision construct, click the **Conditions** tab.
6. To enable or disable monitoring of the decision construct, click the **Monitoring** tab.
7. Click the **Annotation** tab and type additional comments or notes on the decision construct.

You have successfully set the properties of the Decision construct.

Setting the properties of an End event

Every business process should have at least one End event. An End event has at least one incoming connector and does not have any outgoing connectors.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the End event activity in the business process modeling environment. Alternatively, right-click the End event and select **Properties**. The End Event Properties - End pane appears. See [End Event Properties Interface](#).
3. Click the **General** tab and type a **Description** for the End event.
4. Select the **End Type** to end the process.

Rollback	<p>Roll back the process.</p> <ul style="list-style-type: none">■ If a sub-process is rolled back, all Compensate activities defined in the sub-process are executed in a bottom to top sequence.■ If a sub-process includes a nested sub-process with Compensate activities defined, only this nested sub-process is rolled back.■ A rollback from the Process Instance Manager is not possible.
----------	---

Terminate	Terminate the business process.
Abort	Abort the business process.
None	The business process ends without a message, error, or rollback of the process.
Message	Drag an activity output message or Process Specific Messages from Workspace Documents > <Solution> > <Project> tree on the End event.
Error	Click the Error details tab, and type the error details.

5. To enable or disable monitoring and store recovery point for the activity, click the **Monitoring** tab.
This tab is visible only if the selected **Trigger Type** selected is **Message** or **Error**, or **Rollback**.
6. Click the **Error details** tab, and type the **Error Message** and the **Error Detail** for the End event.
7. To throw a custom error, you add the following XML:

```
<CustomErrors>
<Error>
<Code>101</Code>
<Message>My Custom Message</Message>
<Detail>My Custom Detail</Detail>
</Error>
</CustomErrors>
```

8. Click the **Annotation** tab and type additional comments or notes on the End event, if any.

You have successfully set the properties for the End event.

Creating a custom error

Every method (SOAP request) can have its own error codes (SOAP faults). To model errors in a business process, the End event has to be configured in *Properties - End Event* pane, where the **End Type** property should be set as **Error**. The code for the error can be custom-defined or predefined. Use a process error for exception handling when the process is a sub-process in another process. The sub-process can have a link to an exception handling event in the main process. Creating a custom error involves creating the custom error for a sub-process and mapping this to the custom error in the main process.

While creating a custom error, select an error code that is unique, and that does not match a predefined error code. If you do not select a unique error code, the predefined error code will be thrown.

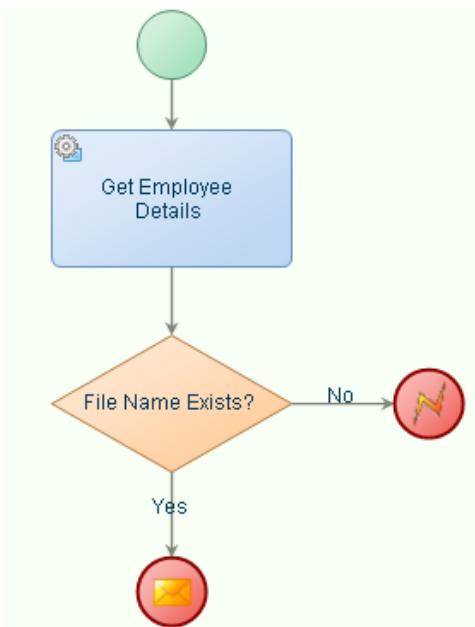
To create a custom error:

1. Double-click the End event in the AppWorks Platform business process modeling environment. Alternatively, you can right-click the End event and select **Properties** from the menu.
The *End Event Properties - End* pane appears. For more information, see [End Event Configuration Interface](#).
2. Click the **General** tab, enter a **Description** for the End event, and select **Error** from the **End Type** list.
3. Click the **Error details** tab, enter the **Error Code**, **Error Message** and, **Error Detail**.

A custom error is created.

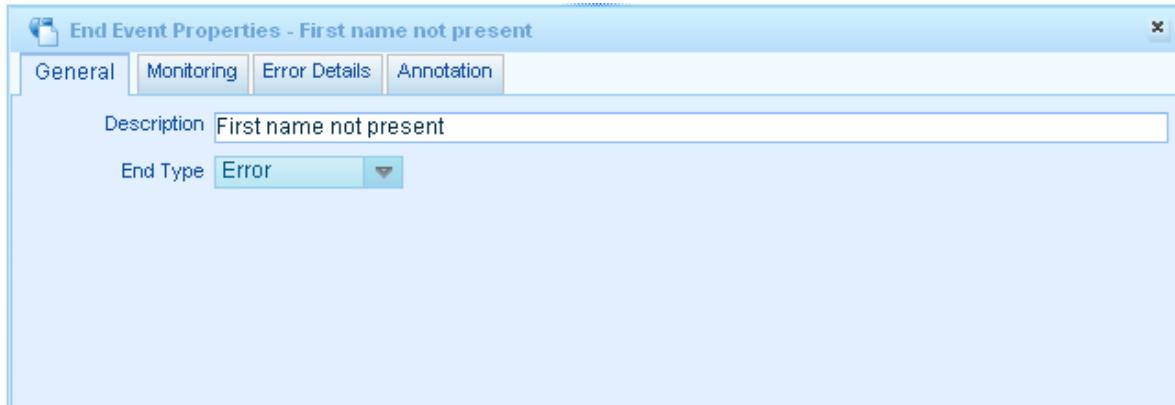
Example of creating a custom error

1. In a business process, Get Employee details, the GetEmployee activity fetches employee details from the database, and returns the First name of the employee if it exists; else an exception is thrown.

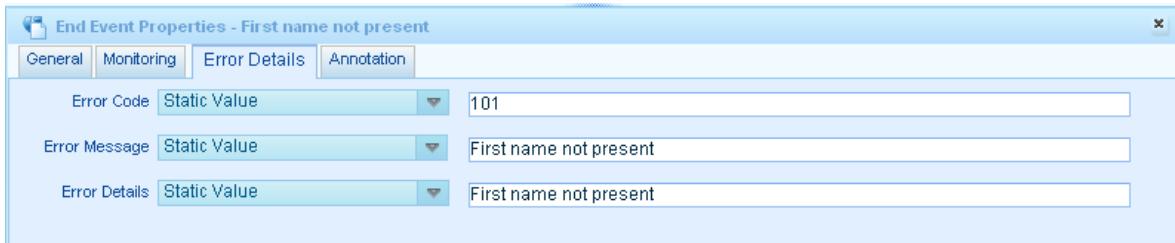


2. If no value is found for the First name of the employee, the process ends with an exception.

The *Properties - End Event* pane for this example is shown.



3. Type the description of the error in the **Description** field and select **Error** as the **End Type**.



4. In the **Error Details** tab,
- Choose **Static Value** from the list.
 - Enter 101 in the text box next to **Error Code**.
 - Enter First Name Not Present in the text boxes next to **Error Message** and **Error Details** fields.
5. After you complete this task, you must map the custom error to an Exception event in the main process.

Using the custom error

Before you begin, create the custom error for a sub-process and map to the **Description** and the **Error Code** of the Exception event in the main process.

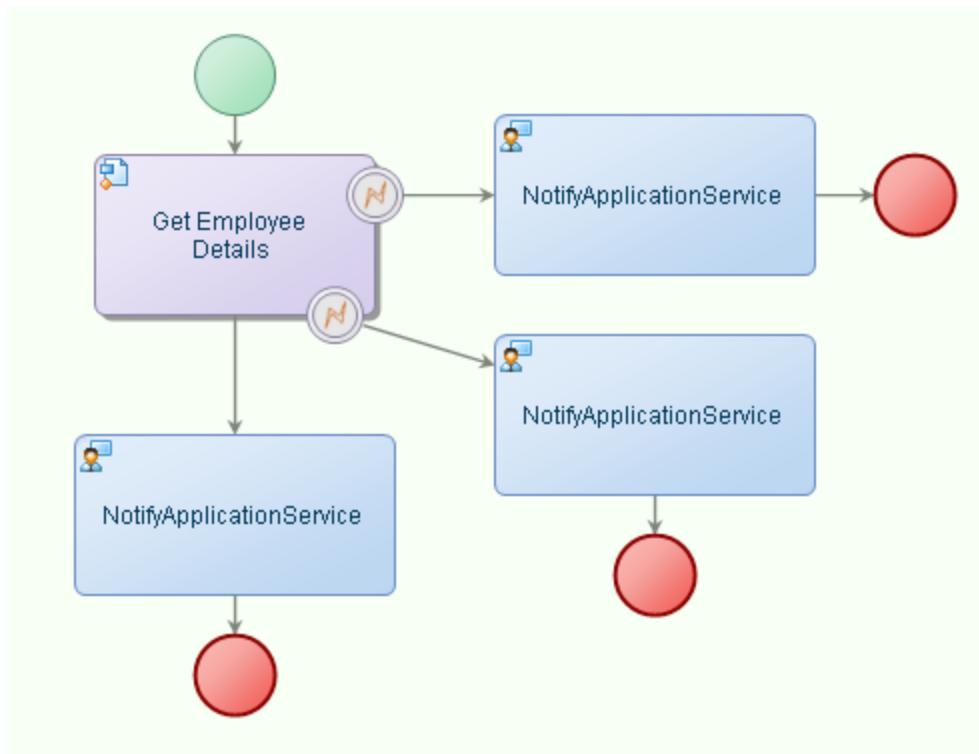
To use the custom error:

1. Select a starting point and click (Business Process Model) to open an existing business process model.
2. Double-click the **Exception** event.
The *Exception Event Properties - Exception* pane appears.
3. Click the **General** tab and type the same **Description** that you used while creating the Custom Error.
4. Click the **Error details** tab and type the same **Error Code** (Process Error) that you used while creating the Custom Error.

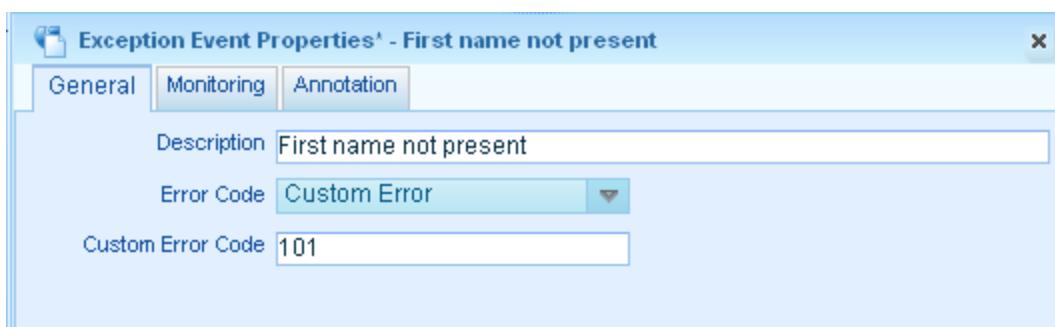
Example of mapping a custom error

The business process modeled below uses the business process model Get Employee Details (see [Creating a custom error](#)) as a sub-process.

Notify Application Service is a task and you have to create it by creating an XForm. In the Get Employee Details sub-process, the First name of the employee is returned if it exists; else an exception (per the example in Creating a Custom Error, 101: First name not filled) is thrown. You can handle the exception using the Exception event. You could also define more exceptions for the sub-process.



The *Properties - Exception Event* pane for this example is shown.



Setting the properties of an Activity

An activity is the smallest unit of a business process that involves an operation. A logical set of activities that are connected and performed in a sequential manner having a start and end point and focus on achieving a common goal form a business process model. While designing a business process model an activity is represented as a BPMN construct, and is a generic term for work that is performed within a business process. An activity is atomic when there is only a single activity to be performed whereas an activity is non-atomic when there are several other activities that it is dependent upon and vice-versa.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Activity whose property you want to set. Alternatively, right-click an activity and select **Properties**, or press F8 on the keyboard.
The < Activity Name > - Activity Properties pane appears.
3. Add a functionality by attaching one of the following to the Activity.

Web service	Use to retrieve or update records from a table without any user intervention. See Attaching Web Services to a Business Process Model.
Decision Table	Use to invoke decision table from the business process model. See Invoking a Decision Table from a Business Process Model.
User Interface or External User Interface	Use this option to attach a User Interface or External User Interface created using an XForm, HTML, or JSP. For information about creating user interfaces, see Working with Web Application. If an activity is not associated with any of the functionality, then the activity is an empty activity. Tabs displayed on the < Activity Name > - Activity Properties pane vary according to the functionality attached.

4. Click each tab of the activity property that appears beneath the business process model, and provide any required information.
For more information on the various properties of an activity and what they imply, see [Activity properties interface](#).

The properties of the current activity are set.

Setting the properties of Receive Message Event

A Receive Message event is a BPMN construct, which is triggered by an incoming message from a participant. The process remains in a waiting state until it receives the specified message and when the message is received, the process flow moves to the next immediate activity.

A Receive Message event can also be used to synchronize a sub-process with its parent process, in which case, the Receive Message is used in the main process.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Receive Message event or alternatively, right-click the Receive Message event and select **Properties**.
The Receive Message - Receive Message Properties pane appears. See [Receive Message Configuration Interface](#).
3. Click the **General** tab and type a **Description** and an **Input Message** for the Receive Message.
4. Select the **Monitoring** tab to enable or disable monitoring for the activity.
5. Click the **Annotation** tab and type additional comments or notes on the Receive Message.

You have successfully set the properties of the Receive Message event.

Setting the properties of a Compensate event

A Compensate event is a BPMN construct, which reverses or rolls back the effects of an operation that has been executed. A Compensate event is triggered when an error is encountered during a process. Compensate event rolls back the actions prior to the occurrence of the error. A Compensate event occurs outside the normal flow of the business process, and is triggered based on the rollback of a transaction.

A Compensate event can be defined for an Activity, Sub-process, For Each, While, Until, and Context BPMN constructs. However, an Activity or Sub-process cannot have more than one Compensate event.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Receive Message event or alternatively, double-click the Compensate event that you want to configure. Alternatively, right-click the Compensate event and select **Properties** from the menu.
The Properties - Compensate Event pane appears. See [Compensate Event Properties Interface](#).
3. Click the **General** tab and type a **Description** for the Compensate event.
4. To enable or disable monitoring of the Compensate event construct, click the **Monitoring** tab .
5. Click the **Annotation** tab and type additional comments or notes on the Send Message Event.

The properties of the Send Message event are set.

Setting the properties of a Delay event

If an Intermediate Event involves a waiting time, then its type is Delay.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the **Delay** event that you want to configure. Alternatively, right-click the Delay event and select **Properties** from the menu.
The Properties - Delay pane opens. See [Delay Event Properties Interface](#).
3. Click the **General** tab and type a **Description** and for the Delay event.
4. To apply a condition for the execution of the Delay event, specify the type in **Execute Condition**.
5. Click the **Duration** tab to specify the calculation of the duration for a Delay event based on the business calendar.
6. Click the **Monitoring** tab to enable or disable monitoring of the Delay activity construct.
7. Click the **Annotation** tab and type additional comments or notes on the Delay event.

You have successfully set the properties of the Delay event.

Setting the properties of a Catch Exception event

A Catch Exception event is a BPMN construct and is used for specific error occurring in the Activity or Context to which it is attached. It reacts to a named error or to any error if no name is specified when attached to an Activity.

To set the properties:

1. In the Workspace Documents (Explorer) view, expand <project> and click <business process model>.
The business process model appears in the business process modeler.
2. Double-click the Catch Exception event that you want to configure. Alternatively, right-click the Catch Exception event and select **Properties** from the menu.
The Properties - Catch Expression Event pane appears. See [Catch Exception Properties Interface](#).
3. Click the **General** tab and type a **Description** for the Catch Expression event.
4. Select an option from the **Error Code** list.
5. Select the **Monitoring** tab to enable or disable monitoring of the Catch Expression event construct.
6. Click the **Annotation** tab and type additional comments or notes on the Catch Expression event.

You have successfully set the properties of the Catch Exception event.

Setting the properties of an independent subprocess

Before you begin:

- You need a parent business process model in which the Independent Subprocess is available.

You may perform this task based on your business needs.

To set the properties:

1. From the **Workspace Documents**, click <project> > <business process model>. The business process model appears.
2. Double-click the Subprocess to set its properties. Alternatively, right-click the Independent Subprocess and select **Properties**.
The Properties - Independent Sub Process pane appears.
3. Click the following tab views and enter necessary details for the Independent Subprocess:
 - General tab
 - Messages tab
 - Monitoring tab
 - Annotation tab
4. Click  on the business process toolbar.

You have successfully set the properties of an Independent Subprocess.

The message structure that is defined in the Independent Subprocess for a Send Message Activity need not be created in the parent process because it is picked directly from the Independent Subprocess.

Setting the properties of a Send Message event

Send Message Intermediate Event is a BPMN construct and is used to send a message from the process to the other.

To set the properties:

1. [Select a starting point](#) and click  (Business Process Model) to open an existing business process model.
2. Double-click the Send Message activity that you want to configure. Alternatively, right-click the Send Message event and select **Properties** from the menu.
The Properties - Send Message activity Event pane appears. See [Send Message Event Properties Interface](#).
3. Click the **General** tab and type a **Description** for the Send Message activity.

4. To specify the **Message to send**, drag a **Process Specific Message** or an Activity output message from the **Message Map** to the Send Message Event in the sub-process or Receive Message in the main process. You can also type the message in the **Message to send** text box.
5. To apply a condition for execution of the Send Message Event, specify the **Execute Condition**.
6. Click the **Monitoring** tab and the **Store recovery data** to enable or disable monitoring of the store recovery point settings.
7. Click the **Documents** tab, and select the documents that you want to link to the Send Message Event.
8. Click the **Annotation** tab and type additional comments or notes on the Send Message event.

The properties of the Send Message event are set.

Setting the properties of Time-out

Time-out event is a BPMN construct that is fired when the Activity or Sub-process is not executed within the specified time. A Time-out event can be defined for an Activity, Group (Context, For Each, While and Until), Receive Message, and Sub-process. An Activity or Sub-process cannot have more than one Time-out event.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Time-Out event that you want to configure. Alternatively, right-click the Time-Out event and select **Properties** from the menu.
The Properties - Time Out Event pane appears. See [Time-Out Event Properties Interface](#).
3. Click the **General** tab and type a **Description** for the Time-Out event.
4. To specify the calculation of the duration for a Time-Out based on the business calendar, click the **Duration** tab.
5. To enable or disable monitoring of the Time-Out event construct, click the **Monitoring** tab.
6. Click the **Annotation** tab and type additional comments or notes on the Time-Out Event.

The properties of the Time-Out event are set.

Setting the properties of a Swimlane

You can set the properties, such as assigning a role or team to the swimlane.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the swimlane to set its properties. Alternatively, right-click the Swimlane construct and select **Properties** from the menu.
The Properties - Swimlane pane appears.
3. Click the **General** tab and type a **Description** for the swimlane.
4. Click the **Assignee** tab and select the role or team that you want to attach to the swimlane.
5. Click the **Annotation** tab and type additional comments or notes on the swimlane.
For more information on the fields on the Properties - Lane pane, see [Swimlane Properties Interface](#).

You have successfully set the properties of the Swimlane.

Setting the properties of an Annotation

You can set the properties of the Text, Text Annotation, or Transparent Text annotation types to modify its description and font size.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Annotation you want to configure. Alternatively, right-click the Annotation and select **Properties** from the menu.
The Properties - Compensate Event pane appears. See [Annotation configuration interface](#).
3. Make the appropriate modifications:
 - Type text or edit existing text in the **Text** field to modify the annotation.
 - Select a value from the **Font Size** list to set the text size of the annotation.

You have successfully set the properties of the annotation.

Setting the properties of While construct

While is a BPMN group construct that groups activities or sub-processes to execute while the condition is satisfied. At the start of every While loop, the condition is tested and if found false, is not executed. Accordingly, the activities or sub-processes are executed zero or more times. The While construct is used when the loop is to be executed more number of times or none at all; that is to say, it will not be executed if the condition is false from the start.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the While group construct whose properties you want to set. Alternatively, right-click the While group construct and select **Properties**. The Properties - While pane appears. See [While Construct Properties Interface](#).
3. Click the **General** tab, type a **Description**, and select the **Font Size** for the While construct.
4. To apply a condition for execution of the activity, specify the **Execute Condition**.
5. To enable or disable monitoring of the while construct, click the **Monitoring** tab.
6. To store the recovery point either every iteration or at the end of all iterations, select **Store recovery data**.
7. Click the **Annotation** tab and type additional comments or notes on the While group construct.

You have successfully set the properties of the While group construct.

Setting the properties of Until

Until is a BPMN group construct that groups activities or sub-processes to execute until the condition becomes true. Unlike the While construct, the activities or sub-processes execute at least once.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Until construct whose properties you want to set. Alternatively, right-click the Until construct and select **Properties**. The Properties - Until pane appears. See [Until Construct Properties Interface](#).
3. Click the **General** tab and type a **Description** and select the **Font Size** for the While construct.
4. To apply a condition for execution of the activity, specify the **Execute Condition**.
5. Click the **Monitoring** tab to enable or disable monitoring of the Until construct.
6. Select **Store recovery data** to store the recovery point for either every iteration or at the end of all iterations.
7. Click the **Annotation** tab and type additional comments or notes on the Until construct.

You have successfully set the properties of the Until construct.

Setting the properties of For Each

For Each is a BPMN group construct and represents activities or sub-processes to execute for each sub part of a message. The For Each loop uses a counter, known as an iterator to specify the number of times the same sequence of activities should be repeated.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the For Each group that you want to modify. Alternatively, right-click the For Each construct and select **Properties** from the menu.
The Properties For Each pane appears. See [For Each Properties Interface](#).
3. Click the **General** tab and type a **Description**, and select the Font Size for the For Each construct.
4. Type a name for the iterator in the **Iterator Name** text box.
5. Click the **Data** tab and configure the For Each construct for streaming support.
6. Select the **Monitoring** tab to enable or disable monitoring of the For Each construct.
7. Select **Store recovery data** to store the recovery point for every iteration or at the end of all iterations.
8. Click the **Annotation** tab and type additional comments or notes on the For Each group construct.

You have successfully set the properties of the For Each loop.

Setting the properties of a Transaction

A Transaction is a BPMN group construct defined as a set of activities or sub-processes to execute entirely or not at all. Thus, in a Transaction, either all operations are committed or all operations are rolled back. A Transaction therefore, is a single unit of work. This group construct is used when either all the activities within the transaction should be executed once or none at all.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. Double-click the Transaction that you want to configure. Alternatively, right-click the Transaction and select **Properties** from the menu.
The Properties - Transaction Out Event pane appears. See [Transaction Construct Properties Interface](#).
3. Click the **General** tab and type a **Description** for the Transaction.
4. To apply a condition for execution of the Transaction construct, specify the **Execute Condition**.

5. Click the **Monitoring** tab to enable or disable monitoring of the transactional construct.
6. Click the **Annotation** tab and details.

The properties of the Transaction are set.

Setting the properties of an Embedded Sub-process

The embedded sub-process is a BPMN group construct to group activities that belong to a particular context. This helps to hide complexity of activities by collapsing an embedded sub-process. It also helps to link time-outs, exceptions and compensates which are common for all activities within the embedded sub-process.

To set the properties:

1. Select a starting point and click  (Business Process Model) to open an existing business process model.
2. To add an embedded sub-process to the business process, drag the  from the business process toolbar and drop it on to the business process modeler. Alternatively, select a set of shapes in the business process modeler, right-click and select **Group as > Embedded Sub-Process**.
The selected set of shapes is grouped as an embedded sub-process.
3. Double-click the embedded sub-process group construct that you want to configure. Alternatively, you can right-click the embedded sub-process group construct and select Properties.
The Properties pane appears. See [Embedded Sub-process Construct Properties Interface](#).
4. Click the **General** tab, type a **Description**, and select the **Font Size** for the **Embedded Sub-Process**.
5. To apply a condition for execution of the Embedded Sub-Process construct, specify the **Execute Condition**.
6. To enable or disable monitoring of the Embedded Sub-Process, click the **Monitoring** tab.
7. Click the **Annotation** tab and type additional comments or notes on the Embedded Sub-Process construct.

You have successfully set the properties of the Embedded Sub-Process.

Using message maps

A Message Map is a container of all messages flowing in (input) and out (output) of a business process model. It specifies how information is communicated from one activity to another. Message maps help transform data during process flow. It is essential to do message mapping for business processes and case management models to ensure that message flow from one activity to another or next activity prevails and execute these models without any interruption.

The Message Map holds values of variables and attributes accessed by other activities of the process. Whenever the process instance receives a new message, it is added to the Message Map. All the request and response messages are always available in a Message Map.

Consider a human interaction activity with a list box or a table that has 10 records. If you select only two records and complete the activity, the response of completing the activity sends all the 10 records and the selected records' business object is associated with an attribute to distinguish it from the unselected records.

You can [Creating a process specific message](#) in the Message Map. Also, you can [assign input data to a business process model](#) using the Message Map.

Structure of a message map

A Message Map includes two views:

- Activity View
- Consolidated View

In the Activity view, the Message Map comprises two tabs used for message map assignments:

- [Pre-Assignment](#)
- [Post-Assignment](#)

Each of these tabs has three columns:

- [Source](#)
- [Assignments](#)
- [Target](#)

Activities can be identified by the adjoining icons used to differentiate whether it is a Web service, a Decision construct, a While loop, or a Start Event.

Source	<p>The Source column displays the list of all available elements whose values can be used during process execution by assigning them to the target.</p> <p>Process Specific Messages: Process Specific Messages are used to create messages and to drag-drop schema fragments created in the project.</p> <p>Instance Properties: A property that is instantiated in the Message Map to provide instance specific properties. The Instance Specific Properties are used to retrieve information about the process instances at runtime, and can be used in process definitions.</p>
Assignments	<p>Includes the mappings made between the source and target elements. It specifies how a source value can be assigned to a target. A map is created by dragging a source from the left tree and the target from the right tree.</p>

Target	The Target value is an element to which a value is provided at runtime. This column lists only the input messages of the currently selected activity. Process Identifiers: Displays the process related identifiers that are used as searchable attributes. Process Specific Messages.
---------------	--

The [consolidated view](#) lists all the message map assignments created between the various Source and Target elements.

Creating a message map

You can create **Message Map** assignments to specify the input and the output messages for activities in the business process.

1. To create a message map for your business process model you may do any one of the following:
 - [Select a starting point](#) and click  (Business Process Model) to [Create a business process model](#).
The business process model appears in the business process modeling environment.
 - If you have the business process model already opened in the business process modeling environment, then perform Step 2.
2. Click the **Message Map** tab at the bottom of the business process model window. The Message Map appears with three columns: **Source**, **Assignments**, and **Target**.
3. [Create a Process Specific Message](#) if you want any specific messages and elements in the business process.
4. [Assign input data](#) to the respective activity in the process model.
5. [Create an XPath Expression](#) to evaluate the conditions based on data from XML.

The Message Map for the activity is created.

Creating a process specific message

You can create a Process Specific Message in the Message Map, if you want messages and elements specific to the business process model.

1. [Select a starting point](#) and click  (Business Process Model) to open an existing business process model.
2. Click the **Message Map** tab at the bottom of the business process model editor.
The Message Map view for the business process model appears.
3. Under the **Source** column, right-click **Process Specific Messages**, and select **Create Message**.
A new message node is added under Process Specific Messages in editable mode.
4. Type a name of your choice for the message node.

5. To create an element under the message in the **Source** column, perform the following steps:
 - a. [Creating XML schema fragments](#) under the <Project> content tree.
 - b. Drag and drop the required schema fragment on to the created process specific message.
Elements are added to the message based on the elements in the schema fragment.
For more information, see [Tool and Menu Options in Source](#).

The process specific message is created.

Process specific messages

Process specific messages and elements are used in the context of the current business process model. You can create process specific messages from the Source column.

Process specific messages are used to create:

- **XML structure:** You must create intermediate XML structures that need to be used in the course of a process. For example, for the update operation of a tuple in a database, in the Northwind MethodSet, the Update method is the same. The table in which the new row is inserted depends on the table tag in the update SOAP request. So you can create the skeleton WSDL for the Update method and add the table tag with all its data columns under the new tag. XML structures can also be used to create intermediate messages.
- **Temporary elements:** Elements created in Source hold temporary values, for example, counting the total or the loop count in a While or Until loop. Create an assignment with the operation set to Fixed Value with value zero to initialize the element count at the start of the process.
- Process specific messages are available only in WSDL format.
- The process specific messages and elements you create are prefixed with the model namespace.

Creating assignments

Message data is based on the XML Schema of the associated document. As data is passed between the activities, you may need to transform the data between these source and target activities to ensure a smooth flow of data. Assignments or message maps enable data transformation and message flow from one activity to another. Message mapping between the activities is performed for the business processes and Case models.

1. [Select a starting point](#) and click  to open a Case model. Alternatively, select a starting point and click  to open a BPM.
The Case model appears in the modeling environment.
2. If you click , BPM appears in the modeling environment.
3. Select an activity for which you want to assign the input data and click the **Message Map** tab.
The Message Map window opens with two tabs: [Pre Assignments](#) and [Post Assignments](#).

- The Start Event has only the post-activity assignment and the End Event has only the pre-activity assignment. Therefore, the **Pre Assignments** and **Post Assignments** tabs do not appear for the Start Event and End Event activities. This is also applicable to Exception, Timeout, Compensate, and Decision constructs.
 - You cannot map the output messages of a Human Task in a Case model.
 - In a Case model, you can map the previous activity details as part of the [Pre-Assignments of an Activity](#). This option is available for the activities that are connected through *Automatic*, *Manual*, or *Intermediate* follow-up modes and if both the previous and current activities are in the same state. However, even though previous activity details are displayed in the Pre-Assignments of the first activity in the state, you cannot pass values to the activities during runtime. Also, you cannot map previous activity details for a Free follow-up activity.
4. Do one of the following to assign the input data:
 - Drag a source element from the **Source** column on the left and drop it on the left text box of the **Assignments** column. Drag a target element from the **Target** column on the right and drop it on the right text box of the **Assignments** column.
 - Select the source element, hold down the Control key on the keyboard, and select the target element.
A new map appears in the **Assignments** column displaying the new assignment. You can drag multiple source elements onto the **Assignments** column for a single target element in an assignment.
 5. Select the [operation type](#) from the **Use** list and then select the required operation from the respective operation sub menu in the **Assignments** column.
For example, to select the operation Replace Content with Expression, you must click the **Use** list, select **Replace Content With** first and then **Expression** from the sub navigation menu. The operation is set for the assignment.
 6. Provide a value in the input area provided under the dropped elements for the Fixed Value or **Expression** assignment in the **Assignments** column.
For example, to retrieve the employee records of the first employee, you must specify a fixed value of **1** as an input for Employee ID. This requires a fixed value assignment for the Employee ID element.
You can achieve this by doing any of the following:
 - Click the **Use** list, select **Replace Content With** first and then **Fixed Value** from the sub navigation menu.
 - Drag the **Fixed Value** icon on the bottom left corner to the map area in the **Assignments** column.
- Important:** The assignments are created for the required elements.
- When you make the assignments in a BPM, for string constants, provide values in double quotes and for others such as conditions, functions, and expressions, provide the values in single quotes. For example, consider the following assignment: add the expression **test**.

- The Message data (XML Schema) of the associated activities are generated based on the document name. If there is a change in the associated document name, then the assignments defined will be invalid. Therefore, you must update the assignments according to the modified message data.
- While creating assignments in the BPM, the attachment definitions defined in the subprocess are also available through the message-map. While performing mappings of such attachment definitions, ensure the following:
 - The operation must be set as **Replace Content With** and the parameter must be **Children with Target NS**.
 - The MIME types for the attachment definitions that are being considered for the message-mappings are compatible. This ensures that during the execution of a process instance, when the attachment details are shared, the process instance aligns with the MIME type that is already defined for the subprocess.
- The changes made to the message-map through assignments are done in-memory and are persisted at the subsequent recovery point. In case of a crash before the persistence of the changes in the message-map, during crash recovery, when the process instance starts execution from the last saved recovery point, there may be repetition of assignments. This must not cause any issue with assignments using regular XPath expressions or fixed value as a source. However, if the source XPath triggers a custom Java API with certain actions in the Java code, the actions may be executed again. Such scenarios must be handled during the application development.
- If a subcase is associated with another Case model, assignments can be provided to both the Case data and [Case variables](#) of the subcase. The tooltip of the Case variable represents its **type**. The option to pass values to the Case variable is available when a subcase is used in both the Case and BPM models. The Case variables of the Case model can be passed to the Case variables of its subcase.
 - From BOP 4.3, you can also update the Case Variables through the message map. It is no longer required to trigger the UpdateCaseVariables API to have the related changes or updates considered explicitly as a part of the Case model execution.

Source

The Source column, a part of the Message Map, lists the input and output messages of all the previous activities in a tree structure as defined in the WSDL of the activity.

For example, if the third activity in a business process model is selected, the Source column displays the input and output messages of the previous two activities.

Elements from the Source column can be assigned to elements in the Target column.

Use the Source column to do the following:

- Assign input to an element by selecting a particular element from the Source column and mapping it to an element in the Target column. You can do this by either using the Control key or by dragging and dropping the respective elements to the Assignments column.

- Copy the XPath of an element.
 - Right-click the element and click **Show XPath**.
 - Copy the text from the **Show XPath** dialog box and use it for checking the conditions in the Decision Connectors, While loop, Until loop, and in the **Target** and **Source** columns in the **Message Map**.
- Copy **Process Specific Messages** from one business process to another business process.
 - Select the process specific message you want to copy in the source business process.
 - Right-click and select **Show XML** and copy the text from the **Show XML** dialog box.
 - Open the target business process and select **Process Specific Messages** in the **Message Map**.
 - Right-click and select **Paste XML as Message**.
- Display the system generated instance-specific information in the **Message Map** of the process instance. For example, start time, process name, version, identifier, and so on. You cannot assign data to these instance properties, but only read data from these properties and map it in the **Assignments** column to the elements under the messages in the **Target** column.
- Search can be performed for a source element available in the **Assignments** column. Right-click an element in **Source**, which has been mapped to a target element and select **Find in Source Assigns**.

See [Toolbar and Menu Options in Source](#) for more information on the toolbar and context menu options.

Toolbar and menu options in source

1. Click the following icon in the toolbar to access the toolbar options:

Tool Icon	Description	Available for...
	Filters by Process Specific Messages, Instance Properties, Activity, and All	

2. To access the menu options, right-click the element type in the Source column. The options on the menu depend on the element type selected - message, element or attribute. The following table lists the menu options of the Source column.

Menu Option	Description	Available for...
Expand All	Expands the node and displays all the child elements under it	Message and Element
Rename	Renames the node or element	Message

Menu Option	Description	Available for...
Delete	Deletes the element	Message and Global Schema Elements
Create Message	Creates a message under Process Specific Messages in the Source column	Root node or root level
Create Element	Creates an element under Process Specific Messages in the Source column	Root node or root level
Paste XML as Message	Creates a message in XML format under Process Specific Messages in the Source column	Root node or root level and Message
Paste XML as Element	Creates an element in XML format under Process Specific Messages in the Source column	Root node or root level, Message, and Element
Create Attribute	Creates an attribute under Process Specific Messages in the Source column	Element
Show XSD	Displays the schema of the selected node	Element
Show XML	Displays the XML structure of the selected node	Message and Element
Show XPath	Displays the XPath of the selected node	Message and Element
Find in Source Assigns	Searches for a source element that is mapped to a target element in the Assignments column	Element

Assignments

The Assignments column, a part of the Message Map, lists the various types of mappings between Source and Target elements for the pre-activity and post-activity assignments. For example, if the third activity in a business process model is selected, the Source column would display the input and output messages of the previous two activities.

When a map (assignment) is clicked, the view is in edit mode. In the edit mode you see the operation or in case of fixed value when you click on a mapping, you see the actual value that was assigned to the fixed value.

Every map shows the elements that have been assigned input data. In certain cases there could be multiple sources and for a single target defined in a given map.

You can rearrange the order of assignments in a map by moving up or down using the toolbar icons. At the same time, you can delete an assignment.

To view an input or output of the activity:

1. In the AppWorks Platform business process modeling environment and select an activity
2. At the bottom of the window, click the **Message Map** tab and select from the following:
 - **Pre-assignments**: To view the input to the Activity and assign input data in **Assignments** that should be executed before the activity, at runtime.
 - **Post assignments**: To view the output from the Activity and assign input data in **Assignments** that should be executed after the activity, at runtime.

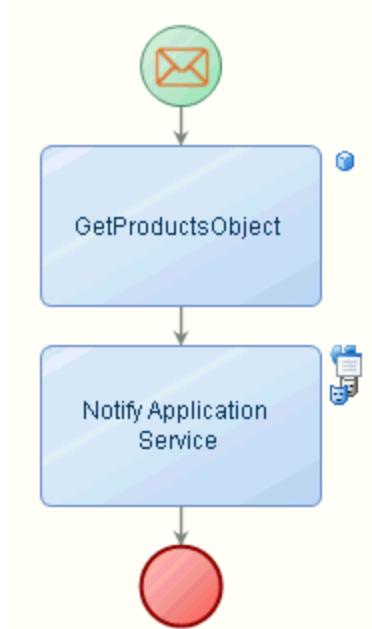
Pre-assignments

A Message Map is used to pass data from one activity to another in the business process. Inputs define the data that the element requires to start the activity or process.

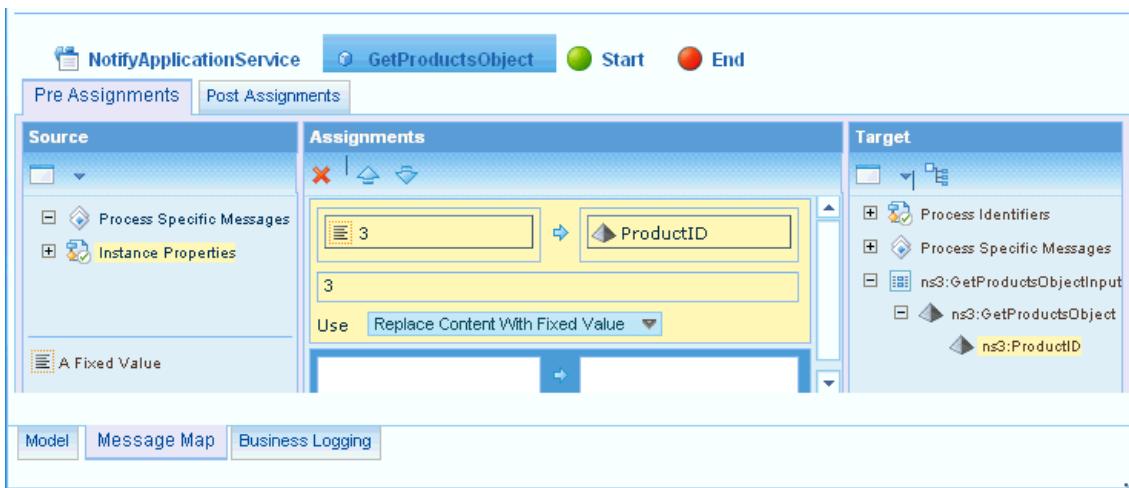
An input triggers an element to start, taking over from a preceding element that has reached a stage of completion. The inputs also define the data that the element needs before it can start. During execution of a business process, the output of an activity in the business process becomes an input for the ensuing activity in the process.

Example of a pre-assignment in the Message Map

A business process is modeled so that it returns the Product ID and the UnitsInStock. The product ID from this output message is used as an input (input message) for the next activity that retrieves information on the number of units in stock.

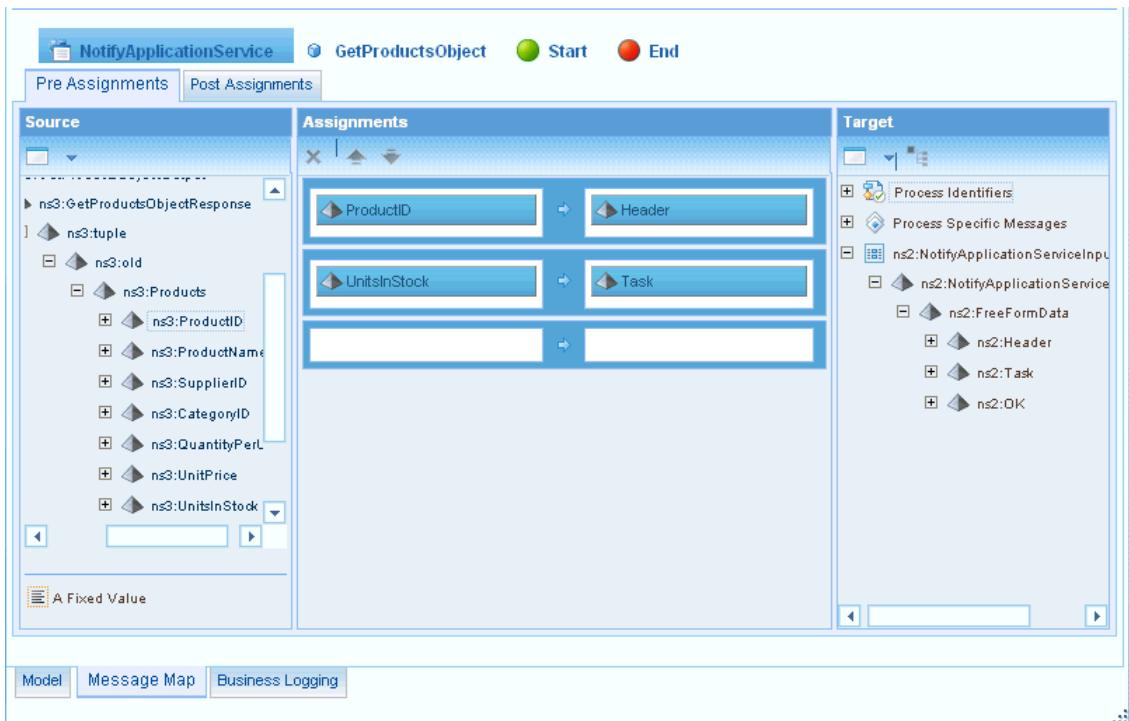


The product ID is the input for the process to start (input for the Start event). The product details are retrieved in the GetProduct activity, based on the product ID that was provided as an input to the Start event.



- Drag the message (GetProductInput) from the **Workspace Documents > Project** pane on to the Start event.

A notification is sent to the Purchase Manager about the units in stock for the particular product. The output of the previous activity (GetProduct) forms an input for the Notify Application Service activity.



In the **Pre Assignments** (Notify Application Service activity), the output of the GetProduct activity is mapped to the input for the notify activity. The ProductID is mapped to the Header and the UnitsInStock is mapped to the XForm in the notify activity.

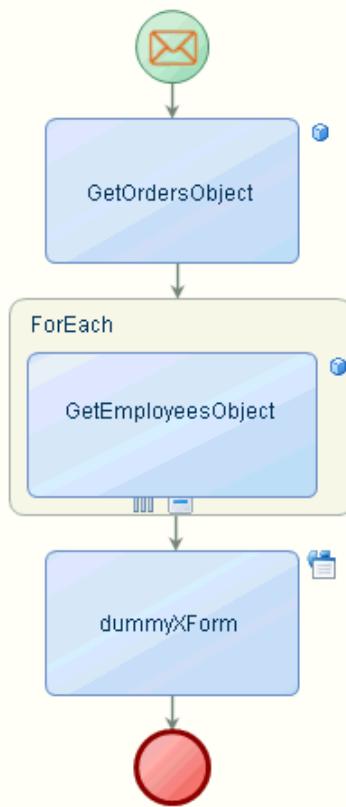
Post assignments

A Message Map is used to pass data from one activity to another in the business process. Outputs define the data that the element produces after it has run.

An output is an exit point through which an element can communicate to the elements in sequence that they can now start. The output from one activity provides input for the next activity in the business process.

Example of a Post Assignment in the Message Map

A business process is modeled so that it returns the employee's name and experience who processes the order. The EmployeeID from the GetOrdersObject method is used as the input to fetch details such as employee name and date of joining.

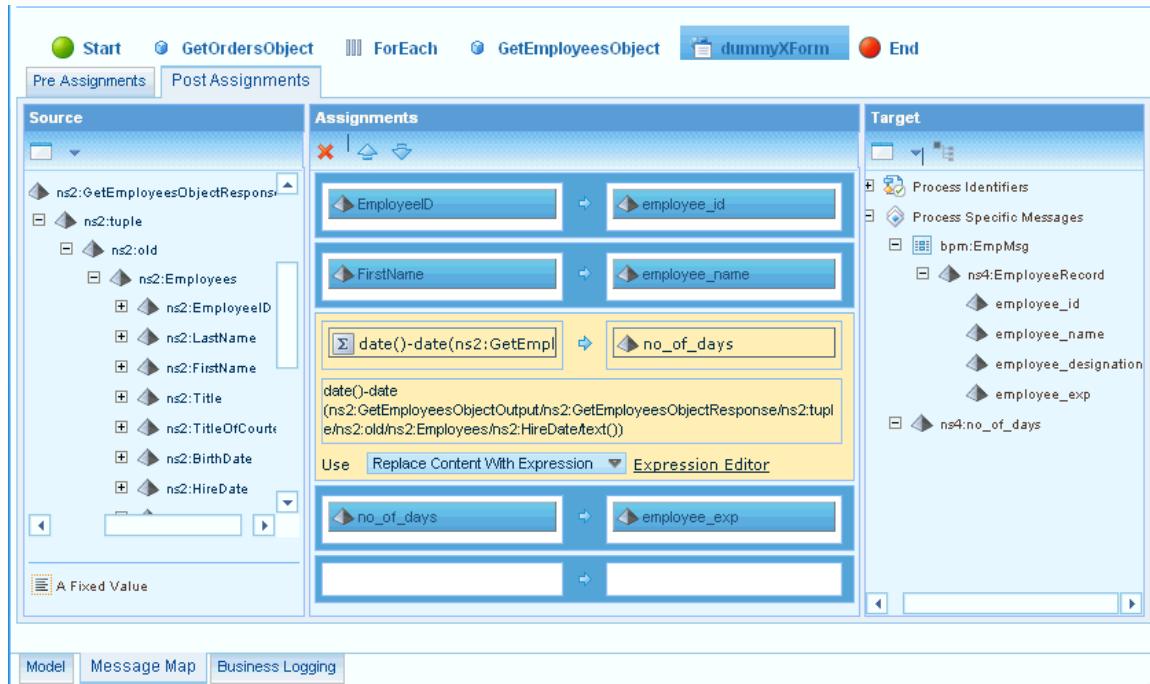


The order ID range is the input for the process to start (input for the Start event). The order details are retrieved (output) in the GetOrdersObject activity.

To create the following process specific elements in the Message Map:

EmployeeRecord	This element stores the employee details such as employee_id, employee_name, employee_designation and employee_exp(employee experience).
no_of_days	This element stores the date of joining of the employee.
EmpMsg	This message stores the details of all employees including first name and experience.

In the **Post Assignments** tab of dummyXForm, the EmployeeID element in the GetEmployeesObjectOutput is mapped to the employee_id, FirstName is mapped to employee_name as follows.



1. Right-click the employee_exp element and select **Create Assignment** to calculate the experience as shown.
2. Create an assignment for EmpMsg as shown.

Target

The Target column displays the input message structure as defined in the WSDL of the activity. You can use this column to create assignments for the target elements by mapping them with the source elements from the Source column. Once you assign input data to the elements, they are displayed in the Assignments column. You can create multiple assignments for the elements by selecting the Create Assignments for Target Leaf Nodes option on the toolbar.

For a selected activity, if the Target column on the Message Map does not contain any element, the Target column shows that there is no input defined in the WSDL for the activity.

You can search for a target element in this column that has been mapped to a source element under the Assignments column by using the Find in Target Assigns option in the toolbar.

See [Toolbar and Menu Options in Target](#) for more information on the toolbar and context menu options.

Toolbar and Menu Options in Target

Click any of the following icons in the toolbar to access the toolbar options:

Tool Icon Option	Description	Available for...
 (Filter)	Filters by Business Identifiers, Process Specific Messages, Activity, and All	
 (Create Assignments for Target Leaf Nodes)	Creates Fixed Value mappings in the Assignments column for all the leaf nodes available under a message - element or attribute	Message, Element, or Attribute - whichever is the leaf node

Right-click the element type in the **Target** column to access the menu options. The options on the menu depend on the element type selected - message, element or attribute. The following table lists the menu options of the **Target** column.

Menu Option	Description	Available for...
Expand All	Expands the node and displays all the child elements under it	Message and element
Show WSDL	Displays the WSDL of the Web service	Message
Show XML	Displays the XML structure of the message node	Message and element
Show XPath	Displays the XPath of the selected node	Message and element
Set Default Values	Uses the default values for the Web service present in the WSDL. If this option is selected the icon for the message node is represented by a  . If there are any assignments created for an element with default values, then the assignments take precedence over the default values.	Message
Show XSD	Displays the XSD definition for the selected element	Element and attribute
Create Assignment	Creates an assignment row in the value assignment table	Element and attribute
Delete Assignment	Deletes the existing assignment	Attribute
Find in Target Assigns	Searches for a target element that is mapped to a source element in the Assignments column	

Deleting a data assignment

This topic describes the procedure to delete a data assignment from the Message Map.

To delete a data assignment:

1. From the **Solution Explorer**, click **<Project> > <your business process model>**.
The business process model appears in the business process modeling environment.
2. Click the **Message Map** tab and select the construct or activity name from the Activity Map Definition.
The existing map of data assignments for the selected construct or activity appears.
3. Do one of the following to delete a data assignment:
 - From the Source column, right-click the message or element to be deleted and select  from the menu.
 - From the Assignment column, select the required assignment and select .

The assignment is deleted from the Message Map.

Creating an XPath expression

XPath 1.0 is a language that you can use to select one or more nodes from XML. It is also used for creating static expressions and evaluating conditions based on data from XML.

You can use the XPath Editor to create or modify XPath expressions. See <http://www.w3.org/TR/xpath> for the XPath 1.0 specifications. For information on the custom functions supported by XPath, see the [Rule Engine Function Library](#).

To create an XPath expression:

1. In the assignment row of the **Message Map**, click the **Expression Editor** link. The XPath Editor dialog box opens.
2. Use the following features to create an XPath expression:

Drag & Drop	You can drag a node from the Tree tab to the XPath text area. The XPath of the node will appear in the XPath text area. Drag a function and/or operator from the Functions and Operators menu to insert it in the XPath.
Intellisense	You can drag a node from the Tree tab to the XPath text area. The XPath of the node will appear in the XPath text area. Drag a function and/or operator from the Functions and Operators menu to insert it in the XPath.
Context Menu	Right-click a node in the Tree tab to access the context menu. The following options are available in the context menu: <ul style="list-style-type: none">■ Expand All - Expands all connected sub-nodes of the selected node.■ Use => Inserts the selected node in the XPath. In the Functions

	and Operators menu, this option will insert the function or operation in the XPath. By default, the selected node will be inserted at the place where the XPath was last edited. This option works the same way as the Use => button under the respective menus.
Auto Validate	Inserts the selected node in the XPath. In the Functions and Operators menu, this option will insert the function or operation in the XPath. By default, the selected node will be inserted at the place where the XPath was last edited. This option works the same way as the Use => button under the respective menus.
Color Code	The text of the expression is color coded. This helps in checking if strings are closed correctly, function names are recognized, and presents an overview of elements, attributes, functions, strings, and numbers that are used in the expression.

3. Click **Validate**.

A message notifies the validity of the expression.

4. Click **Test**.

The expression is validated and evaluated. If the expression and XML is valid, the result is displayed in the **Test Result** area of the *XPath Editor*.

- All XPath expressions are supported in the **Message Map** Expression. You can also use static Java in rules.
- Complex expressions that can be dynamically added through the XPath editor, such as if-then-else conditions, substring (string, start index, length) and so on, are not supported in the **Case Modeling Message Map**. However, you can directly assign values in the message map.

Using the consolidated view

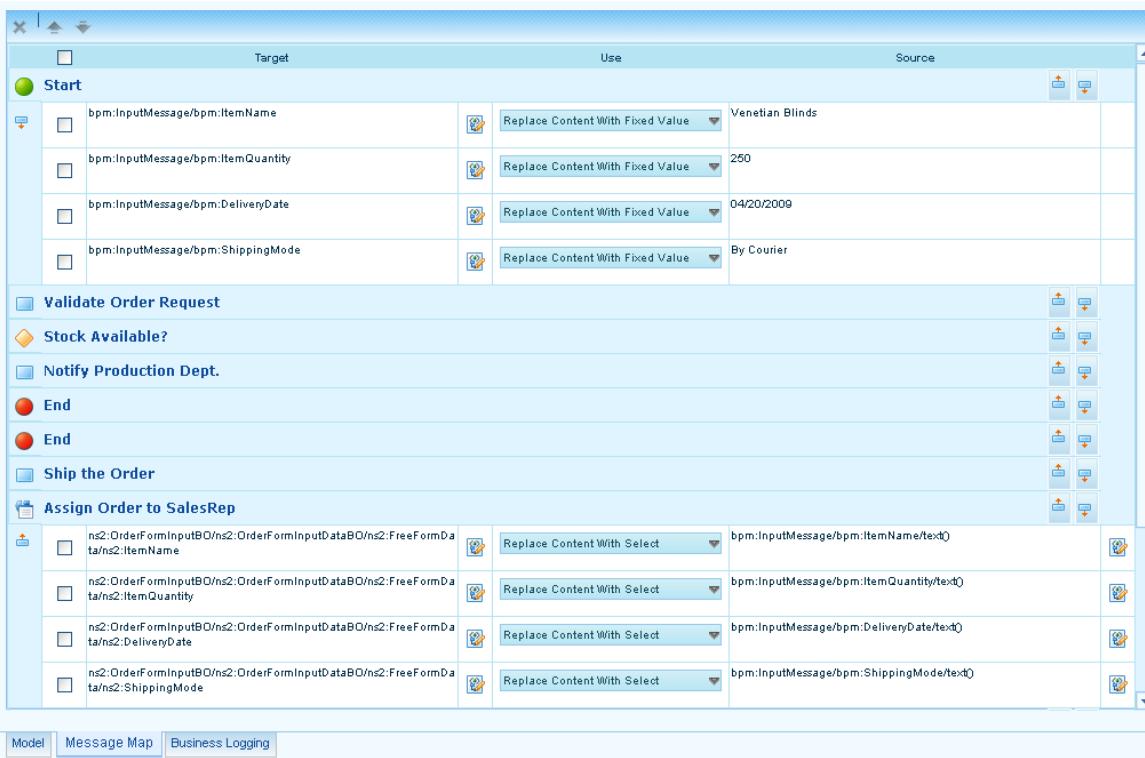
Before you begin, for a business process model, the Message Map tab needs to be in focus to view the Consolidated view.

The Consolidated view presents a complete view of all message map assignments made for the various activities in a process model.

To open an existing business process model:

1. From Workspace Documents (Explorer) view, expand <Project> and click  (Business Process Model).
2. Click **Message Map** tab.
The Message Map appears.
3. Click **Switch to Consolidated View** icon at the top left corner of the Message Map window.
The Consolidated View of the Message Map appears displaying all the activities and all assignments.

The  icon denotes a pre-activity assignment for the activity and the  icon denotes a post-activity assignment.



4. To change the sequence of the activity in the consolidated view, select the activity and click Move Up () or Move Down () on the toolbar.
5. To create a new pre-activity assignment for a given activity, click  in the consolidated view.
6. To create a new post-activity assignment for a given activity, click  in the consolidated view.
7. To reassign or change any of the existing data assignments, select the required operation type from the list under the **Use** column and change the value in the **Source** column.

8. To search for a given text on any of the assignments in the **Source** or **Target** column, follow these steps:
 - a. Click Search  on the toolbar.
 - b. Type a value in the **Find what:** box and select **Find**.
The search results are highlighted.
9. To search and replace the given text with a new value, follow these steps:
 - a. Click the Search  on the toolbar.
 - b. Type a value in the **Find what:** box that needs to be replaced.
 - c. Type a value in the **Replace with:** box and select **Replace**. To replace all the instances with new value, click **Replace All**.
10. You can customize your search to look for a given value only in Source or Target assignments by selecting either **Source Assigns** or **Target Assigns**. Likewise, you can search only from top by selecting **Up** or from down using the **Down** option.

Configuring message filters

When you want to monitor the process and its activities using AppWorks Platform BAM, based on only relevant or specific message map data, you need to configure message filters.

Before you begin:

- [Create](#) and [Design](#) a business process model.

You can configure message filters only for process specific messages and activities that have messages attached such as properties of the process instances, task, web service, independent sub-process, contract first development process, decision table and receive message.

BAM supports monitoring of only message elements and not message attributes.

To configure message filters:

1. From Workspace Documents (Explorer) view, expand <Project> and click  (Business Process Model) to open an existing business process model.
2. Click the **Message Filter** tab on the business process model editor.
The Message Filter tab view appears displaying the **Messages** and **Message Filters** panes. Activities such as Task, Web service, Independent Sub-process, Contract First Development process, Decision Table and Receive Message if any, appear in the pane above Message and Message Filters panes.
See [Message Filter Interface](#) for details on the fields on the Message Filter tab view.
3. To configure message filters, select an activity in the top pane and do any one of the following:

- From the **Messages** pane, select a message element and drag and drop it on to the **Message Filters** pane.
- In the Message Filters pane, click and in the row that appears.
- Click under the XPath column and select the required message element from the XPath Editor.
- Provide a **Name** in the **Message Filters** pane.
- Right-click a message element and select **Add All Children** to add all the children of this node to the message filter.

You can use this option only for a message element that has children nodes.

All the configured message filters appear in the Message Filters pane by default.

If you want to view the message filters of any specific activity, right-click on the required activity in the top pane and select **Message Filters of this activity**.

4. Click .

Message filters are configured for the selected process and its activities.

Message filter interface

The Message Filter tab displays the following views.

Horizontal pane	<p>Appears above the Messages and Message Filters panes, activities such as task, web service, independent sub-process, contract first development process, decision table and receive message, if they exist for the current business process model.</p> <p>To view message filters for a specific activity:</p> <ul style="list-style-type: none">■ Right-click required activity and select Message Filters of this activity.
Messages pane	<p>By default, the Messages pane displays only process specific messages, process instance properties, and contract first development input/output messages, if any.</p> <p>To view the message for activities such as task, Web service, Independent Sub-process, Contract First Development process, Decision Table, and Receive Message, if they exist for the current business process model:</p> <ul style="list-style-type: none">■ Select the required activity in the top pane. Messages for that activity appear in the Messages pane.
Message Filters pane	<p>The Message Filters pane displays all existing message filters for the current business process model.</p> <ul style="list-style-type: none">■ Click (Show All Filters) to view all message filters.

	<p>The following fields are displayed on the Message Filters pane:</p> <ul style="list-style-type: none"> ■ Name - A name of the message filter. You may provide a custom name for the message filter. ■ Type - The data type of the message filter. ■ XPath - XPath of the message filter. <ul style="list-style-type: none"> • Click  to select a message element from the XPath Editor to configure it as a message filter.
--	---

Setting the process execution mode

The [process execution mode](#) for a business process can be set as a long-lived process, short-lived process, or as a page flow. The process execution mode for a business process model must be selected as a long-lived process when it includes human interactions, delays, or intermediate messages and takes a long time to execute. A process execution mode for a business process must be selected as a short-lived process only when it does not require human intervention and when its average life span is short. Select page flow as the execution mode when you require the user to fill multiple forms, either HTML pages or User Interface that appear in a wizard-like mode instead of accessing through the AppWorks Platform Inbox.

To set the process execution mode:

1. [Creating a document](#) and click  (Business Process Model) to open an existing business process model. Alternatively, if the business process model is already open, then perform Step 2.
 2. Double-click or right-click in the business process modeling environment and select **Properties**.
- The Properties - Business Process Model pane appears.
3. In the **General** tab view, select one of the following from the **Execution Mode** list.

Long-Lived	This process is asynchronous in nature and takes a long time to execute as the process involves human tasks, delays, and intermediate messages. By default, a process is a long-lived process unless it is specified otherwise.
Short-Lived	This process is synchronous in nature and has a short life span.
Page Flow	<p>This process executes in a page flow mode.</p> <ol style="list-style-type: none"> 1. To receive a task from the business process in the client UI, include the page flow library on the application select in the client application. <p>For example:</p> <pre>application.addLibrary ('/cordys/cas/vcm/library/pageflow.htm',</pre>

```
pageFlowEnabling);
```

Do not use the library if the user interface is part of the workflow. The library is best suited to invoke a page flow process and continue with the manual tasks present in the page flow.

The `pageFlowEnabling` is the ID of some element on the web page.

2. Pass the SOAP request (`ExecuteProcess` request) document to invoke the method in the library.

For example: `pageFlowEnabling.invoke
(requestStartPageFlow.XMLDocument);`

3. Detach the page flow library on the application close of the page.

For example: `application.removeLibrary
('/cordys/cas/vcm/library/pageflow.htm',
pageFlowEnabling);`

-
4. Click **Save**.
 5. [Activity properties interface](#) and complete the **Message Map** for each activity in the business process.
 6. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**.
The business process model is validated. If any errors appear, resolve them.
 7. [Publishing a document to an organization](#).

The process execution mode for a business process is set.

Attaching web services to a business process model

Before you begin:

- [Generate a Web Service](#). Required Web service must exist in the project content tree to be attached to an activity.

To attach Web Services to a Business Process Model:

You can invoke Web service operations from more than one activity.

1. [Select a starting point](#) and click  to open a business process model.
The business process model appears.
2. Follow the appropriate procedure to attach the required Web service operation to the business process model.
 - Drag the **<Web service operation>** from the **<Project>** tree list in the **Workspace** tab onto the business process model.

- Drag the <**Web service operation**> from the **Web Service Operation** group in the **Insert** tab onto the business process model.
3. Right-click the activity, select **Insert** > **Web service operation** and select the appropriate Web service operation from the **Select a User Interface** dialog box.
 4. The Web service operation is added to the activity in the business process model, and the activity name changes to the name of the Web service operation.
 5. You can add a **Functional Role**, **Organization Unit**, **Work List**, or a **Business Calendar** to the activity. You can also modify the user interface added to the activity.
 6. If you drag and drop a <**Web Service Operation**> onto the activity, you will be asked for confirmation about description change of that activity with the description of the Web Service that is being associated. If you choose to associate the same through the context menu **Insert** > <**Web Service Operation**>, then the confirmation dialog is not shown. The selection overwrites the activity description with the selected <**Web Service Operation**> description.
Similar functionality is also applicable when a selection is made for the <**case model activity**>, <**business process activity**>, <**Decision Table**>, or a <**User Interface**>.
 7. To enable the business process model engine to dynamically select a service group at runtime, specify the XPath of the service group in the **Read Service Group or Service Container from Message** field of the **Receiver** tab in the properties pane.
 8. Select the Web service activity in the business process model and click the **Message Map** tab. The **Message Map** for the Web service activity appears.
 9. [Creating assignments](#) and also for other activities in the business process model as necessary.
 10. [Configure the remaining constructs](#) used in the process model.
 11. Right-click the business process modeling environment and select **Business Process Execution** > **Validate and generate BPML**. Alternatively, go to **Workspace Documents** > <Solution> > <Project> and right-click the <business process model> and select **Business Process Execution** > **Validate and generate BPML**. The Validate progress window appears and when the validation is over, if there are any warnings or errors, then the warnings appear. Resolve the errors and revalidate the process model.
 12. Right-click the business process modeling environment and select **Business Process Execution** > **Publish to Organization**. Alternatively, go to **Workspace Documents** > <Solution> > <Project> and right-click the <business process model> and select **Business Process Execution** > **Publish to Organization**. The business process model is published to organization.
 13. Right-click the process modeling environment and select **Business Process Execution** > **Run**. Alternatively, go to **Workspace Documents** > <Solution> > <Project> and right-click the <business process model> and select **Business Process Execution** > **Run**. The business process model is instantiated.

The required Web service is attached to the business process model.

For information on invoking different kinds of Web services when executing a business process model, see [Using a web service in a business process model](#) and [Using an external web service in a business process model](#).

Attaching a user interface to a business process model

Before you begin, the required tasks must exist in the project content tree. If required create the necessary tasks or interfaces.

To attach a user interface to a business process model:

1. Select a starting point and select  (Business Process Model) to create a business process model. Alternatively, open the existing business process model to which the user interface needs to be attached.
2. Follow the appropriate procedure to add the required user interface or external user interface to an activity in the business process model.
 - Drag the **<User Interface>** or **<External User Interface>** from the **<Project>** tree list in the **Workspace** tab onto the business process model.
 - Drag the **<User Interface>** or **<External User Interface>** from the **User Interface** group in the **Insert** tab onto the business process model.
 - Right-click the activity, select **Insert > User Interface** and select the appropriate user interface from the Select a User Interface dialog box.
The user interface is added to the activity in the business process model.
 - If required, you can further add a **Functional Role, Organization Unit, Work List**, or a **Business Calendar** to the activity. You can also modify the user interface added to the activity.

If you drag and drop a **<User Interface>** onto the activity, you will be asked for confirmation about description change of that activity with the description of the User Interface that is being associated.

But if you choose to associate the same through the context menu **Insert > <User Interface>**, then the confirmation dialog is not shown. The selection overwrites the activity description with the selected **<User Interface>** description.
Similar functionality is also applicable when a selection is made for the **<Case Model Activity>**, **<Business Process Activity>**, **<Decision Table>**, or for a **<Web Service Operation>**.

3. Follow the appropriate procedure to add the required role to an activity in the business process model. This is an optional step.
 - Drag the **<Role>** from the **<Project>** tree list in the **Workspace** tab onto the business process model.
 - Drag the **<Role>** from the **Role** group in the **Insert** tab onto the business process model.

- Right-click the activity, select **Insert > Functional Role** and select the appropriate role from the **Select a Functional Role** dialog box that appears.
The role is added to the activity in the business process model.
4. The procedure for adding an *Organization Unit*, *Work List*, or a *Business Calendar* to an activity is the same as Step 3. Follow Step 3 to add these to the activity in the business process model.
 5. To set the properties of the user interface added to the activity, right-click the **<User Interface>** or **External User Interface** in the business process model and select **Properties**. The **<Activity> - Task Properties** pane appears.
 6. Set the properties of the **<User Interface>** or **External User Interface**.
See [Activity properties interface](#) for detailed information about the options available in the **<Activity> - Task Properties** pane.

A User Interface or External User Interface is attached to the business process model.

Invoking a decision table from a business process model

Before you begin:

- Build a decision table
- Configure the Rule Engine

To invoke a decision table from a business process model:

1. In Workspace Documents (Explorer), expand **<Project>** and click  **<Business Process Model>** to open an existing business process model.
2. Select the **<Decision Table>** from the project content tree view, drag and drop it on the required activity construct in the **<business process>**.

The decision table is successfully attached to the business process.

See [Properties of a decision table](#).

Enabling the rule engine for decision table

You must first configure the usage of a decision table in the business process.

To configure the usage of a decision table:

1. Click **Menu** on the toolbar.
The Menu appears displaying a tree view of all the tasks on the My Applications palette.
2. Click **System Resource Manager**.
The System Resource Manager opens with a list of existing Services.
3. Double click the **Business Process Management** Service.
The Properties - Business Process Management and Memory Status views appear.

See [Service container configuration interface](#) to set the properties of the Business Process Management Service.

4. In the **Properties - Business Process Management** pane, click the **Business Process Management** tab.

The Business Process Management tab view appears displaying Admin Database tab view details as the default view.

See [Business process management service properties interface](#) to set the properties on the **Business Process Management** tab.

5. Click the **Process Engine** tab.

The [Process Engine](#) tab view appears.

6. Select the **Enable Rule Engine** check box to enable it.

The [Admin database, Data Transformation, Scheduler and Rule Repository tabs](#) appears.

7. Select the required database configuration from the **Select Database Configurations** list in the Rule Repository tab view.

The selected database configuration details appear.

8. Click .

9. In the System Resource Manager view, right-click the **Business Process Management** Service and select **Restart**.

The **Business Process Management** Service restarts.

The Rule Engine is enabled for the decision table.

After you complete this task:

- You can use the [decision table in a business process](#).

Enabling reliable messaging in a business process

Before you begin, you must have the following on the system where AppWorks Platform is installed:

- Ensure that a Queue set up is installed.
- Configure Business Process Management Service with client connection point group having Queue connection to create socket and queue connection points.
- Ensure that Socket and Queue connection points are configured for WS-Apps Service Container.
- Ensure that the Business Process Management and WS-Apps Service Containers point to OLEDB Database server.

Reliable messaging ensures that all SOAP requests and responses are processed such that no transaction is lost.

To enable reliable messaging:

1. On the Welcome page > My Applications, click  (System Resource Manager).
The System Resource Manager window opens with a list of all available Service

Containers in the Service Containers App Palette.

2. Right click the **Business Process Management** service container and select **Properties**.
The **Business process management service properties interface** appears.
3. Click **Business Process Management > Process Engine** tabs.
4. Click  (Connection Point) associated with the Client Connect Point field.
The Connection Point Groups dialog box opens with the socket and queue connections points.
5. Select a connection point and click **OK**.
6. Click .
7. In the Service Containers list, right click the **Business Process Management** service container and select **Restart**.
The Business Process Management service container restarts.
8. **Select a starting point** and click  (Business Process Model) to open an existing business process model.
9. Drag a Webservice operation from the project content tree and drop it on the business process modeler.
10. Double click or right-click <Webservice activity> and select **Properties**.
The Properties - Activity pane appears.
11. Click the **Webservice Options** tab in the properties view.
The Webservice options view appears.
12. Enter required time in seconds in **Timeout in seconds**.
13. Select **Use reliable messaging** and **Perform other tasks simultaneously** and other options as required.

The reliable messaging feature is enabled for the Web service activity of a business process.

Documenting business process models

Before you begin:

- You must have the role of an Analyst.

During design time, often there arises a need for discussing, reviewing, and sharing information about business process models. This is done to

- Arrive at decisions on how the business process model should be designed
- What portions of the business process model need to be modified
- Obtain a single view of the entire business process model containing the specified model and activity level properties
- Any other requirement for which you may need to document your business process model(s)

In the context(s), AppWorks Platform extends the **Export** functionality to document your business process model(s), which helps you use and reuse documented process models for immediate and future references.

Currently, documenting support is available only for business process models.

To document business process models:

1. In the Workspace Documents, right-click any one of the following:
 - <business process model>
 - <folder>
 - <project> A context menu appears.
2. Select **Export**. The *Export Wizard* appears.
3. Select **Business Process Model** from the **Model Type** list.
4. Select **ReportExportPlugin** from the **Export Plugin** list.
5. Click **Next**.
The next screen of the Export wizard opens.
6. Click  associated with the **Location to get the models to export from** field, to select a process model for document generation.
7. Type a name for the export file in the **Export File Name** field.
The export file is a .zip format, by default.
8. Click **Export**.
9. Click the **Download** link to download your document.

The selected business process models are documented.

After you complete this task, configure the [Configuring the document generation properties](#).

Configuring the document generation properties

Before you begin, you must have the administrator rights on the computer on which AppWorks Platform is installed.

To generate the document to serve your business needs, set the following platform properties in the Management console:

1. Update the following properties:

bpm.designer.report.page.title	The value for this property is used as the title in the first page. The default value is set as Business Process Document .
bpm.designer.report.author	Represents the owner of the document that is being generated. If no explicit name is provided, the user ID of the user who

bpm.designer.report.page.header	generates the document will be considered while generating the document.
bpm.designer.report.type	The value is displayed as the header on each page of the generated document except for the first page.
bpm.designer.report.include.messageMap	Specify the output format of the document as .doc or .pdf . By default, the report generation format is .pdf .

2. To generate the document with multilingual characters, the following settings must be completed:
 - a. Generate the metrics file for each font to be supported by the **.pdf** or **.doc** with the following command:

Command

```
java -cp %CORDYS_HOME%\ext\fop.jar;%CORDYS_HOME%\ext\avalon-framework-4.2.0.jar;%CORDYS_HOME%\ext\commons-logging-1.1.1.jar;%CORDYS_HOME%\ext\commons-io-1.3.1.jar;%CORDYS_HOME%\ext\xmlgraphics-commons-1.5.jar org.apache.fop.fonts.apps.TTFReader -ttcname "<Font Name>" <Path of TTC/TTF File> <Path of Metrics File> Path of TTC/TTF File: Path where the supported TTF/TTC file is present in the server Path of Metrics File: Path where the generated Metrics file needs to be stored
Font Name : Name of the Font
```

Sample Code

```
java -cp %CORDYS_HOME%\ext\fop.jar;%CORDYS_HOME%\ext\avalon-framework-4.2.0.jar;%CORDYS_HOME%\ext\commons-logging-1.1.1.jar;%CORDYS_HOME%\ext\commons-io-1.3.1.jar;%CORDYS_HOME%\ext\xmlgraphics-commons-1.5.jar org.apache.fop.fonts.apps.TTFReader -ttcname "MS Gothic" C:\Windows\Fonts\msgothic.ttc D:\FontMetricFiles\msgothic.xml
```

- b. Provide the corresponding metrics file and font file details in the configuration file. See <http://xmlgraphics.apache.org/fop/1.1/fonts.html> for more details. Place this configuration file at a location in the <AppWorks Platform_installdir>. A sample configuration file is provided below:

```
<?xml version="1.0"?>
<fop version="1.0">
  <strict-validation>true</strict-validation>
  <renderers>
```

```

<renderer mime="application/pdf">
  <filterList>
    <value>flate</value>
  </filterList>
  <fonts>
    <font kerning="yes" metrics-url "<Path of Metrics File>" embed-
url "<Path of TTF/TTC file>">
      <font-triplet name="<Custom Font Name>" style="normal"
weight="normal"/>
    </font>
    <font kerning="yes" metrics-url "<Path of Metrics File>" embed-
url "<Path of TTF/TTC file>">
      <font-triplet name="<Custom Font Name>" style="normal"
weight="normal"/>
    </font>
  </fonts>
</renderer>
</renderers>
</fop>

```

Sample Configuration File

```

<fop version="1.0"> <strict-validation>true</strict-validation>
<renderers> <renderer mime="application/pdf"> <filterList>
<value>flate</value> </filterList> <fonts> <font embed-
url="C:\Windows\fonts\msgothic.ttc" kerning="yes" metrics-
url="D:\Program
Files\Cordys\defaultInst\custom\metricFiles\msgothic.xml"> <font-
triplet name="MS Gothic" style="normal" weight="normal"/> </font> <font
embed-url="C:\Windows\fonts\Vani.ttf" kerning="yes" metrics-
url="D:\Program Files\Cordys\defaultInst\custom\metricFiles\vani.xml">
<font-triplet name="Vani" style="normal" weight="normal"/> </font>
</fonts> </renderer> </renderers></fop>

```

- c. Set `bpm.designer.report.use.custom.fonts`:
 - Specify whether you want to enable the custom fonts.
 - The values are **true** or **false**.

- d. Set `bpm.designer.report.fonts.config.file`:
 - Provide an absolute path of the configuration file.
 - For example:
 D:\Program

 Files\Cordys\defaultInst\custom\config\fontconfiguration.xml

Validating a business process model

A business process should be validated, published to runtime, and tested before it is deployed.

A general criterion for a well-designed business model is that it should not throw any warnings or error messages when validated. For a business process model, it should generate an executable BPML. Warnings that are generated during validation are classified as Error, Warning, and Info.

To validate a business process model:

1. Select a starting point and select  (Business Process Model) to open an existing business process model.
2. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to **Workspace Documents > <Solution> > <Project>**, right-click the business process model and select **Business Process Execution > Validate and generate BPML**.
The Build progress window appears and when the build is over, if there are any warnings or errors, then the warnings window appears automatically. If there are warnings, resolve them and re-validate the business process model.
3. To view warnings after validating the business process model, click **Show Warnings** or **F6**.
4. Click a warning to identify the location where the error has occurred and rectify it suitably. Resolve the errors and re-validate the business process model.

The business process model is validated.

Debugging a business process model

The Process Debugger helps you test a business process step-by-step and trace any problems that occur during execution.

The Process Debugger has two modes of operation:

- **Run Interactively:** This mode gives a view of how the business process is executed.
- **Debug:** This mode provides additional options such as viewing and changing input and output messages. You can start this mode using the Process Instance Manager (PIM).

If there are any parallel flows in a business process, they execute in parallel during runtime, although in the Process Debugger, the process is debugged in a sequential order.

To debug a business process model:

1. Select a starting point and click  (Business Process Model) to open a business process model.
If you have the business process model already opened in the business process modeling environment, then perform Step 2.

2. Right-click in the AppWorks Platform business process modeling environment and select **Business Process Execution > Debug**. Alternatively, right-click the process model in the AppWorks Platform menu and, select **Business Process Execution > Debug**. The Process Debugger window opens.
3. Select the **Execute Method** check box to execute SOAP applications in the process. This check box is selected by default. If you clear the check box, the output message is generated from the application WSDL, and the output parameter is displayed in the **Message** tab. In this case, you can also type the output parameters manually.
4. Select the **Send Notifications to Users** check box to send notifications to users' Inbox.
If the check box is not selected, notifications are displayed in the Process Debugger.
5. Select the **Step into Subprocess** check box to debug activities in the sub-process. This check box is selected by default.
If you clear this option, the sub-process will not be executed by the debugger and you will need to hard-code the output message (if any) of the sub process in the main process message map.
6. Click **Process Options** on the toolbar in the business process modeling environment and select an option.

The Process Debugger starts debugging the business process model.

In the *Process Debugger* window, the **Process Name** section shows the description of the business process model.

The **Status** section shows information on the status of the business process model (whether it is waiting for input or giving some information.)

The **Message** tab displays the messages sent or received. The message can be an Input Message, Output Message, or Assignment. The name of the tab varies depending on whether it is an Input Message or an Output Message. The Message Map tab displays the complete message map until the point of execution of the Business Process. While editing an activity, you must indicate the changes in the Message Map tab and not the Message tab.

The business process is debugged.

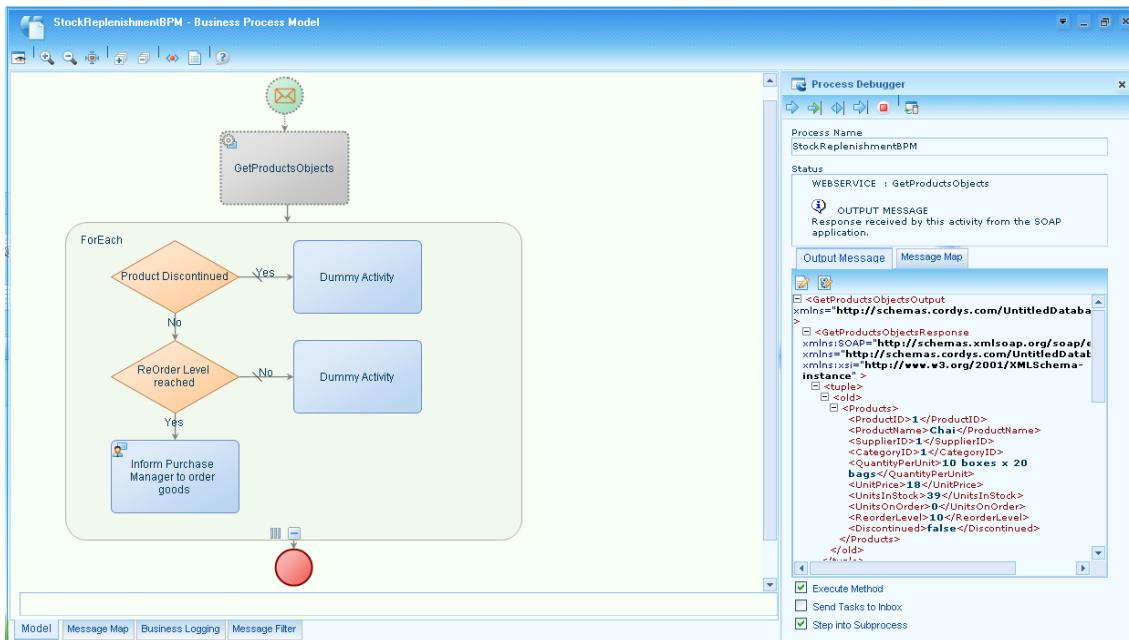
Example of debugging a business process

The following example describes the procedure to debug the Stock Replenishment process. The process is designed to retrieve product information, check if the product reorder level is reached, and inform the Purchase Manager to reorder stock.

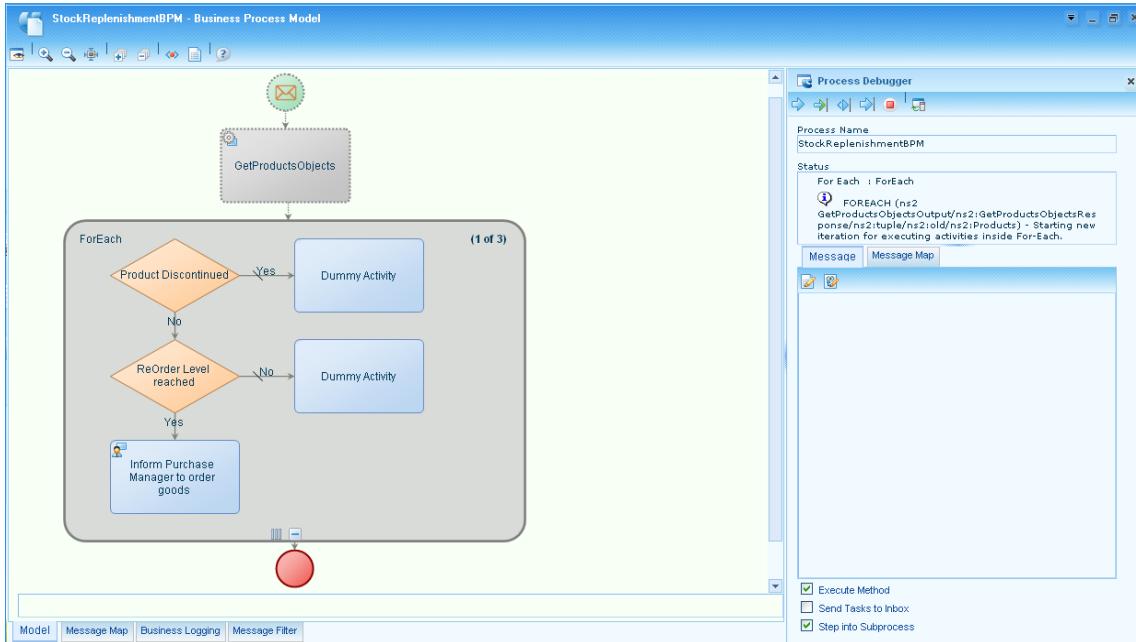
To follow an example of debugging a business process:

1. **Select a starting point** and click  (Business Process Model) to open an existing business process model.
2. From the **Workspace Documents (Explorer)** window, click <project> > <StockReplenishmentBPM>. The business process model appears in the process modeling environment.

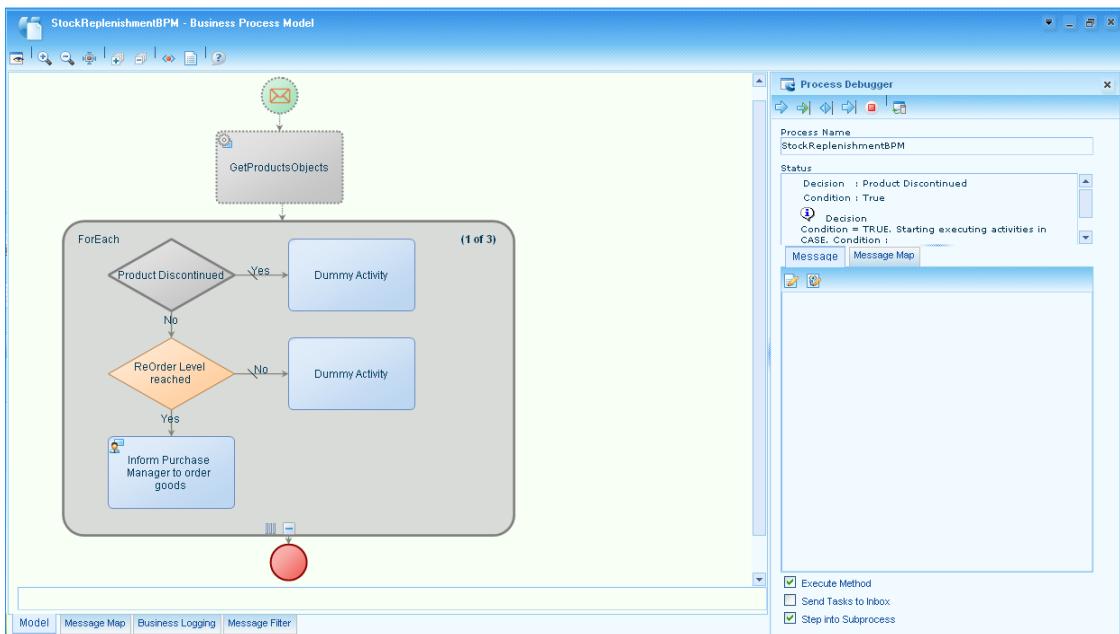
3. Right-click in the AppWorks Platform business process modeling environment and select **Business Process Execution > Debug**.
The *Process Debugger* window opens.
4. On the Process Debugger toolbar, click ➡ (Activity by Activity) or ➡ (Step by Step).
 - If you select the **Step by Step** option, every assignment in the business flow appears in the **Status** section. The business process appears in the **Read Only Editor** that has configurable toolbar items.
 - If you select the **Activity by Activity** option, the debugging starts with the Get Products activity, as shown in the following figure.



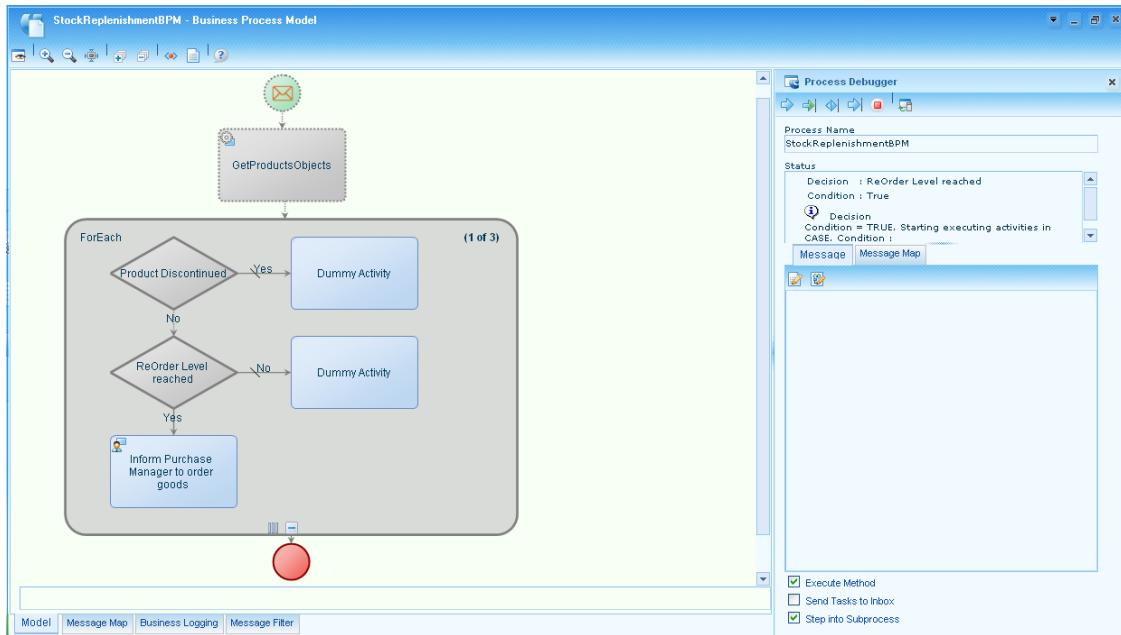
5. To debug the next activity, click **Activity by Activity** again.
The Process Debugger checks the For Each loop. The debugger at this stage appears as follows.



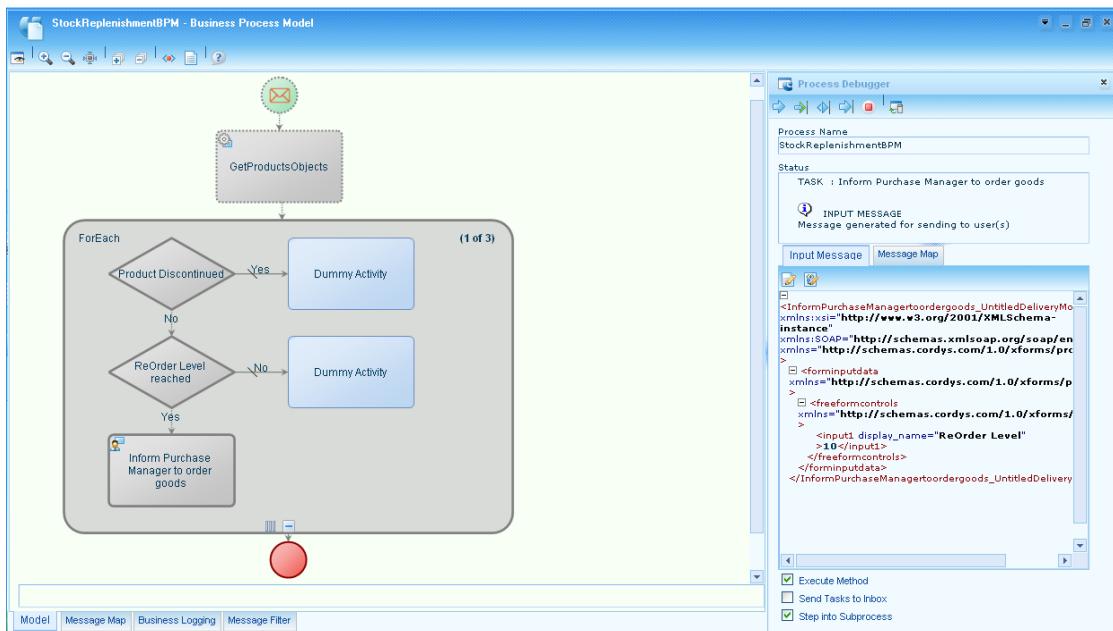
6. The debugger checks for the Products Discontinued activity in the For Each loop.



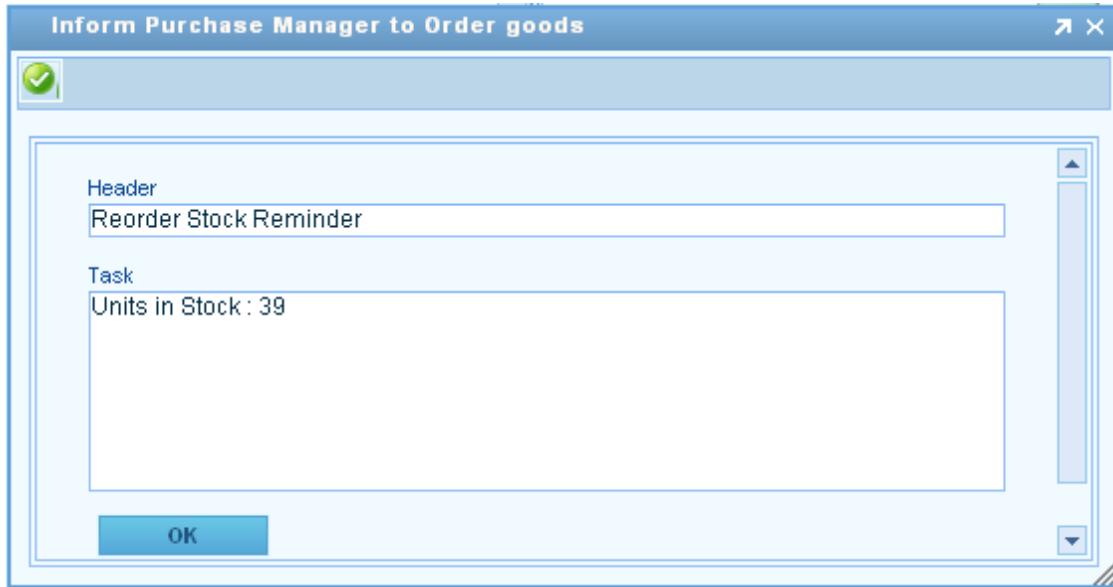
7. Because the product is not discontinued in the first iteration of the For Each loop, the debugger checks the ReOrder Level reached activity.
The debugger at this stage appears as follows.



8. Because the reorder level is reached, Units in Stock (3) is less than the Reorder Level (5), a notification is sent to the Purchase Manager to reorder stock.



9. The Purchase Manager receives the following notification.



After completing one iteration, the debugger returns to the next iteration and repeats the steps described in the example.

Configurable toolbar items in business process and case modelers

The following table describes the toolbar items that you can configure and the modes in which they are available.

Configurable Item	Description	Availability
Business Process Model Help (F1)	Click this icon to view product documentation on the business process model or its process instance.	Process view in design mode and instance view in debug mode.
Collapse All Groups	Displays the collapsed view of all activity groups.	Instance view and debug modes
Expand All Groups	Displays the expanded view of all activity groups, if any, showing sub-process etc.	Instance view and debug modes
Fit in Window	Fits the image to window for better viewing.	Process view, instance view in debug modes
Hide/Show Warnings	Displays or hides any warnings that might appear when publishing or executing the process model	Process view in design mode, instance view in debug modes

Configurable Item	Description	Availability
Print Preview	Displays a preview of the model.	Process view in design mode, process instance view in debug modes, case modeler and case instance in debug mode
Process Options	Displays the various options or operations that can be performed on the process instance.	Instance view and debug modes
Refresh	Refresh is used to restore the process instance to the last saved state of the business process model discarding all changes made after that by the user. You may select either Refresh or Auto Refresh options to have an updated view of the process instance.	Process view in design mode and instance view in debug mode.
Set/Remove Breakpoint	Executes the activities until the set breakpoint.	Process instance in debug mode
Zoom slider	Depending on whether you are minimizing or maximizing the view of the model, zoom slider displays a view of the model as required.	Business process modeler, Case modeler, Organizational modeler, and MDM modeler.

Executing a process model

Once you model a business process as you want it to-be, you need to execute it so as to ensure that the executed process is always the same as the modeled process. Executing a business process model helps to ensure that orchestration, integration and collaboration with other business processes, workflow management processes, Web services, and other business units is achieved as intended.

The Business Process Engine is an important component of the AppWorks Platform Business Process Management Suite (BPMS) and forms the execution layer of the system. The Process Engine executes process models that are designed in the AppWorks Platform business process modeling environment. The process models are stored in the Business Process Engine as BPML documents. When the Process Engine receives a request, it loads the process model from the repository using the name of the process, prepares its executable copy, and instantiates the process. Some of the salient features of Process Engine are crash recovery, reliable messaging and failover.

Execution scenario of a runtime reference

A business process model (or any other content) can exist in two spaces - ISV and Organization. Any content from the ISV space, when customized, gets stored as organizational content for the Organization space in which it is customized. This is the usual behavior of the AppWorks Platform. In case of business process models, there is no explicit way to customize it. However, you may face a situation where, for example, a business process model that is created in Organization "A", is published and executed. An application package of the CWS project is created and deployed on to the same machine on which the business process model was created. Now, there are two business process models deployed when the application package is loaded - one in Organization space and another in the ISV space.

In this case, the Business Process Engine at the time of execution, searches for the business process model first in Organizational space. Next, it searches in the ISV space only if the process model is not available in Organizational space.

In this context, when the ExecuteProcess SOAP request is sent, by default, the lookup for a business process happens in Organization space and if the business process is not available, the lookup happens in the ISV space. There is also an option to explicitly define the scope in the ExecuteProcess SOAP request. Where, the <modelSpace> parameter should be filled with "Organization" or "ISV" as values.

In case of business process model look up during execution, the Business Process Engine also follows same approach for look up, which means that the Process Engine first searches for the business process model in Organizational space. Next, it searches in the ISV space only if the process model is not available in Organizational space.

However, if the business process model is available in both the spaces - ISV and Organization, the organizational copy is considered for execution. If there is a need to always select the business process model from ISV space, technically it is possible by storing the model space in the BPML along with process name. However, if the model space is hard coded for look up of sub-process based on selection from run-time reference in the modeler, then the solution fails. This happens if the sub-process is customized and published to the Organization space. Also, this will not be as per AppWorks Platform standards for customization of content.

The following tasks are performed during or after execution of the business process model:

- [Publishing a Process Model](#)
- [Instantiating a Process Model](#)

Instantiating a business process model

The Process Engine executes processes modeled in the AppWorks Platform business process modeling environment.

You can instantiate a process model in any of the following modes.

Mode	Procedure
Executing a Process Model	This mode provides a view of how the business process is executed and allows you to interact with the execution of the business process. Right-click in the business process modeling environment and select Business Process Execution > Run Interactively . Alternatively, press the Shift + F12 keys.
Run	This mode provides a view of how the business process is executed. Right-click in the business process modeling environment and select Business Process Execution > Run
Debugging a business process model	This mode provides options such as viewing and changing input and output messages. You can start this mode using the Process Instance Manager (PIM). Right-click in the business process modeling environment and select Business Process Execution > Debug . Alternatively, you can press the CTRL + F12 keys.

Viewing process instances

The Process Instance Manager (PIM) enables you to view individual process instances and a summarized view of each published business process. It depicts the total number of instances and the number of instances per status. It provides an overview of the running instances and their states, enabling you to take immediate action to suspend a running or a waiting instance, or restart an aborted instance.

To access the process instance viewer:

1. Navigate to the Process Instance Manager.
2. The list of executed business processes are shown along with the number of process instances (segregated by Status) and the total number of process instances of the corresponding business process.
3. Click on the specified number (number of process instances).
The Instances by Process Definition dialog box opens showing the list of process instances of a business process that are in given status.
4. Alternatively, click on the Total Number of the process instances to view the list of all process instances of the process in Instances by Process Definition dialog box.
5. From the list of process instances obtained, select a process instance and click **Show Graphical View** in the tool bar or from the context menu.
6. The read only view of the process is shown along with some symbols and color coding explaining the execution of that particular process instance.

Interpretation of Process Instance View

- Single, thick border around an activity denotes that the activity executed.
- Double-walled green colored border around an activity denotes that the process execution is in that activity.

- Completed symbol in an activity denotes that the activity is in Completed state.
- Pause symbol in an activity denotes that the activity is in Waiting state.
- Suspended symbol in an activity denotes that the activity is in Suspended state.
- Red icon (with symbol similar to Close) denotes that the activity terminated.
- Red colored border around an activity denotes that the execution of the activity.
- Red colored broken-line border around an activity and a Red icon (with a single diagonal line) in the activity denotes that the activity aborted.
- Blue colored broken-line border around an activity and a Skip icon in the activity denotes that the activity has been skipped.

Viewing a process instance message map

Before you begin:

- You must have the Process Administrator role in AppWorks Platform Business Process Engine application package to view the Message Map of a process instance.

To view a process instance message map:

1. If your starting point is **Process Instance Manager (PIM)**:

Process Instance Manager (PIM)	<ol style="list-style-type: none">a. In the My Applications App Palette, click  (Process Instance Manager). The Process Instance Manager is displayed with a list of business processes that have process instances.b. Click Find () on the toolbar. The Filter Process Instances window opens. If you do not specify the filter criteria for process instances, the default filter options are considered based on which the process instances appear.c. Type the selection criteria.d. Click OK. All process instances that match the specified criteria appear.e. Click the instances having the status for which you want to view the process instance details. The Instances by Process Definition window opens.
--------------------------------	--

2. If your starting point is **Sub-Proceses**:

Sub-Proceses	<p>a. In My Applications App Palette, click  (Process Instance Manager).</p> <p>The Process Instance Manager displays a list of business processes that have process instances.</p> <p>b. Click Find () on the toolbar.</p> <p>The Filter Process Instances window opens. If you do not specify the filter criteria for process instances, the default filter options are considered based on which the process instances appear.</p> <p>c. Type the selection criteria.</p> <p>d. Click OK.</p> <p>All process instances that match the specified criteria are displayed.</p> <p>e. Click  (Show Sub-Proceses) to display the sub-processes.</p> <p>The Sub-Proceses pane appears.</p>
--------------	---

3. Select the required process instance and click  (Show Message Map) on the toolbar. Alternatively, right-click the process instance and select **Show Message Map**. The Message Map Data - View and Edit the Message map Data dialog box opens.
4. Click the **All View** tab to view all the messages.
5. To copy all the messages from the Message Map Data - View and Edit the Message Map Data to the Clipboard, click the All View tab and click  on the toolbar.
6. To select and view a specific message in the Message Map Data - View and Edit the Message map Data dialog box, click the **List View** tab.
7. To copy the currently displayed message to Clipboard, click the List View tab and click  on the toolbar.
8. Click the navigation buttons on the toolbar to navigate between the messages that appear in the Message Map Data - View and Edit the Message map Data dialog box. All input and output messages of the activities in the process are displayed according to the option you select.

You have successfully viewed the message map of a process instance.

Viewing and editing the XML structure of a WS-AppServer model

Before you begin:

- Create a data model using WS-AppServer Package generator.

The XML structure presents the detailed layout of the data model. It displays all the methods, attributes, and relations that exist in that model. It also displays whether a class is inherited or derived.

The option to edit is helpful when you want to add more attributes or methods to the model in the XML editor itself, without having to use the context menu. The moment you save the changes in the XML editor, they immediately reflect on the UI.

1. Double-click the  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > **<solution>** > **<project>**.
The WS-AppServer Package window opens, displaying all the models and their corresponding attributes, relations methods, and parameters.
2. In the **Models** box, right-click a model and select **View/Edit XML**.
The XML dialog box opens, displaying the structure of the model in XML.
3. Modify the layout of the model by adding or removing attributes, methods, and parameters, if required.
For example, if you add an attribute, it is added to the list of attributes on the Attributes tab.

Tip: It is not mandatory to modify the XML structure. You can use this option just to view and study the XML structure of the model and replicate it in another custom model.
4. Click .
The changes, if any, are saved and the details of the model on the UI are updated accordingly.
5. Close the XML dialog box.

You have viewed the XML structure of the WS-AppServer model.

XPDL support

The XML Process Definition Language (XPDL) is a format standardized by the Workflow Management Coalition (WfMC) to interchange Business Process definitions between different workflow products, that is, between different modeling tools and management suites. XPDL defines an XML schema for specifying the declarative part of workflow/business process.

XPDL is designed to exchange the process definition, both the graphics and the semantics of a workflow business process. XPDL is currently the best file format for exchange of BPMN diagrams; it has been designed specifically to store all aspects of a BPMN diagram. XPDL contains elements to hold graphical information, such as the X and Y position of the nodes, as well as executable aspects that would be used to run a process. This distinguishes XPDL from BPEL, which focuses exclusively on the executable aspects of the process. BPEL does not contain elements to represent the graphical aspects of a process diagram.

BPMN is a visual process notation standard from the OMG, endorsed by WfMC, and broadly adopted across the industry. But the BPMN standard defines only the look of how the process definition is displayed on the screen. How you store and interchange those process

definitions is outside the scope of the standard, and this is where XPDL comes in. XPDL provides a file format that supports every aspect of the BPMN process definition notation including graphical descriptions of the diagram, as well as executable properties used at runtime. With XPDL, a product can write out a process definition with full fidelity, and another product can read it in and reproduce the same diagram that was sent.

XPDL is extensible so that it allows each different tool to store implementation specific information within the XPDL, and have those values preserved even when manipulated by tools that do not understand those extensions. This is the only way to provide for a "round trip" through multiple tool and still be able to return to the original tool with complete fidelity.

AppWorks Platform allows you to [Importing an XPDL file](#) and [Exporting a business process model to an XPDL format](#)[Exporting a business process model to an XPDL format](#)[Exporting a business process model to an XPDL format](#)

Importing an XPDL file

You can import business processes designed in other business modeling environments, in XPDL 2.0 format. However, the following XPDL constructs will not be imported:

- DataTypes
 - ArrayType
 - EnumerationType
 - ListType
 - RecordType
 - UnionType
- Association/Data Object
- Category
- Code Page
- Conformance class
- Simulation Data
 - Cost
 - CostUnit
 - Duration
 - Length
 - Limit
 - Precision
 - Scale
 - Working Time

- Waiting Time
- TimeEstimation
- External Packages / Reference
- InputSets / OutputSets
- MessageFlow / MessageType
- ValidFrom and ValidTo
- PartnerLinks / PartnerLinkTypes
- Artifacts
 - Text
 - Text Annotation
 - Transparent Text

To import an XPDL file:

1. In the Workspace Documents, right-click the <Project> and select **Import**.
The Import Wizard appears.
2. Select the **Model Type** and the **Import Plug-in** from the list, and click **Next**.
Only the registered plug-ins of type Import appear in the list. The next page of the Import Wizard appears. Files with the extension .zip are considered for import.
3. Click  with **File to Import** to specify the source path of the .zip file to import.
4. Click  with **Location to store the imported models** to specify the target path where you want to store the .zip file.
5. Click **Finish** to exit the wizard.

The XPDL file with extension .zip is imported into the AppWorks Platform business modeling environment.

After you complete this task:

- After importing the business process models, attach the appropriate methods and roles.
- Complete assignments in the Message Map before you validate and publish the business process model.

Exporting a business process model to an XPDL format

This option helps you to export a business process model in AppWorks Platform to a common industry standard format, using the modeled export plug-in.

1. In Workspace Documents, open <Project> and right-click the business process model that you want to export.
2. Select **Export**.

The Export Wizard window opens.

3. Select the **Model Type** and the **Export Plug-in** from the list, and click **Next**.

The modeled export plug-in that was published to an organization appears in this list.

4. Do all of the following:

- a. Click  icon attached with Model to export to specify the path of the required business process model.

- b. Specify the **Export File Name** for the business process model being exported.

- c. Click **Finish** to exit the wizard.

A confirmation message appears displaying a link to the exported file and indicating successful export of the selected business process model.

5. Click **Download exported file** link to download the exported file on to your system or to any other system within your network.

You have successfully exported the selected business process model to an XPDL format.

Printing a business process model

1. From the Workspace Documents (Explorer view), click <Solution> > <Project> > <Business Process Model>.
The business process model appears.
2. Click  (Print Preview) on the toolbar.
The Print Preview window opens. For more information, see [Print Preview Interface - reference](#).
3. Adjust the size of the image.
4. Click  in the Print Preview window.

Install the corresponding fonts on the AppWorks Platform installation server for multibyte character support while printing required business process model.

The required business process model is printed on the default printer.

Importing VCMDATA or Zip files

Before you begin, do the following:

1. Load the ISVP containing Web services and XForms used in the business process models of vcldata. If the ISVP is not loaded, you can continue with the migration; however, while opening the migrated model, you will encounter the error "WSDL is missing for the activities" and the execution of the processes may not be successful.
2. Ensure that CORDYS_HOME environment variable is set to your <AppWorks Platform_installdir>.

3. If you are using Windows OS, ensure that PATH environment variable has entry for %CORDYS_HOME%\lib folder.
4. Ensure that the Rule Repository and Scheduler Services are started.
5. Unzip the .vcmda or .zip file to a temporary location on the server.
6. If you are migrating through a different user context, do the following:
 - Access the wcp.properties file in the <CordysHOME>\config folder.
 - Add the user.name property. Specify the value as the user whose id is used to migrate. For example: user.name = john

Note

- Only OpenText AppWorks Platform Users can import VCM data, because internally VCM Data tool uses CWS API and LDAPs which OpenText AppWorks Platform Users can access. Non-OpenText AppWorks Platform Users cannot import because they need to have Administrator and Developer roles to import VCM Data.
- The import of Cordys BOP C3 VCM data with organization role is not supported.

In earlier release of Cordys, there was an option to import/export Business Process models between different environments using Content Transfer Utility feature. This feature has been enhanced with a new Collaborative Workspace feature called 'Synchronization'. However, to support backward compatibility for the exported .vcmda/.zip files, AppWorks Platform provides a migration tool to import such content into its environment.

To import VCMDa or ZIP files:

1. Go to <Cordys_HOME>\webroot\shared\com\cordys\bpm\plugins\import\vcmda.
2. Modify the vcmda migration properties with:
 - a. Specify the name of the Organization (specify only the organization name and not the dn. For example, system organization.)
 - b. Specify the name of the Workspace (The workspace should have been created.)
 - c. Specify the name of the Project (The project should have been created in the relevant workspace.)
 - d. Unzip the vcmda location (specify the absolute path of location).
3. Do any one of the following:
 - In Windows: Run the tool vcmdatamigrator.bat
 - In Linux: Run the tool vcmdatamigrator.sh

If the .sh script does not execute properly, then set LD_LIBRARY_PATH=\$Cordys_HOME/lib.

4. If you are using Linux, do one of the following:
 - If the database that you are using is Oracle, then use the following line; otherwise, comment the line:

```
export MIGRATION_CP=$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$MIGRATION_CP
```

By default, this line is commented.

5. If you are using Windows, add the following code lines to `vcmdatamigrator.bat` file:

- If the database that you are using is Oracle, then uncomment the following line:

```
set MIGRATION_CP=$ORACLE_HOME/jdbc/lib/ojdbc14.jar;%MIGRATION_CP%
```

The code lines are added to the `vcmdatamigrator.bat` file.

The `.vcmdata` or `.zip` files are imported into AppWorks Platform business process modeling environment.

Creating a business identifier

Business identifiers help you to associate business data to process instances, making it possible to search for particular instances in Process Instance Manager.

To create a business identifier:

1. Select a starting point and click  (Business Identifier).
The *Untitled Business Identifier - Business Identifier* window opens.
2. Provide the following information:

Name	Type a name for the business identifier. This name is used in the message map for value assignments. Furthermore, use this name must to filter or search process instances through Process Instance Manager. Follow a naming convention to avoid naming conflicts for the business identifiers.
Description	Type a meaningful description for the business identifier. This description appears as a label in the User Interface.
Data Type	<p>Select the required data type for the business identifier. Available data types are String, Numeric, and Date.</p> <ul style="list-style-type: none"> ■ The Precision field is enabled only if the data type selected is Numeric. Precision denotes the maximum number of significant decimal digits permitted. ■ If you select the Data Type as Date then ensure that the Date format is in XSD date time format.

3. Click  to save the business identifier.

You have successfully created a business identifier. It is available in the workspace and can be used in any project within the current workspace. It is also possible to translate just the business identifier. The description that is provided can be translated.

Switching to the improved business process modeling environment

The business process modeling environment in AppWorks Platform has been improved to augment Business Process Modeling Notation (BPMN) compliance. The improvements bring changes in the way business process models are designed, executed, and displayed in the Process Instance Manager. There are also improved error messages and warnings. To take advantage of these improvements, you need to switch your business process model to AppWorks Platform.

Switching a business process

Users can switch their existing business processes that are designed in earlier versions of AppWorks Platform only through the business process modeling editor.

Switching a business process might require you to change its design to achieve the intended behavior. Ensure that you are familiar with the behavioral changes prior to making the switch. See [Behavioral changes of business process models in AppWorks Platform versions](#) for details.

To switch a business process to the improved business process modeling environment:

1. In the Workspace Documents (Explorer) view <Project>, open the business process that you want to switch.
A confirmation dialog box opens.
2. In the confirmation dialog box, indicate whether you want to switch to the improved business process modeling environment now or later.
3. To switch now, select the check box for **I have read the documentation and understood the behavioral differences** and click **Yes**.
The business process is switched to the improved modeling environment.
4. To switch later, click **Later**.
The confirmation dialog box closes and is displayed the next time you open a business process that is designed in a version earlier than AppWorks Platform.
5. If you do not want to see the confirmation dialog box every time you open a business process, you can select the check box for **Do not show this message again**.
6. When you are ready to switch the business process, click  (Switch to the improved business process modeling) on the toolbar.
The confirmation dialog box opens.

After switching to the improved business process modeling environment, you can redesign your business processes, if required and use new and improved features.

Reverting a switched business process

After switching a business process, you can revert the business process to the earlier AppWorks Platform version from which you made the switch, if required. However, you cannot use the business process modeling editor to do so. The process to revert varies based on whether the workspace content is stored in a Source Control Management (SCM) repository such as Subversion (SVN).

To revert a business process to the previous version:

1. In the workspace, go to the project that contains the business process that you switched.
2. Do one of the following:
 - If no changes are committed to the project from the time you made the switch, it is recommended that you revert all your local changes. See [Reverting Changes Made in the Workspace](#).
 - If changes are committed to the project from the time you made the switch, do the following:
 - a. Revert your current local changes made to the project and synchronize the changes with the file system. See [Synchronizing Workspace and File System](#).
 - b. From the file system, revert each business process, one by one, to the version from which it was switched using the SCM command or the user interface.
 - c. Synchronize the changes with the workspace. See [Synchronizing Workspace and File System](#).
3. Verify other functionality in the project with the reverted business processes.
4. If you encounter issues, revert the complete workspace content to the version from which the business processes were switched using the SCM command or the user interface.

To revert workspace content not stored in an SCM repository:

1. Before you begin, you need a copy of the workspace content (ZIP file) before switching the business process.
2. Extract the ZIP file of the workspace content, which contains the previous version of the business process, into a temporary folder.
3. Synchronize the changes with the workspace. See [Synchronizing Workspace and File System](#).
4. In the synchronization folder, delete the project folder that contains the business process to be reverted.
5. Copy the relevant project content from the extracted folder to replace the project folder that you deleted in the previous step.
6. Synchronize the changes with the workspace.
7. Verify other functionality in the project with the reverted business process.

8. If you encounter issues, revert the complete workspace content using the ZIP file that you extracted, synchronize again, and verify the business processes.

Behavioral changes of business process models in AppWorks Platform versions

AppWorks Platform brings some improvements to the way business process models are designed. These improvements have resulted in behavioral changes in AppWorks Platform to constructs, embedded sub-processes, activity assignments, exception handling, display in the Process Instance Manager, and so on when compared to earlier versions of AppWorks Platform. Warning and error messages have been improved to minimize modeling errors and strengthen Business Process Modeling Notation (BPMN) compliance. Therefore, as a best practice, do not ignore the warnings.

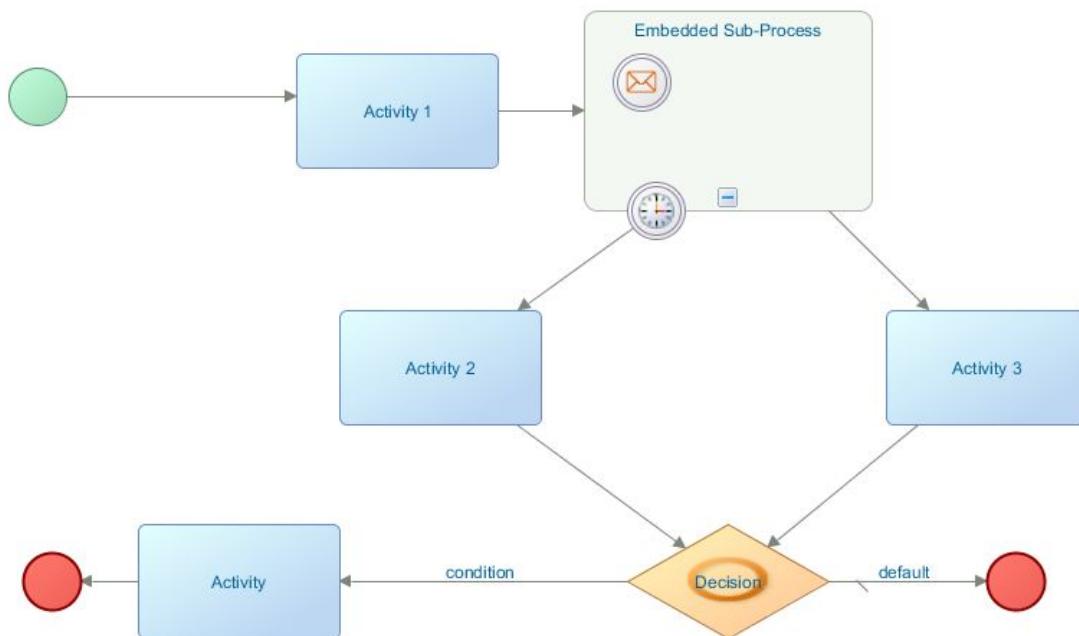
The behavioral changes are present in design time, runtime, and the Process Instance Manager (PIM) display.

Design time

The following changes require you to remodel your business process in AppWorks Platform to achieve the optimal behavior.

Embedded sub-process with normal and timeout paths merging at an inclusive decision

A business process has an embedded sub-process with two paths that merge at an inclusive decision: a timeout path (Activity 2) and a normal path (Activity 3).



Previous versions	When the business process is executed, only one path from the embedded sub-process is execute: either the timeout path (Activity 2) or the normal path (Activity 3). The decision does not wait for the other path to execute and the process ends.
Current version	A business process has an embedded sub-process with two paths that merge at an inclusive decision: a timeout path (Activity 2) and a normal path (Activity 3).

When the business process is executed, only one path from the embedded sub-process is executed: either the timeout path (Activity 2) or the normal path (Activity 3). The decision waits for the other path to execute and therefore, the process waits endlessly.

- You must remodel the business process to use a decision construct of the type Exclusive instead of Inclusive.

To share Message Map property for a sub-process:

- On the Properties pane of the sub-process, click the **Messages** tab.

Previous versions	The Share Message Map check box is enabled.
Current version	The Share Message Map check box has been removed from the user interface. This feature was deprecated in a prior AppWorks Platform version. See the Independent Sub-Process Properties Interface topic in the product documentation.

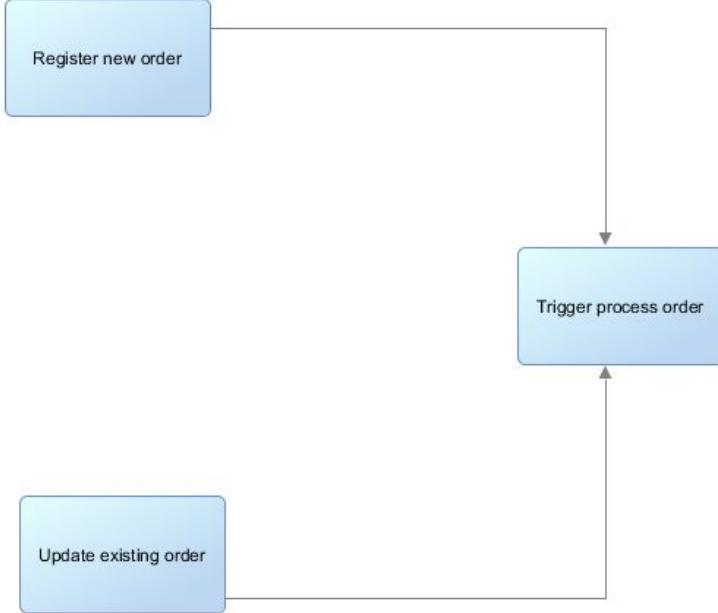
Condition for a single outgoing connector for a decision

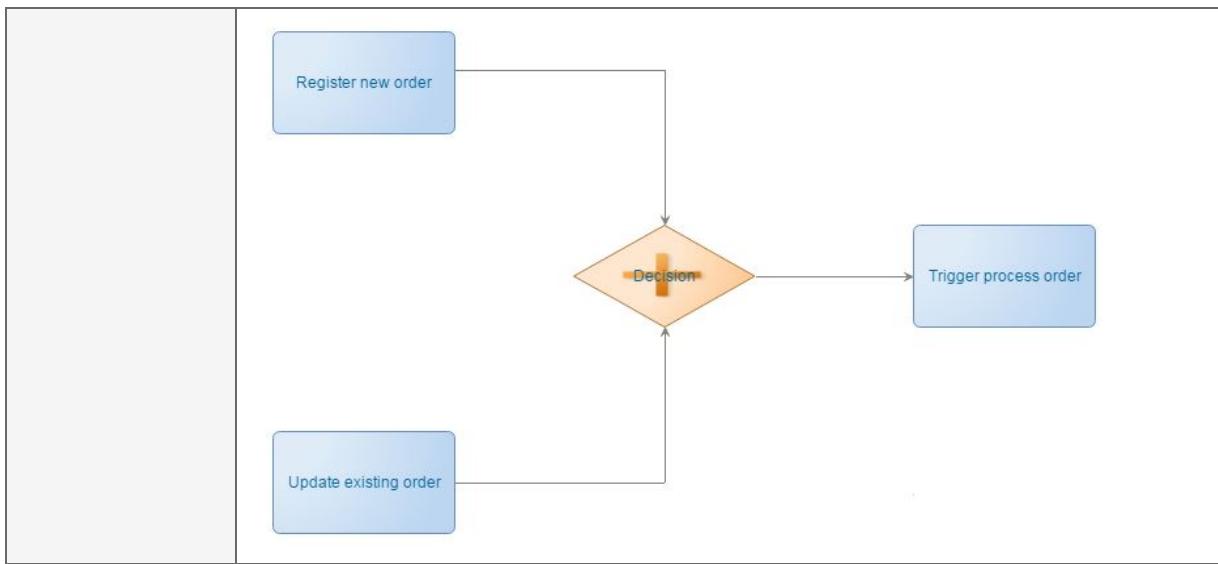
A decision construct in a business process has only one outgoing connector.

Previous versions	You need to define a condition for a connector even if it is the only outgoing connector.
Current version	The Conditions tab on the decision property pane is hidden and the decision is implicitly considered a merge because only one path can be executed. The Conditions tab is visible only if there is more than one outgoing connector. Designing a model with conditions for outgoing connectors of a parallel decision is invalid. Therefore, if the decision type is Parallel, you cannot define conditions even if the decision contains more than one outgoing connector. The Conditions tab remains hidden.

Uncontrolled merge

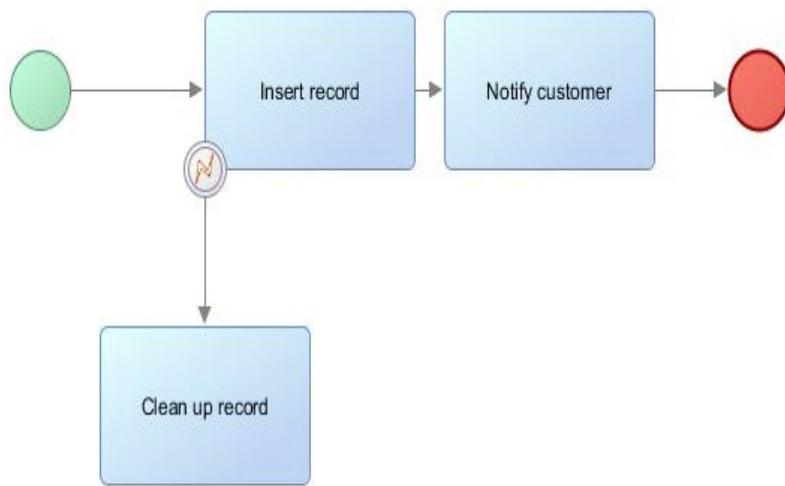
A business process has an uncontrolled merge with two parallel activities: Register new order and Update existing order.

Previous versions	<p>When the business process executes, the two parallel paths implicitly merge at the Trigger process order activity and the business process execution continues only after the merge.</p>  <pre> graph TD A[Register new order] --> B[Trigger process order] C[Update existing order] --> B </pre>
Current version	<p>When the business process executes, the two parallel incoming paths cause the Trigger process order activity to execute twice, followed by the remaining activities.</p> <p>You must remodel the business process with a decision construct of type Parallel to merge multiple incoming paths and to execute the Trigger process order activity only once.</p>

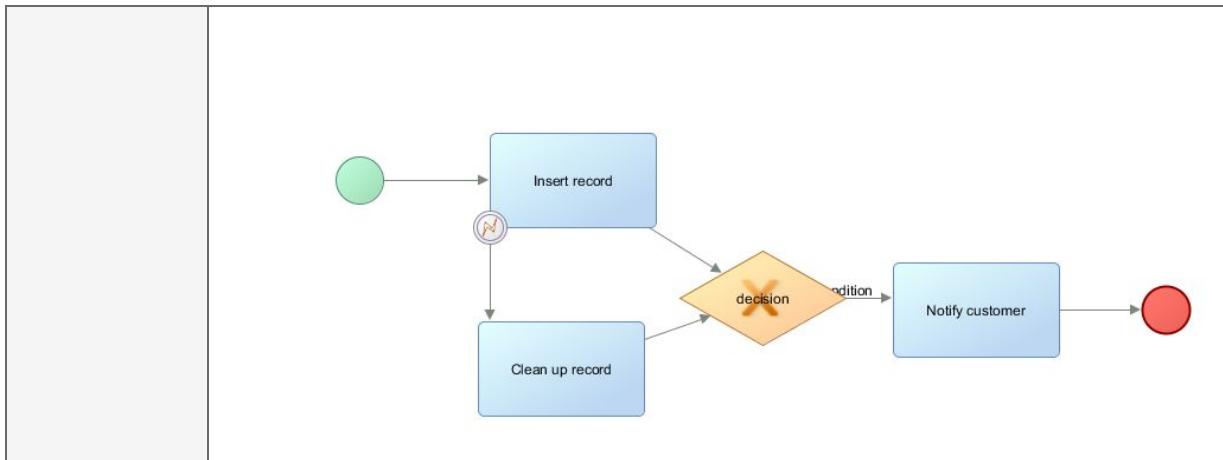


Exception handling

A business process encounters an exception when inserting a record.



Previous versions	After handling the exception (executing the <code>Clean up record</code> activity), the execution continues to the <code>Notify Customer</code> activity.
Current version	After handling the exception (executing the <code>Clean up record</code> activity), it is not possible to continue the execution of the <code>Notify Customer</code> activity because there is no execution flow between the <code>Clean up record</code> and <code>Notify Customer</code> activities. You must remodel this business process in a manner where the incoming paths are merged at an exclusive decision to achieve the same behavior.



Business process without an End event

A business process does not have an End event.

Previous versions	Validating such a process is successful with warnings.
Current version	Validating such a process fails. You must add an End event for successful validation.

Crash recovery for a short-lived business process

The Enable Crash Recovery option is available in the business process model properties.

Previous versions	The crash recovery functionality is enabled for business processes of type Short Lived, Long Lived, and Page Flow. When it is enabled, if the Business Process Management service container crashes and becomes available again, short-lived business process execution is recovered and begins from the point where the crash occurred.
Current version	The crash recovery functionality is disabled for business processes of the type Short Lived but enabled for business processes of the type Long Lived and Page Flow. The life span of the short-lived process is expected to be very short (few seconds) and is expected to be synchronous and transaction-aware. The crash recovery feature is not useful when a user tends to perform the same action on immediately encountering an error. With the improved transaction management across resources and atomicity, crash recovery functionality for short-lived business processes is not relevant.

Business process with a Group construct

A business process has activities enclosed in a Group construct.

Previous versions	Publishing such a business process is successful.
Current version	After switching such a business process to the improved process modeling, publishing it might fail though the activities are properly enclosed in the Group construct. Ungroup and regroup the activities, and publish the business process again.

Compensate activity type

A business process is modeled with a Compensate activity that is linked to more than one activity in the business process.

Previous versions	Publishing of a business process succeeds when a Compensate activity is linked to more than one activity.
Current version	Publishing of a business process fails when a Compensate activity is linked to more than one activity.

Runtime

The following changes are seen during process execution but do not require you to remodel your business process.

Start event with an incoming message

A business process Start event is modeled with an incoming message, and the ExecuteProcess Web service is triggered with either an incorrect message or no message.

Previous versions	The following occurs based on the business process type: <ul style="list-style-type: none"> ■ Long-lived business process: A process instance is created, and the business process waits for the Start event to receive the correct message. ■ Short-lived business process: A process instance is not created, but a SOAP fault is displayed to the user with details of the expected message.
Current version	A process instance is not created regardless of the business process type. Instead, a SOAP fault is displayed to the user with details of the expected message. A process instance is created only if the Start event receives a valid message.

Pre-assignments and post-assignments for an activity

The activity in a business process is monitored and assignments for that activity fail.

Previous versions	Execution of the business process is aborted, but the activity status is not displayed.
--------------------------	---

Current version	Execution of the business process is aborted and the activity status is displayed as ABORTED.
------------------------	---

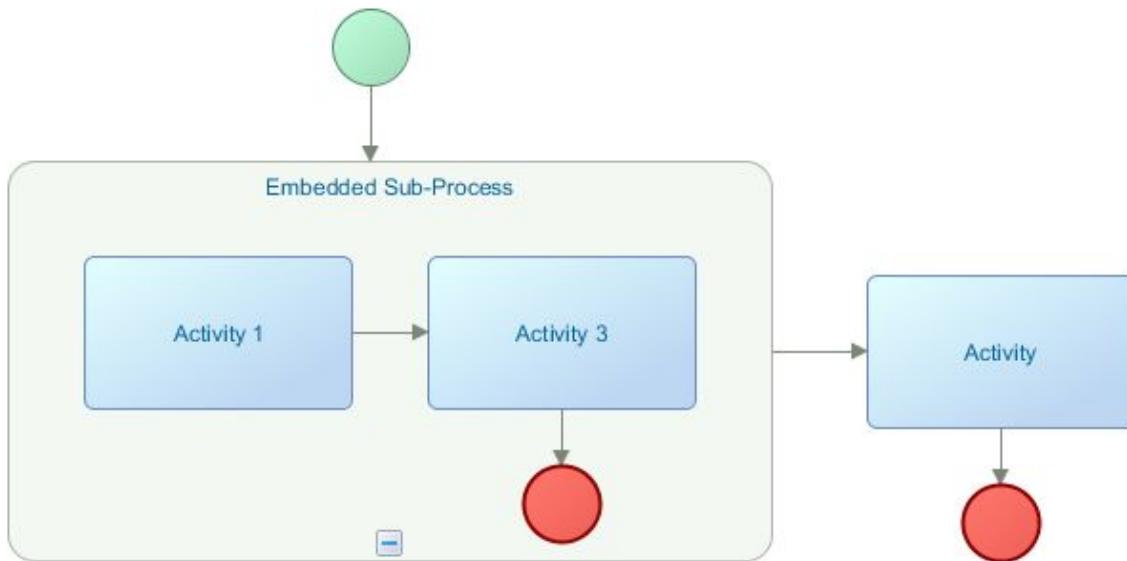
All conditions for a decision evaluating to false

A business process has a decision construct with three outgoing connectors. The conditions on the outgoing connectors evaluate to false at runtime.

Previous versions	When the business process is executed, it goes into the COMPLETED status.
Current version	When the business process is executed, it goes into the ABORTED status. If the business process is designed as intended and you want the status to be displayed as COMPLETED, you must add a default outgoing connector with an End event.

Multiple End events

A business process has more than one End event, one of which is encountered when there are other active paths.



For example, in this business process, the sub-process has an End event. There is also an End event outside the sub-process.

Previous versions	The business process completes when it reaches any of the End events even though there are other active paths. In the above scenario, process execution completes when it encounters the first End event (within the sub-process) and ignores the other active path (outside the sub-process).
--------------------------	--

Current version	The business process waits until all the active paths are complete before completing the business process. In the above scenario, the path outside the sub-process is also executed.
------------------------	--

Short-lived process execution

Short-lived business process execution is enhanced to make it transaction-safe and atomic.

Previous versions	Activities of a short-lived business process model are executed as part of different transactions. For example, a business process has two activities, Activity 1 and Activity 2. If Activity 1 completes and Activity 2 fails, Activity 1 is not rolled back.
Current version	Activities of a short-lived business process model participate in the same transaction. For example, a business process has two activities, Activity 1 and Activity 2. If Activity 1 completes and Activity 2 fails, Activity 1 is rolled back. See Transactional improvements in short-lived business processes.

Process Instance Manager display

The following changes are seen in the Process Instance Manager but do not require you to remodel your business process.

Decision as an independent activity

One or more outgoing paths of a decision are active.

Previous versions	The decision is in a waiting state until all its outgoing paths are complete. The outgoing paths of a decision are considered its children, which makes it a group activity. The decision is marked as Waiting in the Process Instance Manager (PIM) until all its active paths are executed.
Current version	The decision status is marked as Completed as soon as it is evaluated. The decision does not wait for other outgoing paths to be completed. In the case of Exclusive and Parallel decision types, after execution, the decision acts like an independent activity. It is executed and marked the same in the PIM even though one or more outgoing paths from the decision are still active.

Execute condition for an activity

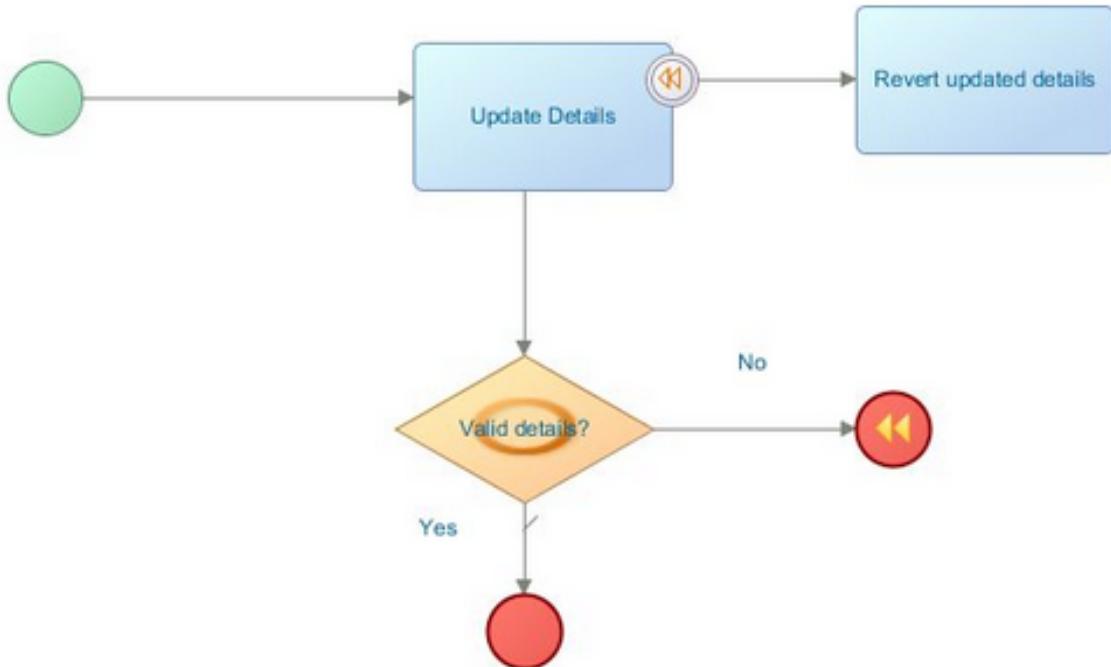
An execute condition is defined for an activity.

Previous versions	Two steps are displayed in the list view of the Process Instance Manager (PIM): one for the decision and another for the activity.
--------------------------	--

Current version	Because the activity definition now holds all the necessary data, only one step is displayed in the list view of the Process Instance Manager (PIM) for the activity.
------------------------	---

End activity with a rollback

A business process has an End event with a rollback and the process instance activities are viewed in the Process Instance Manager.



Previous versions	The Rollback activity is displayed as the last activity.
Current version	The Rollback activity is displayed immediately before the Compensate activity.

Embedded sub-process activity type

The activities of a business process instance that has an embedded sub-process are viewed in the Process Instance Manager.

Previous versions	The embedded sub-process activity is not displayed with the activity type as Embedded Sub-Process. When debugging the process too, the embedded sub-process activity is not highlighted even when executed. If the embedded sub-process has an Execute condition, the activity name is displayed in the Process Instance Manager and its activity type is displayed as Embedded Sub-Process.
--------------------------	---

Current version	The embedded sub-process activity is displayed with the activity type as Embedded Sub-process. This is applicable even if the embedded sub-process has an Execute condition.
------------------------	--

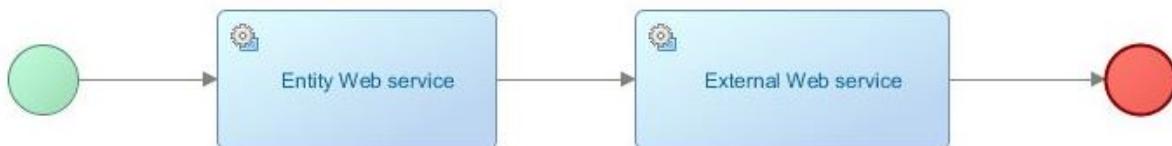
Transactional improvements in short-lived business processes

Short-lived business process execution is enhanced to make it transaction-safe and atomic. All the activities of a short-lived business process model that run in the same Application Server OS process participate in the same transaction and are either committed or rolled back entirely on failure.

After AppWorks Platform is installed, by default, the following Service Containers are assigned to the Application Server OS Process:

- Platform
- Notification
- Business Process Management

Consider a scenario where a short-lived business process model contains an Entity Web service and an external Web service. The Entity Web service is attached to the Platform Service Container and the external Web service is attached to the UDDI Service Container.



In this scenario, the service container that handles the Entity Web service and the Business Process Management service container run in the Application Server OS process. The service container that handles the external Web service does not run in the Application Server OS process.

In runtime, when the business process executes, the Entity Web service participates in the same transaction as the business process model whereas the external Web service is not a part of the same transaction.

In the event of a failure, the Entity Web service is rolled back along with the business process model whereas the external Web service is not rolled back.

Defining runtime security

By default, access is blocked for all the business process models created on AppWorks Platform 16.3.

To define runtime security on a business process model:

1. In Workspace Documents (Explorer), right-click a business process model and select **Define Runtime Security**.

The Security Editor window opens, displaying the name of the business process model on its title bar.

By default, the **Apply to all subprocesses** check box is selected.

2. In the **Roles** pane, click and select a role to grant permissions on the role for that business process model.
3. In the **Permissions** pane, select the permissions for each role.

Permission to...	Operations	Provides permission to...
Initiate instance	Initiate	Execute the business process model.
Manage instances	Read	Read operations on the process instance.
	Update	Update operations on the process instance.
	Delete	Delete operations on the process instance.
Communicate with instances	Send message	Send a message from the current process instance.
Administer instances	Suspend	Suspend instances on the process instance.
	Resume	Resume instances on the process instance.
	Terminate	Terminate instances on the process instance.
	Restart	Restart instances on the process instance.

In addition, the **Apply to all subprocesses** check box applies the parent process authorization on all the sub processes, when triggered by the parent process, which is set by default.

Note

- By default, all the permissions are enabled for the AppWorks Platform Administrator, Developer, and Deployer roles.
- The deployed process model authorization functionality will be deprecated for the newly designed models or re-published models on the migrated environment.

- Process authorization is a security add-on for the new process models and not applicable for the models designed prior to AppWorks Platform 16.0 (BPML based models).
- Process authorization is not applicable, if a process is part of a case model or a lifecycle.

Chapter 4

Working with business activity monitoring

Business Activity Monitoring (BAM) provides an end-to-end, integrated solution for closed-loop monitoring, business event response management, and improvement of business processes. It empowers you with the global visibility of the business processes and the increased responsiveness to them, enabling you to take proactive action based on the actual information. It helps you monitor your company's business processes, identify failures or exceptions, and address them real-time.

BAM provides real-time alerts and notifications on critical events and exceptions, and a centralized performance dashboard. It offers a drill-down analysis to discover trends, patterns, and bottlenecks in enterprise performance. In addition, BAM also supports monitoring of non-process data (Enterprise Data Objects) that leverages the same functionality as available for business processes. It provides real-time visibility into the effectiveness of overall business performance by monitoring key aspects and alerting users about deviations. This helps the firms to react with a level of agility that was previously unavailable through conventional approaches to business monitoring.

The near real-time monitoring process constitutes of a single step called Defining Business Event Responses. Moreover, the periodic monitoring process involves a series of steps such as creating the Process Monitoring Objects, building Business Measures, and KPIs and depicting the monitored results through the Dashboards. The entire process is accomplished through the BAM artifacts.

BAM has the following task-based components.

Business Event Response Composer	The Business Event Response Composer helps defining the Business Event Responses and facilitates monitoring of the business critical events, occurring in that particular Business Process. A Business Event Response keeps track of the multiple events and compares those event parameters and outcomes against a set of business rules. It also takes the appropriate action if the event falls within the predefined criteria.
Process Monitoring Object Modeler	The Process Monitoring Object is the fundamental building block for analyzing the health of the Business Processes. The process

	analysis and monitoring occur for the aggregated process or activity instances on the key interface. The Process Monitoring Object comprises the Business Attributes, taken from a Process and the Attributes of an external Web service.
Business Measure Editor	The Business Measure helps measuring an aspect of a business process or any external data. The Business Measure provides a view on the selected attributes, contained in a Process Monitoring Object or the external data source.
KPI Editor	The KPI Editor helps defining the expression for the KPIs, using the Business Measure already defined. The results monitored through the KPI help in monitoring and capturing the information related to the business process, and in triggering business actions, as specified. For each KPI, the KPI Editor provides the ability to specify conditions warranting business action and outbound events that represent notifications of such conditions and that can trigger business actions such as E-mail, calling Web service and triggering Business Process.
Dashboard Designer	The Dashboard Designer allows you to create the dashboards using the already available Business Measures, KPIs, and the Standard Views, which provide a consolidated view on the Process performance.

To use BAM:

- You must configure the BAM service container and load the BAM MDM Model application package. See [Using Business Activity Monitoring](#).

BAM is installed in System organization, by default.

- If you want to work with BAM in other than the installed organization, you have to [Create a Service Group for BAM](#).

To reconfigure the database for BAM:

1. In the organization where BAM is installed, [Configure a New Database for BAM](#).
2. Repeat the procedure described in [Configure a New Database for BAM](#), if you want to work with BAM in other than the installed organization, which needs a different database configuration.

BAM is not supported on the PostgreSQL database.

Configuring a new database for BAM

If you have selected **Use same inputs for a machine** option, the default option while installing AppWorks Platform1 applications, then BAM is installed and configured to the Cordys system database configuration.

The database entries of BAM and MDM are also created in the Cordys system database configuration. After successful AppWorks Platform1 installation, assume that you have loaded the AppWorks Platform1 BAM MDM Model application successfully, and continue working with the Synchronization and Upload functionality.

There is heavy load on the Cordys system database configuration, because you have huge amount of data in the BPM and BAM related tables. To reduce the burden on the database, you can configure a new Database to segregate BAM and MDM related database content from the Cordys system.

1. Uninstall the **AppWorks Platform1 BAM MDM Model** application.

See Undeploying Applications using Application Deployer in the *AppWorks Platform1 Administrator's Guide* for more information on uninstalling an application.

The AppWorks Platform1 BAM MDM Model application is uninstalled.

2. The table BAM_FILTERDPROCESS that stores the process information selected for synchronization in BAM administration, is dropped. To retain the process selection, backup the table data.
3. Drop the BAM and MDM related tables from Cordys system Database Configuration after taking the data backup from BAM related tables (if needed) and run the respective scripts in the following order.

Component Application	Path of the script file	Script
Cordys Business Activity Monitoring	<AppWorks Platform_installdir>\components\bam\database\dropscripts	BAM_<DB VENDOR>_DROP.sql
Cordys MDM (Drop MDM related tables if no other model is configured to the Database)	<AppWorks Platform_installdir>\components\mdm\dbscripts\repository	MDM_<DB VENDOR>_DROP.sql

The BAM and MDM related tables are dropped from Cordys system database configuration when the respective scripts are executed.

4. Create a new Database Configuration, for example, BAM Repository. The new Database Configuration, BAM Repository is created.
5. Create the BAM and MDM related tables in the BAM Repository Database Configuration and run the respective scripts.

Component Application	Path of the script file	Script
Cordys Business Activity Monitoring	<AppWorks Platform_installdir>\components\bam\database\createscripts	BAM_<DB VENDOR>.sql
		BAM_<DB VENDOR>_DATA.sql
		BAM_<DB VENDOR>_TRIGGERS.sql

Component Application	Path of the script file	Script
MDM	<AppWorks Platform_installdir>\components\mdm\dbscripts\repository	MDM_<DB VENDOR>_TABLES.sql
		MDM_< DB VENDOR >_TRIGGERS.sql
		MDM_INIT.sql

The BAM and MDM related tables are created in the BAM Repository Database Configuration when the respective scripts are executed.

6. Copy the CAP_SCHEMA_REVISION table along with the data from the Cordys System database to the BAM Repository database. For information on copying the table, see the database vendor documentation.
7. Replace the Cordys system Database Configuration with the **BAM Repository** Database Configuration in the following Service Containers:
 - Business Activity Monitor
 - MDM Hub Publisher
 - **MDM Spoke Publisher > Repository tab**
 Check the "Use different database for application" option to enable the Database Configuration for Application tab in MDM Spoke Publisher. The Database Configuration in **Application** tab should point to the Business Process Management (Application) Database Configuration (Cordys system).
 The Cordys system Database Configuration is replaced with the BAM Repository Database Configuration in the Business Activity Monitor, MDM Hub Publisher, and MDM Spoke Publisher Service Containers
8. Install AppWorks Platform1 BAM MDM Model application:
 - a. On the Welcome page, click  (Application Deployer) to open the Application Deployer wizard.
 - b. Click **New**.
 All the available applications that are not deployed are listed in the Applications List.
 - c. Right-click **AppWorks Platform1 BAM MDM Model** application and click **Deploy** to install the application.
 A page appears listing the selected application along with its dependencies and their deployed status.
 - d. Click **Next**.
 The Impact Analysis screen will be displayed.
 - e. Click **Next**, select **MDM Model** tab and do the following:
 - i. In the **Deployment details** tab, leave all options as is with default values.
 - ii. Click **Service Groups** tab to configure the MDM Hub publisher and MDM Spoke publisher service groups.

- Click the zoom button next to MDM Hub Publisher. Select **MDM Hub publisher** that already exists and click **OK**.
 - Click the zoom button next to MDM Spoke Publisher. Select **MDM Spoke publisher** that already exists and click **OK**.
- f. Click the **Database Schema** tab and select the database configuration **Select Database Configuration**.
The selected database configuration must be same as the one configured in the Application tab of MDM Spoke publisher service container (Cordys system).
- g. Click **Next**.
- h. Click the **Deploy** button to proceed with the installation.

Note:

- Restore the backups taken earlier, if any.
- In a primary-distributed setup, it is adequate to deploy the `BAM MDM Model` application package only in the primary server.
- You must have the BAM service container in the organization in order to work with BAM.
- All the BAM service containers in a cluster or a AppWorks Platform1 instance must always point to the same repository.

The new database is configured.

Creating a tablespace

To create a tablespace:

1. Select **Oracle Thin/OCI or PostgreSQL** as your Database vendor in **JDBC Driver** list.
 2. The **Create New User** (if Oracle Thin/OCI is selected) or **Create New Database** (if PostgreSQL is selected) option appears at the bottom of the page.
 3. Select **Create New User** (if Oracle Thin/OCI is selected) or **Create New Database** (if PostgreSQL is selected) option.
 4. Provide the relevant DBA user credentials.
 5. Select the **Create New Tablespace** option, if you want to create a new tablespace. Corresponding fields appear beside this option.
 6. If you do not select the **Create New Tablespace** option, you must enter an existing tablespace name.
 7. Provide the appropriate details in the **Tablespace** and **Tablespace Path** fields.
 8. Click .
- A new tablespace is created.

Creating a service group for BAM

You must create a service group for BAM to use BAM features in organizations other than the default 'system' organization.

To create a service group for BAM:

1. On the My Applications App Palette, click  (System Resource Manager).
The System Resource Manager window opens.
2. Click  on the tool bar.
The Service Groups App Palette appears.
3. Click  in the Service Groups App Palette.
Alternatively, click on the toolbar in the System Resource Manager window instead of accessing the Service Groups App Palette. The New Service Group wizard appears.
4. Select **BAM Connector** under **Available Application Connectors** and click **Next**.
See [Application Connectors](#) for more information on the application connectors.
The Provide Service Group Details page appears.
5. Provide the name of the service group and select the check box corresponding to **Web Service Interfaces** to select all the method sets in the list.
6. Click **Next**.
See Service Group Configuration Interface for information on the Provide Service Group Details page.
The Provide Service Details page appears.
7. Provide the required details and click **Next**.
See [Service Container Configuration Interface](#) for more information.
The Provide Details for BAM Connector page appears.
8. Click the **BAM Repository** tab and do one of the following:
 - Select the database configuration that you have selected on the **Repository** tab of **MDM Hub Service Container** during AppWorks Platform installation.
 - Select the database configuration that you have selected on the **Repository** tab of **MDM Hub Service Container** if it is available in the current organization where you have to create a service group for BAM.
 - If the database configuration already exists, in the Select Database Configuration list, select the same database configuration that you have selected on the Repository tab of the MDM Hub Service Container during AppWorks Platform installation.
 - If the database configuration is not present, and if the database exists, create a database configuration.
 - If the database configuration is not present, and if the database does not exist, [configure a new database for BAM](#).

All the fields on the **BAM Repository** tab are filled in with details.

- Every BAM service container has five MaxReadConnections by default. In heavy load scenarios, it is recommended to increase MaxReadConnections to ten for faster event processing. This option is available under **Advanced properties** on the **BAM Repository** tab of the service container properties. For example, in a workspace, there may be ten process models where each process model has at least one Process Monitoring Object (PMO) and one Business Event Response (BER). In this case, processing of the events can be slower. Faster processing can be achieved by increasing MaxReadConnections.

9. On the **Monitoring Engine** tab, do the following:

- a. In the **Threads Configuration** group:
 - i. Provide the number of threads for processing the events in the **Event Processing Threads** field. These threads are used to process the events generated when a BPM is instantiated. By default, 5 threads are used in event processing.
 - ii. Provide the number of threads for processing business objects that are either **Process Monitoring Object** or **Business Event Response** in the **Business Object Processing Threads** field. By default, 15 threads are used in the business objects processing.
 - b. In the **Profile Configuration** group:
 - i. Provide the email address in the **E-Mail Address** field so that the email actions from KPI and Business Event Response of BAM are sent from this particular email address.
 - ii. Provide the Display Name. This will be used as the display name when the emails are sent to the users.
 - iii. Click  corresponding to the Select proxy user for runtime execution field. The Organization Users dialog box opens.
 - iv. Select the relevant user and click **OK**.
The contextual information such as the Web service operation in both the Process Monitoring Object and Business Event Response will be executed in the context of this user. Therefore, ensure that this user has the required access, by assigning roles from the required application, to execute the Web service operation defined in the contextual information of PMO or BER.
 - v. Provide the number of business objects to be cached in the **Business Object Cache Size** field.
By default, 50 objects are cached.
10. Click **Next**.
The Provide Connection Point details page appears.
11. Provide the required details and click **Finish**.
See [Connection Point Configuration Interface](#) for information on configuring the connection point.

12. Ensure that the service container for BAM is created successfully and is started. The service group for BAM is created.
13. If you require data of the previous instances, perform the upload operation as mentioned in Business Activity Monitoring Administration in the *AppWorks Platform Administrator's Guide*.

Using features of BAM

BAM develops the following interfaces:

- Dashboards - help business users doing business activity monitoring and performing drill-down to perform root cause analysis.
- Business Event Response - involve or intimate business users when a particular set of conditions are satisfied in a business process instance.
- KPI - involve or intimate business users when a performance measure crosses a particular threshold.

A BAM solution comprises the following constituents:

- Dashboards
- Business Event Responses
- KPIs
- [Process Monitoring Object](#)
- Master Data Management (MDM) Entities and Enterprise Data Objects

Defining business event responses

The Business Event Response Composer facilitates the creation of Business Event Responses. You can perform real-time monitoring based on the occurrence of selected process and activity events in relation to the specified business condition and take corrective action if the event falls within the specified business condition. For example, in an OrderToCash process, the Line of Business (LOB) manager expects a notification if an order from a GOLD customer is not shipped within 2 days. Similarly, he might also be interested in getting a notification if a customer with bad credit history has placed an order of value greater than \$1000.

You need roles from each component to execute the Business Event Responses. The roles should be added to the proxy user (specified during AppWorks Platform Business Activity Monitoring application installation). For example, if the action is Send Notification, everyoneInCordys Notification role should be attached.

The Business Event Composer facilitates the creation of Business Events. The high-level tasks that you can perform using Business Event Response Composer are:

- [Creating a Business Event Response](#)
- [Publishing a Business Event Response](#)
- [Unpublishing a Business Event Response](#)
- [Editing a Business Event Response](#)
- [Deleting a Business Event Response](#)

Creating a business event response

Before you begin:

1. Create and design a business process model.
2. If you are creating a Business Event Response with the Email template, in an organization other than the System organization, you need to create the Notification Service Container in that organization.

Business Event Response keeps track of the multiple events and compares those event parameters and outcomes against a set of business rules, and takes the appropriate action if the event falls within the predefined criteria.

To create a business event response:

1. [Select a starting point](#) and select  (Business Event Response) to create a Business Event Response.
The Untitled Business Event Response - Business Event Response appears.
2. Enter **Name** and **Description** of the Business Event Response.
3. [Select a Business Process](#).
If you select the business process as a runtime reference, then you cannot edit or delete the message filter. If the business process does not have message filter, then you can create Business Event Response on Standard attributes only.
In a Business Process, in any input XML element, specify the date parameter value in YYYY-MM-DDThh:mm:ss:S format.
4. Select the required Process and related Activity Events from the Events section, and click **Next** to view [Select Attributes](#) page.
The Select Attributes page appears with the selected Activity Events, and the Process Events, by default, on the left pane under Events section.
Upon the selection of a Process or Activity Event under Events section, the relevant Process or Activity Attributes are displayed on Attributes Selection pane.
5. Select the Process or Activity attributes.
6. Select **Include contextual information** check box.
7. Click **Next** to choose a Web service to get the [Contextual Information](#).
 - Including the Contextual Information is only an optional step.
 - By default, the Process Attributes appear on the Select Attributes page.

8. Fill the required information in the relevant columns on the Select Contextual Information page.
You can also check Model Email option, in case you want to model an email template and send an email when a specified condition for the Business Event Response defined is met.
9. Click **Next** to view the [summary](#) of the selected attributes.
10. If you have selected Model Email, click **Next** to model an email template, which helps sending the attributes information of the process in an email, to the concerned user, in case the specified business condition for the Business Event Response is met.
If you are creating a Business Event Response with the Email template, in an organization other than the System organization, you need to create the Notification Processor in that organization. This is a prerequisite for creating Business Event Response in other than the System organization.
11. Click **Next** to frame rules by defining condition and action.
If the Business Event Response condition in a Business Event Response is built on a multi-valued attribute and for any process instance, the Business Event Response is resulted in multi-valued match, and then the action is fired only once indicating that the Business Event Response condition is met and triggers the appropriate action only once.
12. Complete the process of defining condition and action and click Finish to complete the Business Event Response creation process.

The Business Event Response is created and added to the content tree.

You cannot rename a Business Event Response after it is created.

After you complete this task:

- Publish the Business Event Response to the organization.

Selecting business process events

Select the Business Process as a source to define a Business Event to perform real-time monitoring. Select the Activity and Business Process events from the selected Business Process that are crucial for business and which, you would like to monitor, only after selecting the Business Process.

To select business process events:

1. Click  to select a business process.
The Select Process dialog box opens, displaying all the Business Processes.
2. Select a process and click **OK**.
 - If you select a Business Process from a run-time reference for which, there is no Message Filter configured on it, then a message, "Message Filter not defined on the selected process. Only standard attributes will be available" appears.
 - If you select a Business Process for which there is no [Message Filter](#) configured on it, then a message, "Message Filter not defined on the selected process. Only standard

attributes will be available. Message filter(s) can be created in the Business Process Model editor" appears.

- If you select a Business Process for which **Message Filter** is **configured**, then a message, "Message Filter(s) defined on the selected process. They can be edited in the Business Process Model editor" appears.
- While creating the business process model for an activity, if you clear the Configure Monitoring on Process Level check box, then you cannot monitor that activity from BAM. The Message Filter concept is applicable only on publish of the Message Filter. The Message Filter is published only if the Message Filter is used in a Business Event Response and if that Business Event Response is published. If the same Message Filter is used in multiple Business Event Responses, then at least one Business Event Response must be published to apply the message filter.
- If you have selected a Business Process while creating a Business Event Response, and after selecting the Business Process Model, if you have modified the Message Filter in the Business Process Model, the changes made in the Message Filter will not reflect until you select a Business Process once again. If the Message Filter is modified or updated in the Business Process Model, then at least one Business Event Response, where this message filter is used, must be published to get the modifications.

The Business Process name appears in Business Process and all the Activity Events for the selected Business Process appear under Events. For example, Activity Start and Activity End.

This is required to get attributes of the process available at that particular activity state. Attributes are needed to frame the business condition and activity states such as activity start and activity end is needed for triggering point of an action associated with the business condition. For example, in an Order-to-Cash process, there are three activities namely Register Order, Get Credit Rating, and Approval by Manager. Attributes like ProductID, Quantity, and so on are available at the Activity start of the Register Order activity. At that point, attributes such as Credit Rating are not available and are available only at the Activity End of the Get Credit Rating activity. Using the Quantity attribute (available at RegisterOrder_Start event), a business condition can be framed such as Quantity > 200 and an action Send Notification or Email can be associated to it. Therefore, if an order request comes for Quantity > 200 (at RegisterOrder_Start event), then the associated action Send Notification or Email is triggered. If ProductID is not selected while message filtering in process, then it will not be available for selection at Register Order Activity Start.

Monitoring of the activity events is not possible for short-lived processes.

3. Select the **Show Business Process Readonly View** check box to view the selected Business Process in the read-only mode. The selected Business Process Model is displayed in the read-only mode on the right pane. This property is not stored when the document is stored. Users need to select this property each time they open the document to view the process model.

4. Select the required business event or events.
5. Click **Next** to select the process or activity attributes, related to the Process or Activity Events, as the second step in the creation of a Business Event. The Select Attributes page appears with the selected Activity Events, and the Process Events, by default, on the left pane under the Events section.

On selection of a Process or Activity Event under the Events section, the relevant Process or Activity Attributes are displayed on the Attributes Selection pane.

The Business Process events are selected.

Selecting business process event attributes

You have to select the process or activity attributes of the business process that are crucial for business and which you would like to monitor. You can perform real-time monitoring based on the occurrence of the selected business process and activity events in relation to the specified conditions built on the selected attributes and can take corrective action in case the events fall within the specified business condition.

To select business process event attributes:

1. Click the required **Process or Activity events *under* Events** section to select the required Process or Activity attributes.
If you want to set the lead time of an activity, see [Setting the Lead Time for an activity](#). The attributes, along with their alias names and XPath, for the selected Process Event or Activity appear In the right pane. Standard attributes Process Name, Instance ID and Activity Name appear under the Process and Activity start column respectively. Standard attribute - Lead-time also appears under the Process and Activity start. If you want to select the lead time attribute for an activity, see [Setting the Lead Time for an activity](#).
 - If you selected the **Activity Event**, then the attributes that you selected for message filtering appear along with the Standard attributes.
 - For usage of Standard Attributes, see [Usage of Standard Attributes in creating a Business Event Response](#).
 - By default, the alias name would be the same as the name of the Attribute. Change the name of the alias, Alias Name, if required.
2. Select the required attributes of the Process specific messages from the Process Variable Selection table.
3. Select **Include contextual information** to get the contextual data from a web service, if required.
4. Optional. Click **Next** to select a Web service to get the **Contextual information**, as the third step in the creation of a Business Event Response.

For example, in the Order-to-Cash process, at Activity start of 'Register Order' you have an attribute called ProductID. You want to increase your monitoring context by including some more details about the Product. The attribute you want to add is 'ProductName' and 'UnitsInStock', which is not there as an attribute for that activity but present in

existing database (different table) or in some external database then it can be called as an attribute using Include contextual information option.

- You may note that these attributes have no role to play in process execution and you can utilize them to increase the monitoring context.

5. Optional. Select **Model Email** to define an email template containing attribute information of the process.
The template can be triggered as an action If the specified business condition is met, on occurrence of an event such as process or activity start.
6. If you do not opt for including contextual information or Model Email, click **Next** to view the Summary page.
The Summary page appears displaying the Attribute Source (Process or Activity), Attribute Alias Name and the Data type fields.
7. From the Summary page click **Next** to proceed with rule creation by defining conditions and respective action.

The business process event attributes are selected.

Summary of selected attributes

The following table describes the fields on the summary page of the selected attributes.

Attribute Source	Displays the selected activity on the activities page.
Alias Name	Displays the alias name of the activity assigned on the Attributes page.
Datatype	Displays the data type.

Setting lead time for an activity

For the Start event of an activity under the Events section:

1. Select the Lead-time attribute under the Attribute Selection section.
The Confirm dialog box opens.
2. Select any of the following:
 - Ideal Lead Time
 - Expected Lead Time
 - Minimum Lead Time
 - Maximum Lead Time

Based on the selection of the category, their respective values get filled in the Define Conditions and Actions Interface to build the condition expression. For more details on Estimated Lead Time refer to Fields on the Estimated Duration tab Table of [Activity Properties Interface](#).

The Confirm dialog box opens only if any of the fields in the **Estimated Lead Time** of the **Estimated Duration** tab in the properties window of the activity of type 'Task' is filled.

Usage of standard attributes in creating a business event response

The following table describes the usage of the standard attributes in creating a Business Event Response.

Instance ID	Use the attributes in the e-mail template if you want to trigger an email if the specified business condition is met. On occurrence of an event, you can provide a link to the Instance ID that takes you to the instance level detail of the process wherein you can see the activity level view of that particular process instance.
Lead-Time	For Order-to-Cash process, at the start of 'Approval by Manager' activity, define a condition using the lead-time attribute - if the activity is not completed within 2 days then trigger any action - Send Email, Send Notification, trigger BPM or Call web service.
Process name or Activity Name	You can use these attributes in the e-mail template, if you want to trigger an email if specified business condition is met, on occurrence of an event like activity start/end and so on.

Message filtering

Message Filter is about filtering the message map of a process to enable the monitoring only on the filtered attributes that are crucial to the business. Users can select the specific attributes from the Business Process model that needs to be monitored. This feature has the following advantages:

- Monitoring only the required and specific attributes
- Restricting unwanted data population from the Business Process layer to BAM layer thus contributing to the database size reduction
- Enabling the faster processing in the BAM engine with the help of filtered attributes (limited set)

Example

A Business Process has more than 200 business attributes. However, BAM needs only 45 attributes. So, as a business user, you can select only the 45 required attributes with the help of Message Filter. This results in database size reduction and faster monitoring process.

Selecting contextual information

You have to select contextual information to correlate the process data with external attributes from a Web service for increasing the context of the monitoring.

To select contextual information:

1. Click the required **Process** or **Activity Event** on the left pane.
Options to select the Web service for including the Contextual information appear.

2. Click  to select the required Web service.
The Select Web Service dialog box opens with a list of available Web services.
3. Select a relevant Web service and click **OK**.
The key field for the selected Web service appears under the Input Parameters column.
4. Highlight the parameter under the **Input Parameters** column and click the text field corresponding to the parameter under the Parameter Mapping column to view the list.
The parameter list appears with the attributes list under the Parameter Mapping column.
5. Select the required attribute you want to map.
6. Select the required parameter under the **Output Parameters** column.
7. Click the highlighted text field and rename the alias name, under the **Alias Name** column, if required.
8. Click **Save** to save the information entered.
9. Click **Reset** to clear or change the information already entered.
10. Click **Next** to view the Summary page.
The Summary page appears displaying the Attribute Source (Process or Activity), Attribute Alias Name, and the Data type fields.
11. Click **Finish** to complete the creation of Process Monitoring Object on the [Summary](#) page.

The Process Monitoring Object is created and added to the project content tree.

- The newly created Process Monitoring Object can be consumed in the Business Measure Editor to define a Business Measure. Using the set of attributes of a Process Monitoring Object in a Measure, an aspect of process can be measured by building aggregate queries on the attributes.
- If you choose an Intermediate Message (IM) immediately after choosing a Web service activity, do not select Include contextual information option because if you choose an Intermediate Message immediately after choosing a Web service activity, the response of the Web service is available immediately. The IM is completed immediately. If any contextual information is called in the IM start event, the IM start event is not fetched because the Business Process Management engine does not fire any start event of the IM.

If you need to select any external Web service attributes to fetch the contextual information:

- Ensure that required roles to invoke that Web service are assigned to the Proxy user in the working organization.
For example, if a user is invoking the Web service of Northwind Method set as an external Web service or it is an invocation from Business Event Response or KPI, which is of the Northwind application, ensure that `everyoneInAppWorks Platform Northwind Demo 1.0` role is assigned to the Proxy user.

Modeling an email for a business event response

The objective of the Model Email is to send the process and related contextual information to the user, when a specified condition based on a Business Event Response is met.

In an Email, you can include the information relevant to the business process, user, and the required response of external Web services, and so on. You can customize the contents of the email template as per the requirement in terms of the background color, font size, formatting and so on as per the requirement.

The components of the Email can be modeled by changing the HTML code. The Attributes related to the Business Event Response such as Process attributes and Contextual attributes would be available in the Message area under the Message data tree of the Email template. These attributes can be used in any of the above components by dropping them in the specific area.

To model an email for a business event response:

1. In the Email model page, under the Email Model of the Model Components section, provide the necessary message, which has to be sent as part of the email.
The Model has Header, Salutation, Body, Signature, Footer, and Application Details components.
The selected attributes in the earlier stages are available under the Message section of the Email template. These attributes can be used in any of Model components by dropping them in the specific area.
2. Enter the Recipient's name and Email id in the Recipient's Name and Email Id fields.
The Email template, defined by you, would be made available as a Web service. It can be triggered as an action, in the form of invoking Web service.
The created Email template is available as a Web service in the Custom Space, which is available in the current user organization with the name of the Business Event Response. For more information on the modeling the email template, see [Creating an E-mail Model](#).
3. Select any one of the Web service Definition Set option under Select Web service Definition Set section, as the Email template defined by you would be available as Web service.

New	<p>Select New if you want to provide a new Web service Definition Set to the Business Event Response. This option is selected by default. The Web service Definition Set Name and Web service Definition Set Namespace fields are enabled; enter a name and namespace in the respective fields. By default, it is filled by the Business Event Response name.</p> <p>The Web service Definition Set Name, Web service Definition Set Namespace and Webservice Interface are mandatory fields and should not begin with a number. Blank spaces or special characters are not allowed.</p>
Existing	<p>Select Existing if you want to apply any existing Web service Definition Set. The Existing Web service Definition Set Name field is enabled with .</p>

	The Existing Web service Definition Set Name is a mandatory field and it should not begin with a number. Blank spaces or special characters are not allowed. Click  , and select the Web service definition set name from the Select Web service Definition Set dialog. The Existing Web service Definition Set Namespace field remains disabled. Depending on the Web service Definition Set you select, this field is populated with Namespace automatically. Select a Webservice Interface from Select Web service Interface list.
BAM Default	Select BAM Default option to select the default Web service Definition Set. The Webservice Definition Set Namespace is filled automatically.

The Email Web service can be used as an action, in the form of invoking Web service.

After you complete this task:

- Click **Next** to define actions on the build rules (conditions) and define actions page.

Define conditions and actions

Rules dictate the manner in which a business object reacts to a situation, and eventually charts the course of a business process. You can model the behavior of data or business objects using rules.

To define conditions and actions:

- Set properties to the conditions and define actions as part of building the rules.
- After the rules are created, the Business Event Response is created and listed under workspace documents or project content tree.
- Create rule actions on top of rules that are performed when rule conditions are met.

Restriction

The steps and processes of building rules are similar to that of the Decision Table, with some exceptions. The Define Conditions and Actions page differs from the Decision Table.

In Decision Table	In BAM
A notification appears to select the attributes.	The selected attributes are made available on the left pane of the Conditions and Actions page.
By default, the Properties tab is visible.	The functionality of both the Properties and Namespaces tabs are handled by BAM, they are not visible to you. However, by default, the Behavior tab is visible.
In the Properties pane, the Condition template option of the field Value Type is supported.	On the Properties pane, the Condition template option of the field Value Type is supported for the temporal conditions, in which the condition is modeled using the LEAD_TIME attribute. Refer the field Value Type under

In Decision Table	In BAM
	<p>the section Setting Properties of Conditions in a Decision Table from the AppWorks Platform documentation on how to use the condition template. To use Condition Templates, you have to add them by the following reference, Runtime references. For example: Right-click the project or the folder > Add Runtime References > BAM Condition Templates > Lead Time Greater than and other possible options. This is provided for user comfort, user can select any of the templates for defining temporal condition. For more information on Conditional Template, see Conditional Template. Fill in the following Parameters:</p> <ul style="list-style-type: none"> ■ Mins ■ Hours ■ Days
The Namespaces is enabled.	The Namespaces tab is not applicable.
The right-click (of the mouse) functions for the rule header are supported.	Does not support.
You can create multiple rules.	You can create only one rule.
You have to create the Data template.	The Data template is automatically created.
	<p>You can use only the following actions in BAM:</p> <ul style="list-style-type: none"> ■ Trigger Business Process ■ Invoke Web Service ■ Format Content if you select the Model Email option. ■ Send Notification
	<p>Limitation: You cannot build conditions with Or option as in Decision Table, and only And option is supported.</p>

Tip: For more information on building rules, see building rules of [Building a Decision Table](#).

4. Click **Finish** to create the Business Event Response.

The created Business Event Response appears under Project content tree or Workspace document.

Conditional template types

Conditional Template contains a set of predefined conditions that would be used in the creation of a Decision Table.

Six types of Conditional Templates are available on the Define Conditions and Actions page.

Type	Parameters	Description
LeadTimeGreaterthan	The parameter you select (Mins or Hours or Days) is calculated on the greater than (>) condition.	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is greater than 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_EVENT/process_attributes/PROCESS_LEAD_TIME < (2 * (24*60)) + (2 * 60) + (10)
LeadTimeLessthan	The parameter you select (Mins or Hours or Days) is calculated on the less than (<) condition.	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is less than 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_EVENT/process_attributes/PROCESS_LEAD_TIME < (2 * (24*60)) + (2 * 60) + (10)
LeadTimeEqualto	The parameter you select (Mins or Hours or Days) is calculated on the equal to (=) condition.	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is equal to 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_EVENT/process_attributes/PROCESS_LEAD_TIME = (2 * (24*60)) + (2 * 60) + (10)
LeadTimeNotEqualto	The parameter you select (Mins or Hours or Days) is calculated on the not equal to (!=) condition.	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is not equal to 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_EVENT/process_attributes/PROCESS_LEAD_TIME != (2 * (24*60)) + (2 * 60) + (10)
LeadTimeLessthanEqualto	The parameter you select (Mins or Hours or Days) is calculated on the less than equal to (<=) condition.	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is less than equal to 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_EVENT/process_attributes/PROCESS_LEAD_TIME <= (2 * (24*60)) + (2 * 60) + (10)
LeadTimeGreaterthanEqualto	The parameter you select (Mins or Hours or Days) is calculated on the greater than equal to (>=) condition	If you select 10 (mins), 2 (Hours) and 2 (Days) as the parameters, the condition is applicable if the lead time is greater than equal to 2 days, 2 hours and 10 minutes. The Condition Expression is: BUSINESS_

Type	Parameters	Description
		EVENT/process_attributes/PROCESS_LEAD_TIME >= (2 * (24*60)) + (2 * 60) + (10)

Formatting content

On clicking Invoke Webservice in the Actions section, the related input XML is displayed in the Request area of the Properties tab.

To format content on the Properties tab page:

1. In the Conditions and Actions page of the Business Event Response, click  to open the window.
The Select Attributes window opens.
2. Select the required attributes by using drag and drop at the required place in the input XML.
 - Find the attributes under <BUSINESS_EVENT> <process_attributes/context_attributes>elements.
 - Delete the value PARAMETER before dropping any attributes.
3. Fill the related elements in the entire input XML by removing the value PARAMETER.
 - If there are no attachments specified, remove <MAIL_ATTACHMENTS> <attachment name=""encoded="false">PARAMETER</attachment></MAIL_ATTACHMENTS>" from the input XML.
4. Remove the element structure if there are no inputs in the CC and BCC fields.

The formatting of the content is done.

Publishing a business event response

This topic describes the procedure to publish a Business Event Response.

To publish a business event response:

1. Before you begin, [create a Business Event Response](#).
Publishing the Business Event Response ensures to use it as an application at runtime.
2. [Access the Business Event Response](#).
3. Ensure that the Business Process Management and Rule Service Containers are started and are running.
4. Publish the Business Event Response to an [organization](#).
You can publish multiple Business Event Responses at a time by publishing the folder or project that contains the Business Event Responses.

The Business Event Response is published to organization.

Business event response runtime scenario

When the instance is created from a Business Process, the respective data is available in BAM through MDM. This data is evaluated by the BAM engine against the conditions, created in the Rule (Decision Table). When all the conditions are evaluated, an instance object is created in the BAM layer. Respective actions are triggered once the instance object is created.

Unpublishing a business event response

Before you begin:

- [Create and publish](#) a Business Event Response.

You must unpublish a Business Event Response to stop processing the events (monitoring). If you unpublish a Business Event Response, it becomes inactive, and the Business Event Responses are not evaluated for such Business Event Responses.

To unpublish a business event response:

1. In the Workspace Documents (Explorer) window, right-click the published (Business Event Response) and select Unpublish.
During unpublish, the Confirm dialog box opens with "Do you want to retain old data that talks about the business data collected until that time?" If you choose to delete or retain the business data, a message stating, "Business Event Response Unpublished successfully" appears as a notification. If you choose to retain the business data, the business data is retained.
If the Business Event Response is not published, Unpublish is triggered with a message stating, "Business Event Response is not published."
2. If you choose Yes or No, the Business Event Response Unpublished successfully appears as a notification.
 - If the Business Event Response is already unpublished, then the message "Business Event Response already Unpublished" appears.
 - If the selected Business Event Response fails to be unpublished, an exception message appears stating the error.

The selected Business Event Response is unpublished.

Editing a business event response

Before you begin:

- [Create](#) a Business Event Response.

You can edit a Business Event Response if you want to add or remove attributes, except for the name of the Business Event Response, and the Business Process.

To edit a business event response:

1. Access the [Business Event Response](#) and double-click it. Alternatively right-click the Business Event Response and select Open.
The Business Event Response is displayed in the Business Event Response composer.
2. Make the appropriate modifications to the Business Event Response and click **Save**.
While editing a Business Event Response, if the recipient and email Id are modified, they are not be updated in the action part of the decision table. You have to change them manually.
The modifications made to the Business Event Response are saved.
3. After you complete this task, [publish](#) the edited business event response to the organization to reflect the changes from the design time to runtime.

Deleting a business event response

Delete a Business Event Response if it is no longer required.

Consider the following when you delete a Business Event Response:

- If you delete a Business Event Response, the design time is also deleted from CWS.
- The delete operation deletes the artifacts, which are created during creation of the Business Event Response (Schema Fragment, Decision Case, Email Template).
- If deletion of the selected Business Event Response fails, an exception message is displayed stating the error.

(Business Event Response)

1. Perform one of the following:
 - a. In the Workspace Documents (Explorer) window, right-click **Business Event Response** and select **Delete**.
 - b. In the Workspace Documents (My Recent Documents) window, select **Business Event Response**, click  and select **Delete**.
- The Confirm dialog box opens.
2. Perform one of the following:
 - Click **Yes** to delete the Business Event Response. The Business Event Response is deleted from the project content tree.
 - Click **No** to cancel the operation.
 - Click **Show Used by** to retrieve the details of the documents that use the Business Event Response.

Modeling process monitoring objects

Process Monitoring Object is a set of associated attributes from a business process and the related contextual Web services, which need to be monitored as a logical group. Process Monitoring Object serves as a source for selecting attributes of a business process and can be used for analysis by creating Business Measures (query-based Web services). The created Business Measure would then be available for consumption in the KPI Editor for closed loop monitoring or on the Dashboard for visualization.

The Process Monitoring Object Modeler facilitates the creation of Process Monitoring Objects and thereby helps in monitoring the business process and its activities.

For Example, an Insurance claims process may have multiple stages as Pre-processing, Investigation, Processing, and Final settlement.

You can monitor the Investigation phase as a logical group so the attributes can be grouped as Process Monitoring Object and would be named `Insurance_Claim_Investigation`.

You can monitor certain attributes as part of the `Insurance_Claim_Investigation` object from external systems, which are not part of the Insurance claims process but logically make sense to perform analysis based on them.

For example, `CUSTOMER_CREDIT_LIMIT`. This attribute is external to the business process and to make it available as part of the Process Monitoring Object, an external Web service called `GetCreditDetails (Customer_ID)` can be included as contextual Web service. Basically, all the associated attributes can together make part of a Process Monitoring Object.

See [guidelines to develop Process Monitoring Object](#) for more information on guidelines for developing Process Monitoring Object.

The high-level tasks that you can perform using AppWorks Platform Process Monitoring Object modeler are:

1. [Creating a Process Monitoring Object](#)
2. [Publishing a Process Monitoring Object](#)
3. [Unpublishing a Process Monitoring Object](#)
4. [Editing a Process Monitoring Object](#)
5. [Deleting a Process Monitoring Object](#)

Guidelines for developing the Process Monitoring object

Process Monitoring Object is a set of associated attributes from business process and related contextual Web services, which need to be monitored as a logical group. Each Process Monitoring Object is persisted and can be used for the purpose of analysis by creating business measures (query based Web services) on it.

Select attributes in a Process Monitoring Object only when it makes sense to query on these attributes as logical set. Some guidelines to develop a Process Monitoring Object are as follows:

1. Different Process Monitoring Objects for different flow paths in a process.

You must define different Process Monitoring Objects for different flow paths in a process. There may be parallel or alternative flow paths in a business process. A Process Monitoring Object must not contain attributes from multiple alternative flow paths because all of them are never filled and are not queried. In general, define different Process Monitoring Objects for different flow paths.

2. Only use attributes that you can associate to business process instance context.

Business process modeling is a very generic environment and any Web service can be invoked as some intermediary step to retrieve information and then process to use in Business process. Attributes from a business process activity or a Web service output must be selected only when these attributes are retrieved in context of business process instance.

Example: When organization receives an order for a product, it would like to check if product is internally deprecated, so that it can monitor orders for deprecated products. There may be a generic Web service used in business process called 'getAllDeprecatedProducts()' and then there might be a logic in business process to check products ids which have come as part of Order. In this case, attributes from output of 'getAllDeprecatedProducts()' Web service must not be used as when queried along with other business process instance attributes like OrderID then the wrong result will be rendered.

This same logic of association with business process instance context is applicable when invoking external Web services to define attributes for Process Monitoring Object. In general, when you pass attributes from the business process instance context as input parameters of Web service, then they can be invoked in context of business process instance.

3. Do not select activities inside loop.

Business process message map is a global data structure and same XML structure is encountered then it is updated in message map structure and not appended. This means monitoring on activities inside loop activities do not yield correct result. OpenText advises against using such activities.

4. Only invoke read Web services from enterprise applications

Process Monitoring Object is not meant to manipulate in Enterprise applications by invoking update Web services. Though technically feasible, it is not advised to use it for that purpose and invoke Web services to retrieve information from enterprise applications and not to update information.

5. Use multi-valued attributes from two activities only when they return information in same order.

Multi-valued attributes are XML elements of which multiple tuple (rows) are returned in Web service response. These attributes can be used in Process Monitoring Object, if they are returned based on business process instance context. Selecting such multiple attributes from a single activity does not pose a problem, but when more than one activity in a Process Monitoring Object returns multiple valued attributed there can be improper results while querying when both activities does not return same number of rows and in same order.

6. Do not use multi-valued attributes Web services from Enterprise Applications.
This is a limitation that only the enterprise application Web services should be used that return single valued attributes.
7. Only child elements from Business process message map are supported to select as an attribute of a Process Monitoring Object. This is a limitation that only child elements from a Business process message map are supported to select as an attribute of a Process Monitoring Object.
Throughout the AppWorks Platform Product Documentation, the Process Monitoring Object attributes refer for either child elements from business process message map or Web services child elements.
8. Sub-processes are treated as activities and must be monitored as a separate Monitored Object. Each sub-process is treated as an activity. Therefore it needs to be monitored separately.
9. Build Activity level Process Monitoring Object, when monitoring has to done on Activity Start and End Events Process Monitoring Object, can be built on process events context or an activity events context. When there is a need to build a monitoring query such as: 'Select count of all logistics process instances in which shipment activity started last week and where customer_type= 'Gold' '. In this scenario, monitoring is done in context of the 'Shipment' activity Start event, so the Process Monitoring Object must be of type 'Activity Level' and built on Shipment activity. In the Activity level Process Monitoring Object attributes from the message map preceding that particular activity can be selected: 'Select count of all logistics process instances in which shipment activity ended last week and where customer_type='Gold' '.

While the Process Monitoring Object Editor does not restrain usage, you must follow the guidelines for building queries on the Process Monitoring Object that can return correct results.

Creating a process monitoring object

Before you begin:

- [Create and design](#) a business process model.

A Process Monitoring Object is the fundamental building block for analyzing the health of the Business Processes. The process analysis and monitoring takes places at the appropriate time for aggregated process or activity instances on the key interface. The Process

Monitoring Object comprises the Business Attributes taken from a Process and the Attributes of an external web service.

To create a process monitoring object:

1. Select a starting point and click  (Process Monitoring Object) to create a Process Monitoring Object. The Untitled Process Monitoring Object - Process Monitoring Object window opens.
2. Enter **Name** and **Description** of the Process Monitoring Object.
3. Select **Process** as first step in creating a Process Monitoring Object.
4. If you select the business process as a runtime reference, then you cannot edit or delete the message filter. If the business process does not have message filter, then you can create Process Monitoring Object on Standard attributes only.
5. In a Business Process, for any input XML element, specify the Date parameter value in YYYY-MM-DDThh:mm:ss:S format.
6. Select the required activity events under the **Events** section and click **Next** to view the [Select Attributes](#) page.
7. Select the Process or Activity attributes, select the **Include Contextual Information** option and click **Next** to choose a Web service to get the [Contextual information](#).
 - By default, Process Attributes are listed on the Select Attributes page.
 - Getting Contextual Information is an optional step.
 - The mappings to the database columns for the selected attributes are generated only if there is a Service Container for BAM with the attached Method Set BAM Monitor Web Service Interface. If the column maps are not generated for the attributes of a Process Monitoring Object, when this Process Monitoring Object is packaged and deployed on another machine, the generated column maps for the deployed Process Monitoring Object may not be the same as the original Process Monitoring Object attribute column maps.
 - For building a Process Monitoring Object, you can select the process or context attributes of each data type with the following specified limits:
 - String - 15
 - Numeric - 10
 - Decimal - 10
 - Date - 10
 - If the Service Container for BAM is up and running, then the specified data type limits are validated. There may be a business need, where you have to add more columns in a Process Monitoring Object. In those scenarios, add the necessary columns by altering the table, BAM_OBJECTS_DATA. Follow the column names pattern strictly as mentioned in <AppWorks Platform install directory>\components\bam\database\createscripts\BAM_<DBVendor>_DATA.sql. Perform this task only when you have to add extra columns. For MYSQL Database, you have to append the connection string withzeroDateTimeBehavior=convertToNull, if

the columns added to the BAM Repository for datetime datatype are specified with the default value.

- See [guidelines to develop Process Monitoring Object](#) for more information.
8. Complete the required information in the relevant columns on the Contextual Information page, and click **Next** to view Summary of the Selected Attributes.
 9. Click **Finish** on the Summary page to complete the creation process of Process Monitoring Object.

The Process Monitoring Object is created and added to the project content tree.

You cannot rename a Process Monitoring Object after it is created.

10. After you complete this task, you must [publish](#) the Process Monitoring Object to the organization.

Selecting a business process

You need to select the Business Process as a source to model a Process Monitoring Object, which is the fundamental block for analyzing the health of the Business Process. You need to select the Activity and Business Process events from the selected Business Process only after selecting the Business Process.

To select a business process:

1. Click  to select a Business Process.
The Select Process dialog box opens, displaying all the Business Processes.
2. Select a process and click **OK**.
 - If you select a Business Process from a run-time reference for which there is no [Message Filter](#) configured on it, then a message, "Message Filter not defined on the selected process. Only standard attributes will be available" appears. In a Business Process, in any input XML element, specify the `date` parameter value in the YYYY-MM-DDThh:mm:ss:S format.
 - If you select a Business Process for which there is no [Message Filter](#) configured on it, then a message, "Message Filter not defined on the selected process. Only standard attributes will be available. Message filter(s) can be created in the BPM editor" appears.
 - If you select a Business Process for which [Message Filter](#) is configured, then a message, "Message Filter(s) defined on the selected process. They can be edited in the BPM editor" appears.
 - While creating the business process model for an activity, if you clear the [Configure Monitoring on Process Level](#) check box, then you cannot monitor that activity from BAM.

The Message Filter concept is applicable only when you publish the Message Filter. The Message Filter is published only if the Message Filter is used in a Process Monitoring Object and if that Process Monitoring Object is published. If the same Message Filter is

used in multiple Process Monitoring Objects, then at least one Process Monitoring Object must be published to apply the message filter.

If you selected a Business Process while creating a Process Monitoring Object, and after selecting the Business Process Model, if you modified the Message Filter in the Business Process Model, the changes made in the Message Filter will not reflect until you select a Business Process once again. If the Message Filter is modified or updated in the Business Process Model, then at least one Process Monitoring Object, where this message filter is used, must be published to get the modifications.

The Business Process name appears in the Business Process and all the Activity Events for the selected Business Process appear under Events.

Selection of Activity events such as Activity Start and Activity End is required to retrieve the attributes of the process available for that particular activity state. For example, in an Order-to-Cash Business Process, there are three activities - Register Order, Get Credit Rating, and Approval by Manager. Attributes such as ProductID, Quantity, and so on are available at the Activity start of the Register Order activity. At that point of time, attributes like Credit rating are not available and it is available only at the Activity End of the Get Credit Rating activity.

3. Select the **Show Business Process Readonly View** option to view the selected Business Process in the read-only mode. The selected Business Process Model is displayed in the read-only mode on the right pane. This property is not stored when the document is stored. Users need to select this property each time they open the document to view the process model.
4. Select the required business process and activity events under the Events section.
5. Select **Monitor Activity** if you want to [monitor based on the activity](#).

Monitoring of the activity events and activity level monitoring is not possible for short-lived processes. The  icon next to the Monitor Activity text box is activated.

6. Click .

The Select Process Activity dialog box opens with a list of activities. Select the required activity that you want to monitor. The name of the activity appears in the text box.

This is required when you want to monitor based on activity start and/or end-time and not process start and /or end-time. For example, in an Order-to-Cash process, `Ship order to customer` is an activity and if you want to measure an aspect called `Count of orders Shipped in last 1 week`, then it is important to have the Activity start and Activity end time of the `Ship order to customer` activity and not of the Process. It is explained in detail under the Monitoring Based On field on the [Selecting Datasource](#) the Business Measure Editor page.

Important: For monitoring an activity, select the activity needed while modeling the Process Monitoring Object. Under the Events column, all the activities of the process are listed, which means that all the attributes of all the activities are available for selection on the Select Attributes page. Ensure that you select only those activities of the process that are executed before the selected activity so that the attributes available until that point of time are available for monitoring.

Example: Assume that in an Order-to Cash process, you have three activities, namely Register Order, Approve Order, and Ship Order that are sequential and you have selected the Approve order activity for monitoring. In the current scenario, it is possible that you select all the three activities and then proceed to the Select Attributes page to retrieve attributes for all the three activities. However, as you have selected the Approve order activity for monitoring, the Ship Order activity attributes must not be available as this activity is not executed yet, whereas the attributes of the Register order activity must be available as this activity has executed before the Approve order activity. So, while designing the Process Monitoring Object, ensure that you select attributes for the Register order and Approve order activities only.

7. Click **Next** to select the Process or Activity Attributes related to the Process and Activity Events.

The Select Attributes page appears with the selected Activity Events on the left pane. By default, the Select Attributes page displays the Process Events.

On selection of a Process or Activity Event, the relevant Process or Activity Attributes are displayed on the Attributes page.

The Business Process is selected.

Selecting business process attributes

You have to select the process or activity attributes of the business process that are crucial for business and which you would like to monitor.

To select business process attributes:

1. Select the required **Process Event** or **Activity Event** to select the required Process or Activity Attributes. The Attributes, along with their alias names and XPath, for the selected Process Event or Activity appear on the right pane. The Standard attributes Process Name or Activity Name and Status appear under Process and Activity start column and Lead Time appears under Process and Activity End column.
2. If you selected the **Activity Event**, then the attributes that you have selected for message filtering while designing a business process model, appear along with the Standard attributes.
3. UseStandard Attributes. See [Usage of standard attributes](#).
4. By default, the alias name is same as the name of the Attribute. Change the name of the alias, **Alias Name**, if required.
5. Select the required attributes of the Process specific messages from the **Process Variable Selection** table.
6. Process specific messages (created through Schema Fragment) element does not have a type while modeling a business process model and therefore, by default, type `string` is added to it at BAM layer. You can change the same in business process model if you wish to have a different type for it.
7. Select a Date-time field as Process Monitoring Object attribute to monitor data over a business date range.

8. The date-time field that is available as a message map element in Business Process Model flow is the business date.
9. Select the **Include contextual information** option to get the [contextual data from a Web service](#), if required.
10. Click **Next** to select a Web service to get the Contextual information, as the third step in the creation of a Process Monitoring Object.

Select this option if you require some contextual data for a process or an activity. In the Order-to-Cash process example, at Activity Start of Register Order you have an attribute called ProductID. Assume that you want to increase the monitoring context by including some more details about the Product. You want to add ProductName and UnitsInStock attributes, that are not there as attributes for that activity and are present in existing database (different table) or in some external database, then you can call these attributes using Include contextual information option. These attributes have no role to play in process execution, and you can utilize them to increase the monitoring context.
11. Click **Next** to view the Summary page, if you have not opted for including contextual information. A Summary page appears displaying the Attribute Source (Process or Activity), Attribute Alias Name, and the Data type fields.
12. Click **Finish** to complete the creation of Process Monitoring Object on the [Summary](#) page.

Activity-level monitoring

Activity-level monitoring is about monitoring the activities or sub-processes of a process against a defined time frame.

By default, only the processes are monitored over a period. For example, if you consider Order handling processes, where various processes involved in the Order handling mechanism like Order booking, Completed orders, Pending orders, Shipping orders and their corresponding execution time are monitored. However, in case of activity-level monitoring, you can monitor the activities involved in a process. For example, if you select Pending order process, activity-level monitoring helps, you to monitor the total number of pending orders, classify the pending orders per client, and know the pending status and so on.

Note: This type of monitoring excludes Info type activities and Events.

Publishing a process monitoring object

Before you begin:

- [Create](#) a Process Monitoring object.

Publishing the Process Monitoring Object ensures to use it as an application at runtime.

To publish a process monitoring object:

1. [Access the Process Monitoring Object.](#)
2. [Publish the Process Monitoring Object to organization.](#)
The Quick Access Menu option is not supported in the linked topic.
3. Publish multiple Process Monitoring Objects at a time by publishing the folder or project that contains the Process Monitoring Objects.

The Process Monitoring Object is published to the organization.

Example

In the Order-to-Cash process, when you create a Process Monitoring Object with attributes OrderID, UnitsInStock and ProductID and publish the Process Monitoring Object, then the attributes appear as a list in the database.

On instantiation of the process, the corresponding columns for the attributes are filled with respective data. If you want to edit the published Process Monitoring Object (delete an attribute, UnitsInStock, and want to add attributes, ProductQuantity and CustomerID to the published Process Monitoring Object), then you should republish the Process Monitoring Object.

- If you choose not to delete the data in the database, the modifications are updated in the database tables. You can find the following attributes in the database tables: OrderID, UnitsInStock, CustomerID and ProductQuantity. You may notice that UnitsInStock attribute still exists in the database with old content.
- If you choose to delete the data, then the previous data is removed from the database tables. You may notice that the attribute UnitsInStock along with the related data will be deleted from the database tables.

Runtime scenario

When the Business Process is executed, the data of that process is available in BAM through the MDM. This data is used in the Dashboard through Business Measures and KPIs.

Unpublishing a process monitoring object

Before you begin:

- [Publish](#) the Process Monitoring Object.

You must unpublish a Process Monitoring Object to stop monitoring the selected attributes. If you unpublish a Process Monitoring Object, it becomes inactive. The business data is not collected for such Process Monitoring Objects, and the metadata from runtime is not removed as it is required for republishing.

To unpublish a process monitoring object:

1. In the Workspace Documents (Explorer) window, right-click the published (Process Monitoring Object) and select **Unpublish**.

During unpublish, a Confirm dialog box opens stating, "Do you want to retain old data that talks about the business data collected until that time."

If you choose to delete or retain the business data, a message appears stating, "Process Monitoring Object Unpublished successfully." If you choose to retain the business data, the business data is retained.

If the Process Monitoring Object is not published before Unpublish is triggered, a message appears stating, "Process Monitoring Object is not published."

2. If you select **Yes** or **No**, then "Process Monitoring Object Unpublished successfully" appears as a notification.
 - If the Process Monitoring Object is already unpublished, then the message stating "Process Monitoring Object already Unpublished" appears.
 - If the selected Process Monitoring Object fails to get unpublished, an exception message appears stating the error.

The selected Process Monitoring Object is unpublished.

Editing a process monitoring object

Before you begin:

- [Create](#) a Process Monitoring Object.

You can edit a Process Monitoring Object to add or remove attributes. You cannot edit the name of the Process Monitoring Object.

To edit a process monitoring object:

1. [Access the Process Monitoring Object](#) and double-click it. Alternatively right-click the Process Monitoring Object and select Open.
The Process Monitoring Object is displayed in the Process Monitoring Object Modeler.
2. Make the appropriate modifications to the Process Monitoring Object and click **Save**.
3. If you modify or edit Process Monitoring Object attributes, then you should run through the respective Business Measure editor to change the query and then publish.

The modifications made to the Process Monitoring Object are saved.

After you complete this task, you must [publish](#) the Process Monitoring Object.

Deleting a process monitoring object

Delete a Process Monitoring Object if it is not required anymore.

Consider the following when you delete a Process Monitoring Object:

- If you delete a Process Monitoring Object, the design time is also deleted from CWS.
- If deletion of the selected Process Monitoring Object fails, an exception message is displayed stating the error.

To delete a process monitoring object:

1. Perform one of the following:
 - a. In the Workspace Documents (Explorer) window, right-click the Process Monitoring Object and select **Delete**.
 - b. In the Workspace Documents (My Recent Documents) window, select the Process Monitoring Object, click  and select **Delete**.
The Confirm dialog box opens.
2. Perform one of the following:
 - Click **Yes** to delete the Process Monitoring Object.
The Process Monitoring Object is deleted from the project content tree.
 - Click **No** to cancel the operation.

Building business measures

A Business Measure is a measurement of an aspect of a process to assess business performance. The Business Measures are calculated across multiple runs of the process combined with business data to find the Average, Maximum, Minimum, or Total number of occurrences.

A Business Measure can be used to define the calculation for a Graph or Key Performance Indicator (KPI), which measures performance against a business objective. The value of the Business Measure can also be captured and visualized through a Dashboard.

Apart from processes, there is a possibility that an enterprise would like to monitor data objects, which do not have any relation with any AppWorks Platform business processes. Enterprises can reuse the BAM infrastructure for monitoring by synchronizing these data objects to a BAM system and then build Business Measures on them.

The Enterprise Data Object (EDO) is the persistence object for data of the enterprise applications like Supply Chain Management (SCM) and Customer Relationship Management (CRM).

Dashboards (consuming the Business Measures created on EDO) and KPI can be built on EDO but the Business Events cannot be built because it requires Event support from the enterprise applications, which is not supported by many applications. BAM supports EDO, which is synchronized using AppWorks Platform MDM component to the same hub database where the BAM hub is configured.

A Web service can also be registered as a Business Measure where Web services are consumed in the Business Measures and these Business Measures are used to visualize the required data in an output format using a custom dashboard in BAM. The Web services can be used to define complex queries such as making a compound measure using two Business Measures. The same can also be used in the KPI Editor for closed loop monitoring.

The Business Measure Editor facilitates the creation of Business Measures and thereby helps in monitoring the performance of a Business Process, its activities, and also an Enterprise Data Object.

The high-level tasks that you can perform using AppWorks Platform Business Measure editor are:

1. [Creating a Business Measure](#)
2. [Publishing a Business Measure](#)
3. [Unpublishing a Business Measure](#)
4. [Editing a Business Measure](#)
5. [Deleting a Business Measure](#)

Example

You can define a Business Measure called Average Shipping duration for a high valued customer John through the Business Measure Editor by using the aggregate function Average and using the attribute `Lead time for Shipping` in a select clause and using customer name John in the where clause of the query. This could be used, in turn, to create a KPI called `Average Shipping Duration < 5 days` and can also be viewed on a dashboard, displaying the value of the monitored item as a trend; alternatively it (Business Measure) can be directly consumed in the Dashboard to display the value of the monitored item.

Creating a business measure

A Business Measure helps in measuring a particular aspect of a business process or any external data. The Business Measure provides a view on the selected attributes, contained in a Process Monitoring Object or the external data source.

To create a business measure:

1. [Select a starting point](#) and select  (Business Measure) to create a Business Measure. The Untitled BusinessMeasure - BusinessMeasure window opens.
2. Enter **Name** and **Description** of the Business Measure.
3. [Selecting a data-source](#) for the Business Measure.
4. Click **Next** to build Query. See [Building query](#).
5. Click **Next** to select a Web service and a time-frame. See [Selecting a web service and a time frame for a business measure](#).
6. Click **Finish**.

The Business Measure is created and added to the project content tree.

After you complete this task:

1. The business measure created is available as a composite control. Drag the composite control in Dashboard (XForms) designer.
2. Right-click or double-click the composite control and specify properties to it such as view type, ranges, and input parameters, and then view it in the dashboard.
3. For further details on creating a User Interface (XForm), see [Creating XForms](#).

4. You can save and assign the created XForms (using composite controls) to a user or role to view it as a dashboard.
5. For further details on assigning the XForm to a user, see [Assigning Tasks to Users in the AppWorks Platform Administrator's Guide](#).
6. You must publish the Business Measure. See [Publishing a business measure](#).

You cannot rename a Business Measure once it is created.

Creating a business measure using standard measures

Standard measures are used for monitoring business processes and to track volumes - that is, values and counts of different aspects of the processes and their associated transaction.

To create a business measure using standard measures:

1. [Select a starting point](#) and [reuse Standard Measures](#).
The Untitled Webservice Interface - Webservice Interface window opens.
2. Ensure that you select the Standard Measures from Cordys Business Activity Monitoring folder > MethodSet BAM Standard Measures 1.0.
The Standard Measures are added to the existing project as a reference.
3. [Create a Business Measure](#) by selecting **Webservice** option as datasource from the Create Measure From list.
The Select Web Service field appears with .
4. Click the  corresponding to the Select Web Service field.
The Select Webservice dialog box opens.
5. Select a Standard Measure, for example,
`GetTotalNoOfInstancesForProcessByActivityByStatus`.
The Standard Measure is selected.
6. Add the remaining Standard Measures similarly.
7. Ensure that **KPI** is selected for **Business Measure For**.
8. Complete the [creation of the Business Measure](#).

Selecting a data-source

When creating a Business Measure, select a data-source on which you want to build a Business Measure to measure an aspect that is used to assess business performance.

To select a data-source:

1. From the **Business Measure For** field, choose **KPI** or **Graph**.

KPI	If you need the Business Measure in form of a KPI, where the monitored result will be a single value, select the KPI option. The Business Measure, of type KPI, can be consumed in the KPI Editor for closed-loop monitoring. Here, based on the defined objectives, conditions and actions can be set and
-----	--

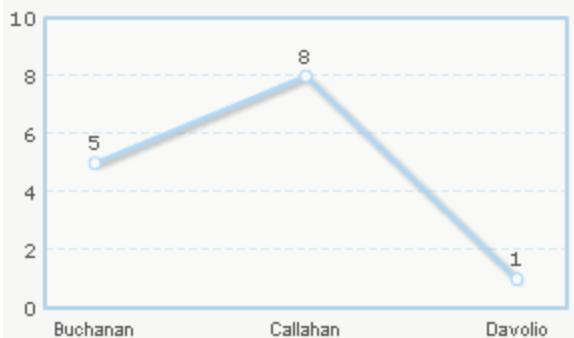
	<p>on meeting of the condition, Action can be triggered. Note: If you choose the Web Service as the data-source, and want to view the Web service in the form of a KPI, you have to create the output in the format as shown here:</p> <pre><tuple> <old> <business element name> <aggregate value name>0</aggregate value name> </business element name> </old> </tuple></pre> <p>For example:</p> <pre><tuple> <old> <employee> <employeeCount>0</employeeCount> </employee> </old> </tuple></pre>
Graph	<p>If you need the measure in form of a Graph, where the monitored result will be in form of Series (X-axis) and Category (Y-axis), select the Graph option. The X-axis denotes the Group by in the query where as the Y-axis denotes the aggregated or calculated value. The Business Measure, of type Graph, can be consumed in the Dashboard Designer (XForm) for creating a view. It is mandatory to specify Group by field for the Graph option.</p> <p>Note: If you have chosen the Web Service option as a data-source, you have to create the output of Graph in the tuple-old (Transactional) format or in a predefined XML Graph output format as shown here:</p> <pre><category> <name>Y-Axis group name</name> <series> <name>group name to be plotted in x-axis</name> <value>group value to be plotted in y-axis</value> </series> </category></pre> <p>For example:</p> <pre><category> <name>Count of Orders per Customer</name> <series> <name>Buchanan</name> <value>5</value> </series> <series> <name>Callahan</name> <value>8</value> </series> <series> <name>Davolio</name></pre>

```

<value>1</value>
</series>
</category>

```

Graphical representation of the XML output format:



- From the **Create Measure From** list, choose Datasource, Process Monitoring Object, Enterprise Data Object, Web Service, or Business Measure.

You can [create a Business Measure using Standard Measures](#) related to Process and Activity. See [Functions of Measure](#) for detailed description of the Standard Business Measures.

Web Service	<p>If you select Web Service, the Select Web Service field appears with .</p> <ol style="list-style-type: none"> Click . The Webservice dialog box opens. Select the required Web service. <ul style="list-style-type: none"> Web services having schema with prefixes other than XSD, are not supported in Business Measure. While creating a Business Measure on top of a Web service, if you need data to be monitored with respect to a timeframe, you should have startTime and endTime as the input parameters of datatype - datetime. You cannot create a Business Measure with UDDI Web services.
Process Monitoring Object	<p>If you select Process Monitoring Object, a query is built on the attributes of the selected Process Monitoring Object. By default, the selection of Process Monitoring Object option appears on the screen.</p> <ol style="list-style-type: none"> Click  corresponding to Select Monitoring Object. The Select Monitor Object dialog box opens. Choose the required Process Monitoring Object.

	<p>3. In the Monitor based on column that gets activated, you need to specify whether you want monitor data based on process or activity start-time and end-time both or independently with only process or activity start-time or end-time. While modeling the Process Monitoring Object if you selected the Monitor Activity option, then the time would apply for activity, that is, whether you want to monitor based on activity start-time and end-time both, or activity start-time independently, or activity end-time independently. This concept is explained with the help of examples. This is also applicable while modeling the Process Monitoring Object, if you have not selected the Monitor Activity option, then you have to specify Process start-time and end-time.</p>
Enterprise Data Object	<p>If you choose Enterprise Data Object, a query is built on the attributes of the selected EDO.</p> <ul style="list-style-type: none"> ■ In the Select Enterprise Data Object list, choose the required EDO.
Business Measure	<p>If you select the Business Measure option, the Select Business Measures field is enabled with .</p> <ol style="list-style-type: none"> 1. Click . The Select Business Measures dialog box opens. 2. Select more than one Business Measures whose response has some common data and then create a single composite Business Measure. 3. Click OK. <ul style="list-style-type: none"> ■ You cannot combine Business Measures of type KPI and Graph. ■ You cannot build expression on the chosen Business Measures because it just combines the response of more than one Business Measures. A composite Business Measure is required if you want to combine data from one or more Business Measures (type: Graph) and view it. For visualization, you can combine the data either in series (X-axis) or in category (Y-axis).

3. From **Monitor based on** field, choose the relevant Dates option to monitor the data.

Process Dates option or Activity Dates option	<ol style="list-style-type: none"> 1. Select Process Dates or Activity Dates to monitor data over a process instance range. The Process Start Time or Activity Start Time and Process End Time or Activity End Time options are enabled. 2. Select the time for monitoring, specifying whether monitoring
---	---

Business Dates option	<p>is based on both the Process or Activity start-time and end-time, or on the Process or Activity start-time and Process, or Activity end-time independently.</p> <p>This option is disabled for EDO and Web service.</p> <ol style="list-style-type: none">1. Select Business Dates to monitor data over a business date range.2. From the Select date-time attribute list, choose the relevant field. <p>For example, in the Order to Cash process, you can select Shipped date as one of the attributes and use it as business date while creating a business measure, which can give you Count of orders based on Shipped date. This means a query can be built irrespective of process or activity start or end time.</p> <ul style="list-style-type: none">■ You can select a business date only if the Process Monitoring Object has business dates.■ You can select only one business date from the list of available business dates selected in a Process Monitoring Object.■ You can select Business Dates option only for Business Measures on Process Monitoring Object and EDO, and this option is disabled for Web services.■ You have to specify business dates in UTC.
-----------------------	---

The Datasource is selected as first step in creating a Business Measure.

After you complete this task:

- Click **Next** to build a query as the second step in the creation of business measure. The Build Query page appears displaying all the attributes, of the already selected Process Monitoring Object or EDO, under the Attributes column, Expressions, and query clauses.

If you are creating Business measure with [Web service](#) or [Business Measure](#) data-source, specify the Web service definition set details in a second step.

Web services

Web Services are self-contained, modular applications that can be described, published, located, and invoked over the Web. These services range across different businesses, partners, customers, and so on catering to various business activities. The service could be as simple as World Time Service that gives the time for a city, a credit card transaction, or a full blown business service. A complex service would include other services within. Web Services share the business logic, data, and processes through a programmatic interface and are just like application programming interfaces (APIs) that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

In BAM context, BAM can monitor data from a Web service. The Web service can get the result either from an expression or from a database query result.

Web services are consumed in the Business Measures and these Business Measures are used to visualize the required data in an output format using custom dashboard in BAM.

Web services can be used to define complex queries like making a compound measure using two business measures. For example, in an Order handling process, the order status changes from Received to In-progress to Accepted. One of the Service Level Agreements (SLAs) for this process is that for 90% of the Orders received, the time between In-progress and Accepted is maximum 5 days. You need to show a KPI that reflects when the SLA is met (if 90% of the orders received are handled on time). For this we need to define two business measures (that can be achieved using Business Measure Editor): (a) one that counts the total number of processes that reached the Accepted state, (b) one that counts the number of processes that reached the Accepted state in 5 days.

On top of these two measures we need to define a compound measure that calculates the percentage of processes on time, in other words, $(A/B)*100$. This can be achieved by a custom Java call and making it available as a AppWorks Platform Web service with an expected output format that BAM requires.

BAM can monitor the Web services that are available in AppWorks Platform environment only.

Process and activity monitoring types

Monitoring Type	Example
Process Monitoring based on Start-time and End-time	The data for monitoring is fetched from the Process start-time and Process end-time. For example, in an Order-to-Cash process you want to measure an aspect called Total no. of orders processed in the last 3 days. Assuming that today is April 22, 2008, the Total no. of orders are fetched for the process which started 19th April onwards and ended on 22nd April 2008. In such case, the following condition is applied for fetching the data: Process start-time \geq 19.04.2008 and Process end-time \leq 22.04.2008
Process Monitoring based on Start-time	The data for monitoring is fetched from the Process start-time, irrespective of end-time. For example, in an Order-to-Cash process, you want to measure an aspect called Total no. of orders processed in the last 3 days. Assuming that today is April 22, 2008, the total no. of orders are fetched for the process which started between the 19th and the 22nd April 2008. The following condition is applied for fetching the data: Process start-time is between 19.04.2008 and 22.04.2008
Process Monitoring based on End-time	The data for monitoring are fetched from Process end-time irrespective of start-time. For example, in an Order-to-Cash process you want to measure an aspect called Total no. of orders processed in last 3 days. Assuming that today is April 22, 2008, the

Monitoring Type	Example
	total no. of orders are fetched for the process, which ended between 19th and 22nd April 2008. The following condition is applied for fetching the data: Process end-time is between 19.04.2008 and 22.04.2008
Activity monitoring based on start-time and end-time	The data for monitoring is fetched between Activity start-time and Activity end-time. For example: In an Order-to-Cash process, <code>Ship order to customer</code> is an activity and if you want to measure an aspect called <code>Count of orders Shipped in the last 1 week</code> . Assuming that today is April 22, 2008, the total number of orders is fetched for the activity that started April 15, 2008 onwards and ended on April 22, 2008. The following condition is applied for fetching the data: <code>Activity start-time >= 15.04.2008 and Activity end-time <= 22.04.2008</code>
Activity monitoring based on start-time	The data for monitoring is fetched from Activity start-time irrespective of Activity end-time. For example, in an Order-to-Cash process, <code>Ship order to customer</code> is an activity and if you want to measure an aspect called <code>Count of orders Shipped in last 1 week</code> . Assuming that today is April 22, 2008, the total no. of orders are fetched for the activity which started between April 15, 2008 and April 22, 2008. The following condition is applied for fetching the data: <code>Activity start-time is between 15.04.2008 and 22.04.2008</code>
Activity monitoring based on end-time	The data for monitoring is fetched from Activity end-time irrespective of Activity start-time. For example, in an Order-to-Cash process, <code>Ship order to customer</code> is an activity and if you want to measure an aspect called <code>Count of orders Shipped in last 1 week</code> . Assuming that today is April 22, 2008, the total number of orders are fetched for the activity which ended between April 15, 2008 and April 22, 2008. The following condition is applied for fetching the data: <code>Activity end-time is between 15.04.2008 and 22.04.2008</code>

Building query

When creating a Business Measure, you must build a query on the Process Monitoring Object or the EDO to retrieve the business information from the selected data source.

To build a query:

1. To build a **Select** clause query, click the **Select** clause text box and add the required attributes.
 - It is mandatory to use the Aggregate Functions **expression** in the Select query. You can use any number of aggregate functions in the query.

- If you are using a lead-time attribute in the query, you can specify the unit of measure (second, minute, hour, day, week, or month) for the same by using the list provided below the list of attributes.
 - When you change the alias name in the query of the Business Measure query builder, you must edit the KPIs built on that Business Measure to rebuild the expression. Consider the select clause: Select Count(OrderID) TotalOrders , where TotalOrders is the alias name. When you change the alias name, TotalOrders, in the query of the Business Measure query builder, you must edit the KPIs built on that business measure to rebuild the expression.
 - While writing a query on any AppWorks Platform table in the PostgreSQL database table, you must enter aliases only in upper case.
2. To add a **Where** clause query, select the **Where** text box and add the required attributes.
 3. To add a **Group by** clause query, select the **Group by** text box and add the required attributes.
 - The Group by clause appears only if the Business Measure you are trying to build is of the type Graph.
 - You can use any number of Group by fields.
 4. To add a **Having** clause to the query, select the Having text box and add the required attributes.
 - The Having clause is used in conjunction with the Group by clause.
 - An alias name cannot be validated when the Having clause is used in conjunction with the Group by clause. For example, Select sum(PROCESS_NAME) processname group by PROCESS_STATUS having sum(processname) > 5 , is not allowed.
 5. Click **Validate Query** to run and validate the built query.
A dialog box opens upon successful validation or failure of the query.

A query is built and validated.

After you complete this task:

- Click **Next** to select a Web service and a time-frame for the Business Measure.

Querying by Select clause

When you build a query, you have select the attributes as a Select clause on which you have to build the query.

To add fields to the Select clause:

1. Click the **Select** box.
The Select box gets the focus, indicated by the highlighted title bar. By default, the Select window is highlighted.
2. Add the required fields to the Select box by double-clicking the field.
This actions adds that particular field to the Select clause. If you are adding only EmployeeID from the fields list, the Select query box appears as EmployeeID.
The selected field or fields are populated into the Select box.
You can add various expressions for each field in the Select clause. You must use the Aggregate Functions expression to build a Select query.

Query data for the Select clause is created.

Querying by Where clause

When you build a query, you have to build a business condition as a Group by Where clause to get the business data.

To query by Where clause:

1. Click the **Where** clause box to highlight the Where text box.
2. Select the attribute from the **Attributes** list and double-click the highlighted attribute.
The selected attribute is added to the Where clause box.

Tip: You can add expressions for the Where clause to run a query.

A Where clause for each query will be a combination of the following macro:

<field name> <operator> <value>

3. To replace the operator tag, click **<opr>** and select the required expression from the **Expressions** list.
4. To replace the value tag, click the **<value>**, and define the value by a parameter or a string.

By a parameter	Double-click the corresponding attribute in the Attributes list: <ul style="list-style-type: none">■ The field name is added with a colon before it, indicating that the value has to be input by the user. A parameter value can be recognized only if it has a colon before the name. This should be provided when you type in the value of the parameter manually.
Type in	After selecting the tag, type the field name or the value, as required.

The query data for the where clause is created.

Querying by Group By clause

Before you begin: The aggregate fields present in the Group by text box should be present in the Select text box also.

When you build a query, you can group the result of the query as a Group by clause to get the business data.

To query using the Group by clause:

1. Select the **Group by** clause text box.
2. Double-click the required attribute from the **Attributes** column.
The selected attribute is added to Group by clause text box.

Querying by Having clause

Before you begin: The Having clause is used in conjunction with the Group by clause.

To query using the Having clause:

1. Select the **Having** text box to add the Having clause to the query.
2. Select the required attribute from the **Attributes** list and double-click it.
3. You can add any kind of expressions to the Having text box by double-clicking the Expression list.

When the Having clause is used, the fields present in the Having text box should also be present in Group by expression.

Fields and expressions on query builder

Attributes	All the attributes of the selected Process Monitoring Object appear here. You can add these attributes to Select box and validate against the fields selected under Where clause box.
Clear	Clears the defined query.
Data type	The data type of all the attributes of the selected Process Monitoring Object appear here.
Group by	This query box displays the Group by clause of the total query.
Having	Having clause should go along with the Group by clause.
Select	Displays the Select clause of the query. This contains a text area where the expressions can be built or typed.
Validate Query	Validates the defined query.
Where	Displays the Where clause of the query. You can enter data inside this box, only if some data is already entered in the Select box. The content inside this box can be typed, or can be completed by adding the field selected to the query box.

The Expression group contains operators, which are used to frame various conditions for validation.

Arithmetic Operators

%	Shows percentage
*	Multiplies
+	Adds
-	Subtracts
/	Divides

Comparison Operators

<	Less than
<=	Less than or equal to
<>	Not equal to
=	Equal to
>	Greater than
>=	Greater than or equal to

Logical Operators

And	Connects two
Between	Displays relation
In	Displays direction
Like	Displays similarity
Not	Displays negation
Or	Displays alternative

Aggregate Functions

Avg	Displays average of values
Count	Displays total number or amount
Max	Displays maximum value
Min	Displays minimum value
Sum	Displays sum of values

Constants

Null	Void
()	Closing
'	Separation
"	Opening quote

Selecting a web service and a time frame for a business measure

The published Business Measure (BM) is saved as a Web service and can be used to in a KPI or Dashboard to view the results of the defined query. You can define a time frame to execute the BM in that time period.

To select a Web service and a time frame for a business measure:

1. Select any one of the **Web service Definition Set** option under **Select Web service Definition Set** section.

New	<p>Select New option if you want to provide a new Web service Definition set to the BM. This option is selected by default. The Web service Definition Set Name and Web service Definition Set Namespace fields are enabled; enter a name and namespace in the respective fields. By default, the Web service Definition Set Name gets filled by the BM name.</p> <p>The Web service Definition Set Name and Web service Definition Set Namespace are mandatory fields and should not begin with a number. Blank spaces or special characters are not allowed.</p>
Existing	<p>Select Existing option if you want to apply any existing Web service Definition set. The Existing Web service Definition Set Name field is enabled with .</p> <p>The Existing Web service Definition Set Name is a required field and it should not begin with a number. Blank spaces or special characters are not allowed.</p> <ul style="list-style-type: none"> ■ Click , and select the web service Definition Set name from the Select Web service Definition Set dialog box. The Existing Web service Definition Set Namespace field remains disabled. Depending on the Web service Definition Set you select, this field is populated with Namespace automatically.

2. Select the required Time Frame option.

Rolling Dates	Select the Rolling Dates option to fetch data from a repeating time period of a specific length that moves continuously, which can be year, month, week, day, hour, minute, or second. Only one such period is selected to
---------------	--

	<p>calculate the BM; for example, the last month. You must also select whether to evaluate data for the Last full period or for the Current period in progress.</p> <ul style="list-style-type: none"> ■ For example, if you set the period to 1 month and select Last full period, the BM (BM) will always reflect the value based on the last completed month. In the month of March, the last completed month would be February. When April starts, the BM value is no longer based on the data from February but would be based on the data from March. At all times, the BM calculation is based on a full month of data. Alternatively, if you select the Current period in progress, the BM calculation is based on the current month and date. On March 12th, the BM is calculated based on the Feb 12th to March 12th data. When April starts (say April 1st), the BM calculation value will then be based on the data from March 1st to April 1st. <p>When the Last full period option is considered and the time period is configured in terms of hours, minutes, or seconds, it will be based on the UTC standard.</p> <ul style="list-style-type: none"> ■ For example, if a Business Process Model(BPM) is executed at 5:20 PM IST (UTC+5:30), it will be recorded in the BAM repository as 11:50 AM UTC. If a time frame in a BM is configured on the last completed 1 hour, then the process instance data is reflected in the BM response only after 5:30 PM IST, as the one hour completion in terms of UTC is between 11:00 AM UTC to 12:00 PM UTC.
Static Range	<p>The Static Range option fetches data for the range falling within the specified From and To dates. Both the From and To are mandatory fields for Static Range. The Static Range option is selected when you visit this page for the first time, by default.</p>

If you choose Static Range option, From and To date fields are enabled. If you choose Rolling Dates choice, the Period and Base Period fields are enabled. When you select one of the above options, the fields in the remaining option are automatically disabled.

3. To get data for a specified range, select **Static Range**.
 - a. To select the start date to fetch the data, click  next to From.
 - b. To select the end date to fetch the data, click  next to To.
4. To get data based on rolling dates, select **Rolling Dates**.
 - a. Type the period in Period, and select the duration in the range of Minute(s) or Hour (s) or Day(s) or Week(s) or Month(s) from the list.
 - b. Select one of the following options as the **Base Period**:

Last Completed Period	Data required for last full period.
Current Period in-progress	Data required for current period in-progress.

The Web service and Time Frame for the BM are selected.

After you complete this task:

1. Click **Finish** to complete the creation of a BM.
The BM and the corresponding composite control is created and is added to the project content tree. The BM gives a processed value based on the underlying query and the time frame (if applicable) defined in BM Editor.
2. The BM created is available as a composite control and you can drag the composite control on Dashboard (XForms) designer.
3. Specify properties (on right-click or double-click the composite control) such as view type, ranges, and input parameters, and then view it in the dashboard.
4. For further details on creating a User Interface (XForm) and the task that is automatically generated when you create a user interface see [Creating XForms](#).
5. Save the XForm and assign it to a user or role. For more details on assigning the XForm to a user, see [Assigning Tasks to Users in the AppWorks Platform Administrator's Guide](#). You can access the dashboard created through XForms designer.
6. You must [publish](#) the BM.

Publishing a business measure

This topic describes the procedure to publish a Business Measure. The Business Measure can be used for creating a KPI and also can be consumed as composite controls to create a dashboard.

Before you begin:

- [Create](#) a Business Measure.

Publishing the Business Measure ensures that you can use it as an application at runtime.

To publish a business measure:

1. [Access the Business Measure](#)
2. [Publish it to organization](#)

The Business Measure is published to organization.

While creating a Business Measure, if an unpublished Process Monitoring Object is used, then on Publish of Business Measure the used Process Monitoring Object also gets published.

If you have edited a published a Business Measure and saved the changes, then you have to re-publish the Business Measure to reflect the edited changes from design time to run-time.

Unpublishing a Business Measure

You need to unpublish a Business Measure if you no longer need it.

Before you begin:

- [Publish](#) the Business Measure.

To unpublish a business measure:

1. In the Workspace Documents (Explorer) window, right-click the published  (Business Measure) and select Unpublish.

A dialog box opens stating "Business Measure unpublished successfully."

- During unpublish, the Web service created is deleted to make it unavailable for querying.
- If a Business Measure is not published before, Unpublish is triggered, a message stating "Business Measure is not published appears."
- If the selected Business Measure fails to get unpublished, an exception message appears stating the error.

The selected Business Measure is unpublished.

Editing a Business Measure

Before you begin:

- [Create](#) a Business Measure.

You can edit a Business Measure if you want to modify the query.

To edit a business measure:

1. [Access the Business Measure](#) and double-click it. Alternatively right-click the Business Measure and select **Open**.
The Business Measure is displayed in the Business Measure Editor.
2. Make the appropriate modifications to the Business Measure and click **Save**.
See [Selecting Datasource](#) and [Query Builder](#) pages to complete the editing process of a Measure with the status unpublished.

The modifications made to the Business Measure are saved.

After you complete this task:

- You must [publish](#) the Dashboard (XForm).

Deleting a business measure

Delete a Business Measure if it is no longer needed.

Before you begin:

- [Create](#) a Business Measure.

Consider the following when you delete a Business Measure:

- If you delete a Business Measure, the design time is also deleted from CWS.
- The delete operation deletes the artifacts, which are created during creation of the Business Measure (Web service, Composite Control, HTM).
- If deletion of the selected Measure fails, an exception message is displayed stating the error.

Delete a (Business Measure) in one of the following ways:

- In the Workspace Documents (Explorer) window, right-click the Business Measure and select **Delete**.
- In the Workspace Documents (My Recent Documents) window, select the Business Measure, click  and select **Delete**.

The Confirm dialog box opens.

Perform one of the following:

- Click **Yes** to delete the Business Measure. The Business Measure is deleted from the project content tree.
- Click **No** to cancel the operation.
- Click **Show Used by** to retrieve the details of the documents that use the Business Measure.

Defining a KPI

AppWorks Platform BAM provides you the facility to monitor and capture critical information pertaining to Business Process or Activity with the help of KPIs. KPIs are created with the sole objective of monitoring and capturing process or activity related information and in triggering business actions and outbound events as specified. KPIs work on aggregated process instances.

You can create KPIs using multiple types of data sources: external data and business process data. When the KPI has to be created using external data, a Business measure has to be created and can be used to build the KPI. The KPI can be associated with different business measures that enable monitoring and capturing information about various aspects of the data sources. It is possible to define specific conditions, which when evaluated to *true* can trigger various business actions.

If the source for creating a KPI is only the business process data, then you can directly create the KPI on the business process model.

You can create dashboards using KPIs and configure to view the trend behavior. KPI trends are shown in the dashboard based on each schedule run of the KPI and monitored results captured based on it.

Note: Trend Analysis refers to the analytical comparison of monitored KPIs and identifying their patterns over time. Trends can be plotted over a period based on the processed values of the expression.

The following tasks can be performed using the KPI Editor:

1. [Creating a KPI](#)
2. [Creating a KPI directly from a Business Process Model](#)
3. [Publishing a KPI](#)
4. [Editing a KPI](#)
5. [Unpublishing a KPI](#)
6. [Deleting a KPI](#)

Creating a KPI

KPIs help to monitor and capture critical information pertaining to Business Process or Activity. KPIs are created over a various data sources, one of which can be a Business Measure. KPI creation can re-use the data set that is already filtered or defined in the business measure. It is also possible to enhance the data set that is being received from business measure. Use scheduled capturing of the details that are to be monitored with an explicit scheduler.

Before you begin:

- [Create a Business Measure.](#)

To create a KPI:

1. Select a starting point and select  (**KPI**).
The Untitled KPI - KPI window opens.
2. *Required.* Enter a **Name** and **Description** of the KPI.
Special characters are not allowed and the KPI name should not begin with a number.
3. Provide a **Goal** of the KPI.
For example: Monitor the average cycle time of OrdertoCash Process to ensure that all the escalations are handled in time.
4. [Define Objectives](#) as part of creating a KPI.
5. Click **Next** to attach a Business Measure, which is the source for monitoring.
6. Click **Next** to build expressions on top of the Business Measure selected and also provide Web service definition set details.
You can modify this query further in the Expression builder.
7. Click **Finish** to create KPI with an attached Business Measure.
A KPI is created and added to the project content tree.
8. Alternatively, you can also define Scheduled execution of the KPI, and define actions when particular conditions get satisfied.
 - a. Select the **Define Schedule** option.
 - b. Click **Next** to define the Schedule for the executing KPI. It is possible to have multiple time slots for each rolling period that is selected during which the KPI should be executed.

9. Select the **Define Actions** option (optionally you can also select Model Email to be triggered as an action check box, if you want to model an email template and send email as part of action) and click **Next**.
10. Click **Finish**.

A KPI is created and added to the project content tree.

You cannot rename a KPI after it is created.

After you complete this task:

- [Publish](#) the KPI to the organization, and use it to create the required dashboards.

Defining objectives of a KPI

Defining the objectives form is the first step in KPI creation. This topic explains the options available and the details needed for each of the parameters being considered.

To define objectives of a KPI:

1. Click **Unit of Measure** under the Define Objectives area and select the required unit: **Second, Minute, Hour, Day, Week, Month, Year, Integer, Percentage, Float**.
 - For measures such as Average completion time of a process and Average completion time of an activity, select **Unit of Measure** in terms of time, such as Minute, Day, Hour, Week, Month, or Year. The Unit of Measure specified while creating a Business Measure takes priority over the same specified in KPI execution. The Unit of Measure specified in the KPI execution is for visualization.
 - For measures such as Total no. of instances for all processes, select **Unit of Measure** as Integer.
 - For measures such as Percentage of Waiting instances of a Process, select **Unit of Measure** as Percentage.
2. *Required.* Enter a **Target Value**.
Target value is a value that an enterprise sets to measure the performance of a process, organization unit, enterprise data object, and Web service.
3. Click  to add the range limits.
4. For each range limit, specify a name, and enter the lower and upper limits in their respective columns as required.

You can have lower limit of the first range and upper limit of the last range as unbounded, that is, you can leave the lower limit of the first range and the upper limit of the last range empty - the boundaries for these are filled at run-time accordingly.

Attaching a business measure

Before you begin:

- Create a business process model.
- Create a Business Measure.

Attach a Business Measure to KPI to enable monitoring and capturing information about different aspects of the Business Process or external data source as against business objective.

To attach a business measure:

1. Click  and select a measure from the list of measures.
All the Business measures created through Business Measure Editor are listed.
2. If input is required for the selected Business Measure, then the columns under Set Input Parameters section are enabled and the Input, Datatype, Relation and Value fields appear. Enter values for input parameters.

Datatype	Displays the type of the parameter.
Input	Displays the input parameters of the selected business measure, for which input is required.
Relation	Displays the operator, for example, >, =.
Value	Type in the input value depending on the parameter and data type of the content.

3. The same time frame as defined while creating a Business Measure appears under the Set Time Frame section. If you have specified Static Range while creating Business Measure then only Static Range option is enabled here and same applies for Rolling Dates. You can change the values of the time frame, if required.

Functions of measures

The following tables list the Standard Measures.

Standard measure for all processes

Measure name	Web service operation name	Functionality
Total no. of aborted instances for all processes	GetTotalNoOfInstancesForAllProcessForStatus	Provides the count of all instances with aborted status for all processes.
Total no. of completed instances for all processes	GetTotalNoOfInstancesForAllProcessForStatus	Provides the count of all instances with complete status for all processes.

Measure name	Web service operation name	Functionality
Total no. of instances for all processes	GetTotalNoOfInstancesForAllProcess	Provides the count of all instances for all processes.
Total no. of waiting instances for all processes	GetTotalNoOfInstancesForAllProcessForStatus	Provides the count of all instances with waiting status for all processes

Standard measure for specific process

Measure name	Web service operation name	Functionality
Average completion time of a process	GetAverageCompletionTimeForProcess	Provides average completion time for all instances of a specific process.
Total no. of instances of a process	GetTotalNoOfInstancesForProcess	Provides the total count of all instances for a specific process.
Total no. waiting instances of a process	GetTotalNoOfInstancesForProcessForStatus	Provides the count of all instances with waiting status for a specific process.
Total no. of completed instances of a process	GetTotalNoOfInstancesForProcessForStatus	Provides the count of all instances with complete status for a specific process.
Total no. of aborted instances of a process	GetTotalNoOfInstancesForProcessForStatus	Provides the count of all instances with aborted status for a specific process.
Percentage of waiting instances of a Process	GetPercentageOfInstancesOfProcessByStatus	Provides the percentage of process instances with waiting status with regard to total number of process instances.
Percentage of completed instances of a Process	GetPercentageOfInstancesOfProcessByStatus	Provides the percentage of process instances with complete status with regard to total number of process instances.
Percentage of aborted instances of a Process	GetPercentageOfInstancesOfProcessByStatus	Provides the percentage of process instances with

Measure name	Web service operation name	Functionality
		aborted status with regard to total number of process instances

Standard measure for specific activity

Measure name	Web service operation name	Functionality
Average Completion Time for Path	GetAverageCompletionTimeForPath	<p>Provides the Average Time taken to traverse from the start of the process to end of the selected activity. Note: This measure is used for processes which have multiple activities leading to multiple completion paths. This is explained below with an example. For example, assume that a manufacturing company has an Order Approval Process that approves the customer order based on its Account status. If the Account status is OK, the order is shipped otherwise, the Order is canceled. This process comprises two essential activities leading to completion of the process: Order shipped Order canceled If the Process Manager wants Average Completion Time only for shipped orders then, it can be calculated by making use of the above measure which would calculate the average difference between Completion time of the Shipped Order activity and Start time of the Order Approval Process.</p>
Average completion time	GetAverageCompletionTime	Provides average completion

Measure name	Web service operation name	Functionality
of an activity	ForProcessAnActivity	time for all instances of a specific activity.
Percentage of aborted instances of an activity	GetPercentageOfInstancesOf ProcessByActivityByStatus	Provides the percentage of activity instances with 'aborted' status with regard to total number of activity instances.
Percentage of completed instances of an activity	GetPercentageOfInstancesOf ProcessByActivityByStatus	Provides the percentage of activity instances with 'complete' status with regard to total number of activity instances.
Percentage of waiting instances of an activity	GetPercentageOfInstancesOf ProcessByActivityByStatus	Provides the percentage of activity instances with 'waiting' status with regard to total number of activity instances.
Total no. aborted instances of an activity	GetTotalNumberOfInstancesFor ProcessByActivityByStatus	Provides the count of all instances with 'aborted' status for a specific activity.
Total no. completed instances of an activity	GetTotalNumberOfInstancesFor ProcessByActivityByStatus	Provides the count of all instances with 'complete' status for a specific activity.
Total no. of instances of an activity	GetTotalNumberOfInstancesFor ProcessAnActivity	Provides the count of all instances for a specific activity.
Total no. waiting instances of an activity	GetTotalNumberOfInstancesFor ProcessByActivityByStatus	Provides the count of all instances with 'waiting' status for a specific activity

Building expressions

Usually you work with only one Business Measure and therefore, the default expression is readily available for the selected Business Measure. However, when you want to create a composite Business Measure, you need to pick more than one Business Measure. In that case you need to build expressions.

To build expressions:

1. In the expression builder, click **Measures**.

The list of already selected measures appears.

If the KPI is created from Business process model context, then the Business measure which is automatically generated based on the aggregation definition selected, will appear.

2. Select and click **Operators**.

The Arithmetic Operators, Delimiters, and Constants list appear. Arithmetic Operators can be used to perform four operations: Addition (+) , Subtraction (-) , Multiplication (*) and Division (div). Delimiters ('(' and ')') can be used to split expressions. Constants (0-9) can be directly used in the expressions.

3. Click the required type of operator list and double-click the preferred operator.

The selected operator is added to the business measure already selected. If you select only one Business Measure, the default expression is filled.

4. Double-click the second measure to appear on the pane.

5. Use operators, if required, to complete the expression building.

For example, if you want to make a composite business measure and want to monitor the Percentage of waiting instances of a process with respect to all started processes, select **No. of waiting instances** business measure and **Total no. of instances** business measure. Build the following expression:

(No. of waiting instances) / (Total no. of instances) * 100

6. Click **Clear** to delete a selected expression.

The expression using a Business Measure is built.

Defining a schedule

You can ensure that a set of actions is executed at a set of preset times by scheduling activities. For example, to trigger a Web service that summarizes orders at the end of each business day, you can set a schedule to be triggered at 17:30 pm every day.

To define a schedule:

1. In the Schedule page, [Create a Schedule](#).

If the time-frame set for the Business Measure attached to the KPI is same as the Rolling Dates the schedule interval would get defaulted with the period selected for the rolling dates. For example, if the period selected for rolling dates is Month then schedule interval would be set to Monthly by default. However, if the period selected for rolling dates is Minute then schedule interval would be set to Hourly by default.

The Schedule is created.

2. In the Schedule page, click  next to Deploy Date to select the date on which the process of monitoring based on schedule should start.

Deploy Date must be greater than the current date and time and lower than the End Date and time. Deploy Date is selected.

3. Click  and select the End Date on which the process of monitoring based on schedule should end.
End Date must be greater than the current date and Deploy Date.
4. Click **Finish** to complete the scheduling process. A KPI and the corresponding composite control is created and added to the project content tree. This composite control is also available under Quick Access Menu of the XForms designer.
5. If you want to visualize the KPI trend of the KPI created as a composite control, drag it on to the Dashboard (XForms) designer.
You can perform Trend Analysis with the help of KPIs. Trend Analysis refers to the analytical comparison of monitored KPIs and identifying their patterns over a period. Trends can be plotted over a period depending upon the processed values of the expression.
6. Double-click or right-click the composite control to specify the properties of the composite control and select **Properties** to define properties for the composite control.
7. Save the XForm and assign it to a user or role.
You can access the dashboard created through XForms designer based on the access permission.
For further details on creating an XForm see [Creating XForms](#) and for further details on assigning the XForm to a user see [Assigning Tasks to Users in the AppWorks Platform Administrator's Guide](#).
8. Alternatively, before clicking **Finish**, select the **Define Actions** option (optionally you can also select Model Email to be triggered as an action check box, in case you want to model an email template and send email as part of action) and click **Next** to [define Actions](#).

A schedule is defined.

Creating a model email template for KPI

The objective of the Model Email template is to send the KPI and related contextual information to the user, when a specified condition based on a business objective is met.

In the Email template, you can include the information that is relevant to the KPI, for example, KPI name, Target Value, Processed Value and so on. You can also customize the contents of the Email template for background color, font size, and formatting.

The components of the Email template can be modeled by changing the HTML code.

The Attributes related to the KPI such as KPI name, Process name, Activity name, KPI processed value, and attributes such as `mailid`, `displayname` and `targetvalue` are available in the Message area under the Message data tree of the Email template and these attributes can be used in any of the components by dropping them in the specific area.

To create a model email template for KPI:

1. In the **Model** tab under the Email Model component, provide the necessary message, to be sent as part of the mail.

2. The Model has Header, Salutation, Body, Signature, Footer, and Application Details components. The selected attributes in the earlier stages are available under the Message area of the Email template. Use these attributes in any Model components by dropping them in the specific area.

For more information on the Email Model template, see **Creating an Email Model**.

The Email template is now created. You can now [generate a Web service on this E-mail template](#) and invoke it to trigger actions.

While creating an Email template:

- The Email template integration with BAM will handle the Message Template creation.
- The Message Metadata (EMail Template > Model Source > Message > Data > Message Metadata) section on the Model Source tab page does not have any significance. Even if you select Message Metadata, the Email template does not show any changes.

Defining actions

The defined actions get triggered when on scheduled interval the specified conditions are met.

To define actions:

1. In the Actions page, click  (**Add Rule**) under Actions and start defining actions. Before the Actions are defined, set the condition for when the actions should be triggered.
2. In the Condition list, select the required condition.
The Condition list appears with conditions based on the range name and predefined conditions based on the target value defined in the KPI definition page.
3. For the Action(s) that should take place, three options are possible.
Select the set based on the need.

Call Web Service Operation	You have to select a Web service to use this action. The system automatically compiles a request in the Request field based on the WSDL of the service selected. If you have selected to send an email notification, then choose the same methodset as defined during email creation and fill the related elements in the entire input XML by removing the value PARAMETER by replacing it with appropriate content.
Invoke Business Process	Allows you to trigger a process. Specify the name of the process and the message for the process to be instantiated.
Notification	Sends a notification to the intended audience.

You can select any of the required type of action or actions. Depending on the selection of Action, that particular tab opens in the entry mode. If more than one type of Action is selected, the relevant Action tabs appear on the top of the pane.

4. Click **Finish** to complete the KPI creation process.

The newly created actions and conditions appear as a list on top of the page. The actions are triggered, when on scheduled interval, the specified conditions are met.

Defining action elements

The Actions interface displays the defined Actions.

Type	Description
	Create a new rule.
	Delete an action.

To define action elements:

1. In the system generated rule name, you can edit the rule name.
2. Select the required condition from the list.
3. Select the required action.

Element	Description
Name	Displays the system generated rule name. You can edit the rule name.
Condition	Displays condition based on business objective. You can select the required condition from the list.
Action(s)	<p>Displays different actions that execute based on a condition. Select the required action(s). The following actions are displayed and selected by default:</p> <ul style="list-style-type: none"> ■ Invoke Business Process: The Invoke Business Process action allows you to trigger a process using a rule. All you need to specify is the name of the process and the message to be sent to the process. For example, a customer has placed an order. When the order enters the CoBOC, a rule is triggered, which checks whether the items exist in the inventory. If the items do not exist, the transaction aborts. If the items exist, the order processing business process is triggered. You can accomplish this by using the Invoke Business Process action. ■ Call Web Service: When a customer places an order, you may want to invoke a Web service or SOAP request to calculate the discount. For this purpose, you can use the Call Web Service action. ■ Notification: The role assigned to the user. A notification is sent to all users having the selected role. Depending on the selected action or actions, corresponding fields appear in the action configuration details group box.

Invoking a business process

Invoke a Business Process as an action if the provided business objectives for the KPI are achieved. The Invoke Business Process action enables you to trigger a process using a rule.

To invoke a business process:

1. On the Actions page, under the Action Properties section, on the Invoke Business Process tab, click  beside Process. The Select Process dialog box opens.
2. *Required.* Select the required process from the displayed available processes. The Business Process is selected.
3. The sample message content is provided if the process model starts with a Message. Provide the appropriate values for the elements, to be considered for the process model, as an input as shown in the following sample.

```
<message>
<root>
  <salesorder>
    <orderitem>
      <itemname>ItemName</itemname>
    </orderitem>
    <orderqty>
      <path><KPIResponseElement>/metricdata/processedvalues/value</path>
    </orderqty>
  </salesorder>
</root>
</message>
```

If the Business Process Model (BPM) expects a message as an input, it is provided by reading the BPM. The KPI response can be sent as input to the BPM as displayed for the `orderqty` element in the sample.

4. Click the **Call Web Service Operation** tab under the Action Properties to define the Call Web Service Operation action. The Call Web Service Operation tab appears in the entry mode.

The Invoke Business Process action is defined.

Calling a web service

Call a Web Service if the specified conditions (business objectives) for the KPI are met.

To call a Web Service:

1. On the Actions page, under Action Properties, on the Call Web Service tab, click  beside **Web Service**. The Select Web service dialog box opens.

2. *Required.* Select the required Web service.
This is a mandatory field. Do not leave it blank. The relevant request is automatically compiled and displayed in the Request field.
3. If you have selected the Notification option under Action Properties, click the **Notification** tab to define the Notification action.
The Notification tab appears in entry mode.

The Call Web Service action is defined.

Defining a notification

Send a Notification when the specified conditions (business objectives) for the KPI are met.

To select the required roles to whom you want to send a notification:

1. On the Actions page, under Action Properties section, click  beside Notify Role(s). The Select Roles dialog box opens displaying all the available roles under Available Roles.
2. Select the role(s) and click  (Add Selected Roles).
The selected roles appear under Selected Roles list.
3. Click **OK**.
The selected roles appear in Notify Role(s) separated by semi-colons.
4. Enter the required URL of the page to be loaded when you open the message from the Inbox, in **URL to Load**.
For example, URL could be <http://www.opentext.com>.
5. *Required.* Enter a **Description** for the URL.
The description text appears as the title of the page as specified in the URL to Load box.
6. Provide a **Subject** of a Process Event.
7. Provide a **Message**.

Message is the data to send to the AppWorks Platform Inbox.

A Notification is defined.

After you complete this task:

- Click **Finish** to complete the KPI creation process.
- The defined actions are triggered when on scheduled interval, the specified conditions are met.

Publishing a KPI

Publishing the KPI ensures to use it as an application at runtime.

Before you begin:

- [Create](#) a KPI.

To publish a KPI:

1. [Access the KPI](#)
2. [Publish it to organization](#)

The KPI is published to organization.

The publishing process of a KPI has two more associated scenarios - [Publishing scenario](#) and [Runtime scenario](#).

Publishing Scenario

After you publish the KPI to runtime, a Web Service is published. The Web Service is created with the KPI name during creation of the KPI.

Runtime Scenario

During runtime, when the Scheduler is invoked:

- The Scheduler calls the Web service being created for the KPI based on the scheduled time.
- A processed value is derived from the measure(s) used in the KPI.
- When the business condition is evaluated, the KPI objects are inserted into a BAM database as XML objects.
- The rules are triggered when they are evaluated against the created KPI object(s).

Unpublishing a KPI

Before you begin:

This task applies only to the published KPIs.

You must unpublish a KPI to stop monitoring a business process or an enterprise data objects. If you unpublish a KPI, it becomes inactive and from then on, the business data is not collected for such KPIs.

To unpublish a KPI:

1. In the Workspace Documents (Explorer) window, right-click the published  (**KPI**) and select **Unpublish**.
2. During unpublish, a Confirm dialog box opens stating "Do you want to retain old data that talks about the business data collected until that time."
 - If you choose to delete or retain the business data, the message stating "KPI Unpublished successfully" appears as a notification.
 - If you choose to retain the business data, the business data is retained.

If a KPI is not published before, Unpublish is triggered. A message stating "KPI is not published" appears.

3. If you select **Yes** or **No**, then the message "KPI Unpublished successfully" appears as a notification.

If the KPI is already unpublished, then the message stating "KPI already Unpublished" appears.

If the selected KPI fails to be unpublished, an exception message appears stating the error.

The Trend Analysis service based on the KPI data still works, and it queries the data accumulated until the unpublished information point.

Editing a KPI

Before you begin:

- [Create](#) a KPI.

You can edit a KPI if you want to modify the created Business Measure, time frame, ranges defined, expression in KPI, name of the KPI, the Web service in KPI, the defined schedule and actions, and email.

To edit a KPI:

1. [Access the KPI](#) and double-click it. Alternatively right-click the KPI and select **Open**.
The KPI is displayed in the KPI editor.
2. Make the appropriate modifications to the KPI and click **Save**.

The modifications made to the KPI are saved.

After you complete this task:

- [Publish](#) the KPI to the organization.

Deleting a KPI

Delete a KPI if it is no longer required.

Consider the following when you delete a KPI:

- If you delete a KPI, the design time is also deleted from the Collaborative Workspace (CWS).
- The delete operation deletes the artifacts, which are created during the creation of the KPI (Composite Control, HTM, Web service, Email Template, Schema fragment).
- If deletion of the selected KPI fails, an exception message is displayed stating the error.

Delete a (KPI) in one of the following ways:

- In the Workspace Documents (Explorer) window, right-click the KPI, and select **Delete**.
- In the Workspace Documents (My Recent Documents) window, select the KPI, click  and select **Delete**.
The Confirm dialog box appears.

Perform one of the following:

- Click **Yes** to delete the KPI. The KPI is deleted from the project content tree.
- Click **No** to cancel the operation.
- Click **Show Used by** to retrieve the details of the documents that use the KPI.

Composing dashboards

A Dashboard equips you with actionable business information in a format that is intuitive and insightful. A dashboard leverages operational data primarily in the form of metrics and KPIs.

BAM Dashboard allows you to analyze the process performance information while monitoring it through the pre-configured Standard Views. Custom views help in monitoring the content of the process and Enterprise Data Object (EDO), which you have defined through Business Measures and KPIs.

The AppWorks Platform BAM provides the following two types of Views:

- [Standard Views](#)
- Custom Views

You can create Custom dashboards using the Dashboard Designer (XForms) with the help of the already available Business Measures and KPIs. Custom Dashboards help you with visualization of the process performance.

You can visualize the non-process data through Dashboard Designer, by creating a Business Measure on the Enterprise Data Object. The Business Measure, KPI, Process Monitoring Object Instances are available as composite controls and you can use these composite controls to create a Dashboard.

To compose a dashboard:

1. Drag the composite control to the XForms designer.
2. Specify properties (on right-click or double-click the composite control) such as view type, time-frame, ranges, and then view it in the dashboard.
3. For further details on creating a User Interface (XForm), see [Creating XForms](#).
4. You can save and assign the created XForms (using composite controls) to a user or role to view it as a dashboard.
5. For further details on assigning the XForm to a user, see Assigning Tasks to Users in the *AppWorks Platform Administrator's Guide*.

For the dashboards to render properly, disable transitional mode. Transitional mode is available as a property for the XForms. The high level tasks that you can perform using AppWorks Platform Dashboard Designer (XForms) are:

1. [Creating Dashboards](#)
2. [Viewing Dashboards using Standard Views](#)

Creating dashboards

The purpose of a dashboard is to equip you with actionable business information in a format that is intuitive and insightful.

Before you begin:

- [Create a Business Measure](#) and a [KPI](#).

To create dashboards:

1. Drag the Business Measure or KPI control to the Dashboard designer (XForms).
2. Right-click or double-click the control to specify properties such as view type, time-frame, ranges, and view it on the dashboard.
3. To define a drill-down between the views, you can use the **Configure** option from the context menu of each view.
4. Save and assign the created XForm to a user or role to view it as a dashboard.

The dashboard is created.

The dashboard can be viewed only if the user or role has access permissions on all the underlying Web services of the Business Measure or KPI objects used in the dashboard. These Web services are available along with the composite controls of the Business Measure or KPI.

You can invoke the method `<controlID>.refresh()` through a script to achieve the Refresh functionality for the Business Measure and KPI controls. This method can be invoked for any available event.

For the modified properties of the Business Measure or KPI to reflect in an already created XForm (dashboard), drag the control again to the XForm and save it.

Drill down between composite controls

On a dashboard (XForm Designer), you can drive from one view to another while making one of the content panes as the Source and other content as the Target.

While creating a dashboard, you can select any number of composite controls content of type Business Measures and define drill-down between them. Map the output parameter of the source view composite control (Business Measure) to the input parameter of the target view composite control (Business Measure or KPI).

The following example explains the drill-down concept.

Example

You have created two Business Measures of type Graph called No. of Orders by Product and Units in Stock for Product. The query is as follows:

1. No. of Orders by Product:

```
Select Count(OrderID) No._Of_Orders, ProductName Group by ProductName
```

2. Units in Stock for Product:

```
Select Avg(UnitsInStock) Units_In_Stock, ProductName Where ProductName = :ProductName Group by ProductName
```

The composite controls for the respective Business Measures are created. The objective is to create a drill-down from No. of Orders by Product to Units in Stock for Product.

1. Drag both the composite controls in a dashboard (XForms) designer and configure drill-down between them.
2. Map the output parameter of Source ProductName to Input parameter ProductName for Target view.
3. You can save and assign the created XForm (using composite controls) to a user/role to view it as a dashboard.
4. Preview the XForm.
In the XForm preview, click on a data point in the view generated for source view (Business Measure).
For example, click the bar if the view type for the content is Bar chart. The target view (Business Measure) appears in the pane already defined for it.
5. If you have created a drill-down for composite controls of multiple business measures, then you can view the target views (of multiple business measures) in the panes already defined for them.
6. For the drill-down between composite controls of composite Business Measures for example, consider Composite Business Measure, CBM1 (constituting Business Measures, BM1 and BM2) as source, and Composite Business Measure, CBM2 (constituting Business Measure, BM3 and BM4) as target to work:

Map the output parameter of the source of BM1 of CBM1 to:	<ul style="list-style-type: none">■ The input parameter of BM3 of CBM2 and■ The input parameter of BM4 of CBM2
Map the output parameter of the source of BM2 of CBM1 to:	<ul style="list-style-type: none">■ The input parameter of BM3 of CBM2 and■ The input parameter of BM4 of CBM2

7. In the XForm preview, open the No. of Orders by Product view.
8. Click the **No. of orders bar** to populate the Units in Stock for Product view that appears in the pane already defined for it.

KPI cannot be a Source but only a Target view.

Creating drill down between composite controls

Creating a drill down between composite controls helps drive from one view to another while making one of the content panes as the Source and other content as the Target.

To create a drill down between composite controls:

1. From the project content tree, drag the composite controls of the respective business measures of type Graph, for which you want to create drill down, on the Dashboard (XForms) designer.
2. Right-click the Business Measure composite control on the XForm Designer from which you want to create drill down and select **Configure Drill Down**.
The Message Map dialog box opens.
3. Map the output parameter of the source view (Business Measure) to the input parameter of the target view (Business Measure).
Refer to using Message Map for more information on using message map.
4. Click **OK**.
 - a. Create drill down for composite controls of multiple business measures of type Graph on similar grounds.
 - b. For further details on creating a User Interface (XForm) see [Creating XForms](#).
 - c. Save and assign the created XForms (using composite controls) to a user or role to view it as a dashboard.
5. Click  (Preview).
The Preview window opens, displaying a preview of the selected XForm.
6. In the XForm preview, click on a data point in the view generated for source view (Business Measure).
For example, click the bar if the view type for the content is Bar chart. This action opens the target view (Business Measure) in the pane already defined.
If you have created drill down for composite controls of multiple business measures, then you can view the target views (of multiple business measures) in the pane already defined for them.
KPI cannot be a Source but only a Target view.

Drill-down between the composite controls is created.

Assigning a time frame for dashboard

Assign a time frame for Dashboard to view the BAM composite controls with different time periods. The Business Measure or the KPI that is created is available as a composite control in the project content tree. For more information on creating a Business Measure and creating a KPI refer to [Creating a Business Measure](#) and [Creating a KPI](#).

To assign a time frame for a Dashboard:

1. Drag a Business Measure or KPI, or Standard View(s) (available as a composite control), onto an XForm Designer.
2. Reuse the TimeFrame composite control. In the Untitled Composite Control - Composite Control window that appears, ensure that you select the TimeFrame composite control from AppWorks Platform Business Activity Monitoring folder. The TimeFrame composite control is added to the project as a reference.

3. Drag the TimeFrame composite control from the project content tree onto the XForm Designer.
4. Right-click or double-click the TimeFrame composite control and specify properties to it. Mapping of Range and Period in the drill-down do not work when you specify the TimeFrame composite control as static. If TimeFrame is rolling, you can map either starttime-endtime or Range-period depending upon the available input parameters for the target. Conversely, if the input parameters in the target are static, you can select the TimeFrame composite control as either static or rolling and map starttime-endtime.
5. Right-click the TimeFrame composite control on the XForm Designer and select **Configure Drill Down**.
The Message Map dialog box opens.
6. Map the output parameter of the Time-frame (staticstarttime and endtime {use this if you expect processed value from Business Measure or KPI based on static time-frame} or rolling - range and period {use this case if you expect processed value from Business Measure or KPI based on rolling time-frame}) to the input parameter of the target view, that is, Business Measures or KPIs time-frame (staticstarttime and endtime or rolling - range and period). Refer to using Message Map to message map the data.
7. Click **OK**.
 - For further details on creating an User Interface (XForm), see [Creating XForms](#).
 - You can save and assign the created XForms (using composite controls) to a user or role to view it as a dashboard.
8. Click  (Preview).
9. In the XForm preview, enter the required period and click  (Refresh).
10. Assign the XForm to a user or role.
11. Open the task and view Dashboard runtime.
12. In Dashboard runtime, the time-frame for the view is first rendered with the time-frame as specified in composite control properties.
13. If you wish to change the time-frame, then specify time period at appropriate place in the dashboard and click  (Refresh).

The time frame is assigned to the Dashboard.

Process monitoring object instances

The purpose of Process Monitoring Object Instances on a dashboard is to do problem analysis of a process, by drilling down to the information at the instance-level. Such analysis helps tracing the exact problem location. The content of the Process Monitoring Object Instances is a set of attributes as defined on a Process Monitoring Object, using which a query is being built to measure an aspect of the process by creating a business measure and the instance related to the same.

For example:

Assume that the Process Monitoring Object created by you contains attributes OrderID, CustomerID, and ProductID for the Order-to-Cash process. The Business Measure defined on this attributes is Count of Orders by ProductID for Customer XYZ Inc. The Process Monitoring Object Instances would then list out the Order instances related to Customer XYZ Inc. as against the whole Instance view as being displayed in the Process Instance View.

Creating a drill down for Process Monitoring object instances

The views help in root-cause analysis, whereby you can point out the bottlenecks, if any, in the process instance. Analysis of the instance-level details provides information such as Status, Duration, and Scheduled to.

Before you begin:

- [Create](#) a Business Measure from which you want to create a drill-down.
- You must have the Process Administrator Role of AppWorks Platform Business Process Engine application to view the Activity View.

You cannot drill down to MonitoringObjectInstances for a Business Measure built on EDO or Web services.

To create a drill down for Process Monitoring Object instances:

1. From the project content tree, drag a Business Measure, which is available as a composite control, on Dashboard (XForms) Designer.
2. [Reuse](#) the ProcessMonitoringObjectInstances, ProcessInstanceGraphicalView, and ProcessInstanceActivities composite controls.
The Untitled Composite Control - Composite Control appears.
3. Ensure that you select the ProcessMonitoringObjectInstances composite control from the AppWorks Platform Business Activity Monitoring folder, and select the ProcessInstanceGraphicalView and ProcessInstanceActivities composite controls from the AppWorks Platform Business Process Engine folder.
The respective composite controls are added to the project content tree.
4. Drag the ProcessMonitoringObjectInstances composite control to the XForms Designer.
5. Right-click the Business Measure for which you want to attach the ProcessMonitoringObjectInstances and select the Configure drill down.
The Message Map dialog box opens.
6. Map the output parameter of the source view (Business Measure selected) to the input parameter of the target view (ProcessMonitoringObjectInstances).
In case of ProcessMonitoringObjectInstances, map the measureid of the Business Measure to the measureid of the ProcessMonitoringObjectinstances.
7. Click **OK**.
See [Using Message Maps](#) for more information on performing parameter mapping.
The parameters of the source view and the target view are mapped.

8. From the project content tree, drag the ProcessInstanceGraphicalView and ProcessInstanceActivities composite controls that you have added as composite controls to the XForm Designer.
9. Right-click the Monitoring View on the XForm Designer, and select Configure Drill Down. The Message Map dialog box opens.
10. Map the INSTANCE_ID of the source view (ProcessMonitoringObjectInstances) to the InstanceID of the target view (ProcessInstanceGraphicalView and ProcessInstanceActivities).
For further details on creating a User Interface (XForm), see [Creating XForms](#).
11. Click  (Preview).
The Preview window opens, displaying a preview of the selected XForm.
12. In the XForm preview, click a data point in the view generated for source view (Business Measure). For example, click the bar graph if the view type for the content is Bar chart. This opens up the target view (ProcessMonitoringObjectInstances).
13. In the XForm preview, click any instance on the ProcessMonitoringObjectInstances. The ProcessInstanceGraphicalView and the ProcessInstanceActivities of the process instance appear as the target view at a specified location in the layout.
14. Save the XForm and assign it to a user or role.
You can access the dashboard created through XForms designer based on your access permissions.

Drill-down for the Process Monitoring Object is created.

Viewing dashboards using standard views

Standard views are pre-configured views. They help you in analyzing the process performance and thus identifying bottlenecks, if any.

To view dashboards using standard views:

1. [Reuse the relevant Standard View composite controls](#). In the Untitled Composite Control - Composite Control that appears, ensure that you select the [Standard View](#) composite control from the AppWorks Platform Business Activity Monitoring folder. You get the Standard Views composite controls.
2. Drag the relevant standard view composite control in a Dashboard (XForms) designer.
3. [Specify properties](#) (on right-click or double-click the composite control) such as view type, ranges, input parameters and then view in the dashboard.
For more information on creating a User Interface (XForm) see [Creating XForms](#).
4. Click .
5. Click  (Preview) to preview the standard view.
6. You can assign the saved XForms to a user or role to view it as a dashboard.

You can achieve Refresh functionality for Standard Views by invoking the method `<controlID>.refresh()` through script. This method can be invoked for any available event.

Activity-level views

The following table describes the activity-level views that you can use to monitor an aspect related to a selected activity of a business process.

Indicator	Description	Corresponding Composite Control
Activity Cycle Time by Period	<p>Displays the average lead time for an activity across completed instances by the specified period.</p> <p>Data related to the period displayed on the graph is grouped in terms of the Groupby period and the cycle time is displayed on the basis of the time you select.</p>	ActivityCycleTime_SpecificActivity
Cycle Time for Activity per User	<p>Displays the average lead time taken by each user to complete a task.</p> <p>Cycle time is displayed based on the selected period unit.</p> <p>Click the table header to sort the table view.</p>	ActivityCycleTimePerUser_SpecificActivity
No. of Activity Instances per Status	<p>Displays a high level summarized view of the number of activity instances, based on status, for a specified period.</p> <p>Displays 'Aborted', 'Waiting', 'Completed', 'Suspended' and 'Terminated' statuses for all instances for the selected activity.</p>	ActivityInstancesPerStatus_SpecificActivity
No. of Tasks per User by Status	<p>Displays the number of tasks for each user per status.</p> <p>Displays the number of instances under 'Aborted', 'Waiting', 'Complete', 'Suspended' and 'Terminated' statuses for the selected activity.</p> <p>Click the table header to sort the table view.</p> <p>On the dashboard, if you, as a Manager, want to monitor the progress of tasks in the worklists, and if you want to have an</p>	ActivityTasksPerUserPerStatus_SpecificActivity

Indicator	Description	Corresponding Composite Control
	overview of the pending tasks, you can drill down to the Inbox view and complete the pending tasks. See Creating a drill-down for Inbox	

The time values displayed will be based on the time zone of the client browser in a 12-hour format.

Configuring drill down from activity-level views

- The composite controls found at the location /Cordys/WCP/Controls/runtime/bam/standardviews/ have been deprecated. Open Text recommends using the composite controls provided in the corresponding sub-folders at /Cordys/WCP/Controls/runtime/bam/standardviews/ with same name as the deprecated controls.
- Similarly, the ActivitySelection composite control available at the location /Cordys/WCP/Controls/runtime/bam/genericviews/ has also been deprecated. Open Text recommends using the composite control available at /Cordys/WCP/Controls/runtime/bam/genericviews/activitydashboard/ with the same name.

Configure a drill-down from Activity-Level views to drive from one view to another while making one of the content panes as the Source and other as the Target.

To configure drill down from Activity-Level views:

1. All the Activity-Level Views can be reused as composite controls. In the Untitled Composite Control - Composite Control that appears, ensure that you select the composite controls from Cordys Business Activity Monitoring folder.
2. You can define a drill-down from the ActivitySelection composite control, available at /Cordys/WCP/Controls/runtime/bam/genericviews/activitydashboard/, to the composite control related to specific Activity-Level views to get the details of a particular activity. Consider the ActivityTasksPerUserPerStatus_SpecificActivity composite control, available at /Cordys/WCP/Controls/runtime/bam/standardviews/activitydashboard/ as an example of Activity-Level views. Perform the following to define drill-down:
 - a. Reuse the ActivitySelection and ActivityTasksPerUserPerStatus_SpecificActivity composite controls. The corresponding composite controls are added to the project content tree.
 - b. From the project content tree, drag the ActivitySelection and ActivityTasksPerUserPerStatus_SpecificActivity composite controls on to the XForm designer. Consider the ActivitySelection composite control as Source and ActivityTasksPerUserPerStatus_SpecificActivity composite control as Target.
 - c. Right-click the ActivitySelection composite control on the XForm designer and select Configure Drill Down. The Message Map dialog box opens.

- d. Map the input parameters of the Source, `processName` and `activityID` to input parameters, `ProcessName` (`InputParameters > ProcessName`) and `ActivityID` (`InputParameters > ActivityID`) of the Target.
 - e. Click **OK**.
The Message Map dialog box closes.
 - f. Click  (Preview). The Preview window appears, displaying a preview of the selected XForm. All Activity-Level views for the selected process and activity appear.
3. In the XForm preview, you can change the Activity-Level views by performing the following steps:
- a. Select required process name from Process.
 - b. Select required activity name from Activity.
 - c. Click  (Refresh).
All the Activity-Level views for the selected process and activity appear.
4. You can define a drill-down from the Activity-Level View to the User-Level Views. Consider `ActivityTasksPerUserPerStatus_SpecificActivity` composite control related to Activity-Level View and `CompletedVsScheduledTasksForActivity_User` composite control related to User-Level View.
Perform the following to define the drill-down:
- a. As runtime reference, add `ActivityTasksPerUserPerStatus_SpecificActivity`, `InboxView` composite controls, and `CompletedVsScheduledTasksForActivity_User` composite controls, as examples of the User-Level Views. Ensure that you select the `ActivityTasksPerUserPerStatus_SpecificActivity`, available at `/Cordys/WCP/Controls/runtime/bam/standardviews/activitydashboard/`, `CompletedVsScheduledTasksForActivity_User`, available at `/Cordys/WCP/Controls/runtime/bam/standardviews/userdashboard/` and the composite controls from the Cordys Business Activity Monitoring folder, and the `InboxView` composite control from the Cordys Notification folder. The corresponding composite controls are added to the project content tree.
 - b. From the project content tree, drag the `ActivityTasksPerUserPerStatus_SpecificActivity`, `CompletedVsScheduledTasksForActivity_User` and `InboxView` composite controls on the XForm designer. Consider `ActivityTasksPerUserPerStatus_SpecificActivity` composite control as Source and the `CompletedVsScheduledTasksForActivity_User` and `InboxView` composite controls as Targets.
 - c. Right-click the `ActivityTasksPerUserPerStatus_SpecificActivity` composite control on the XForm designer and select Configure drill down. The Message Map dialog box opens.
 - d. Map the parameters of the Source, `ProcessName` (`InputParmaters > ProcessName`) and `ActivityID` (`InputParmaters > ActivityID`) to the input parameters `ProcessName` (`InputParmaters > ProcessName`) and `ActivityID` (`InputParmaters > ActivityID`) of the Target (`CompletedVsScheduledTasksForActivity_User` composite control).

- e. Map the output parameter Name (ActivityData > tuple > old > GetNumberOfTasksForActivityForStatus > Name) of the Source to the input parameter User (InputParameters > User) of the Target (CompletedVsScheduledTasksForActivity_User composite control).
 - f. Map the output parameter Name (ActivityData > tuple > old > GetNumberOfTasksForActivityForStatus > Name) of the Source to input parameter value (Criteria > Parameters > Parameter > value) of the Target view (InboxView composite control). For the input parameter, type of the Target (InboxView composite control), provide a fixed value, **user**.
 - g. Click **OK**.
The Message Map dialog box closes.
 - h. Click  (Preview).
The Preview window opens, displaying a preview of the selected XForm.
 - i. In the XForm preview, to drill down to the User-Level Views, click the required data-point on **No. of Tasks for Activity per Status** view.
5. Similarly, you can add TimeFrame composite control as a runtime reference and use it as a Source. You can map the output parameters, (startTime and endTime) of the TimeFrame composite control to input parameters of the Target (FromDate andToDate), any of the composite controls related to Activity-Level views.

For more details on drilling down to Inbox, see [Drill down to Inbox view](#).

The drill-down from the Activity-Level views are configured.

Creating a drill-down for Inbox

If you, as a Manager, want to monitor the state of tasks in the worklist(s) in the dashboard, the state of tasks in the worklist(s), and get an overview of the pending tasks, you can create drill-down view of Inbox.

- Create a dashboard (XForm) and drag drop the composite control of the Standard View - **ActivityTasksPerUserPerStatus_SpecificActivity** view composite control to it.
- Drag and drop the composite control of the **Inbox** view in the same Dashboard. Wire both of composite controls on the dashboard (create message map) and visualize the dashboard.
- You can drill-down to Inbox by clicking a data-point in the **No. of Tasks per User by Status** view in the dashboard, the Inbox appears with all the pending tasks in the pane already defined for it and complete the pending tasks.

To create a drill-down for Inbox:

1. Drag the composite control of Standard View - **ActivityTasksPerUserPerStatus_SpecificActivity** (view name - **No. of Tasks per User by Status**) on dashboard designer (XForms).
2. Drag the **Inbox** view that is available as a composite control in the same dashboard designer (XForms).

3. Right-click the Standard View composite control on the XForm Designer from which you want to create drill down and select Configure Drill Down. The **Message Map** dialog box appears. Map Name of source view to Target of the target view. Click **OK**.
 - For further details on creating a User Interface (XForm) see [Creating XForms](#).
 - Save and assign the created XForms (using composite controls) to a user or role to view it as a dashboard.

 4. Click  (Preview).

The Preview window opens, displaying a preview of the selected XForm.

 5. In the XForm preview, click a data point in the view generated for source view (User). For example, click the bar if the view type for the content is 'Bar chart'. The target view (Inbox) appears in the pane already defined for it.
- For further details on using an Inbox refer [Using Inbox as a Composite Control](#).

Drill-down to Inbox composite control is created.

Process-level views

The following are the process-level views:

View	Description	Corresponding Composite Control
Process Load for Period	This view displays the number of started instances for a specific process over a specified period. The data displayed in the graph is grouped in terms of the Groupby period you select. For example, if you select 'Days' as the Groupby period, the graph will group and display the data on a daily basis.	ProcessLoadForPeriod_Specific Process
Process Cycle time by Period	This displays a view on the average lead time for a selected process across completed instances by a period. The data related to the period displayed in the graph is grouped in terms of the Groupby period and the cycle time is displayed on the basis of the time you select.	ProcessCycleTime_SpecificProcess
Process Instances per Status	Provides a high level summarized view of the number of process instances based on status for a selected process over a period. This view displays the number of	ProcessInstancesPerStatus_SpecificProcess

View	Description	Corresponding Composite Control
	instances under Aborted, Waiting, Completed, Suspended and Terminated statuses for the selected process.	
No. of Activity Instances per Status	For a selected process, displays the number of activity instances per status for a specific period. You can click the table header to sort the table view.	ActivityInstancesPerStatus_SpecificActivity
Cycle time per Activity	Displays a view on the average lead time for each activity across completed instances for a selected process for a specified period. You can click the table header to sort the table view.	ProcessActivitiesCycleTime_SpecificProcess

The time values displayed will be based on the time zone of the client browser in a 12-hour format.

Configuring drill down from process-level views

- The ProcessInstancesPerStatus_SpecificProcess composite control available at the location `/Cordys/WCP/Controls/runtime/bam/standardviews/` has been deprecated. Instead, it is recommended to use the composite control available at `/Cordys/WCP/Controls/runtime/bam/standardviews/processdashboard/` with the same name.
- The ProcessSelection composite control available at the location `/Cordys/WCP/Controls/runtime/bam/genericviews/` has also been deprecated. Hence, it is recommended to use the composite control available at `/Cordys/WCP/Controls/runtime/bam/genericviews/processdashboard/` with the same name.

Configure a drill-down from Process-Level Views to drive from one view to another while making one of the content panes as the Source and other as the Target.

To configure drill down from Process-Level views:

1. All the Process-Level Views can be reused as composite controls. In the Untitled Composite Control - Composite Control that appears, ensure that you select the composite controls from AppWorks Platform Business Activity Monitoring folder.
2. You can use the ProcessSelection composite control, available at `/Cordys/WCP/Controls/runtime/bam/genericviews/processdashboard/`, to get a list of the processes to drill-down to specific Process-Level Views. Consider the ProcessInstancesPerStatus_SpecificProcess composite control, available at

/Cordys/WCP/Controls/runtime/bam/standardviews/processdashboard/ as an example of specific Process-Level Views, as Target.

3. Perform the following to define drill-down:
 - a. From the project content tree, reuse the ProcessSelection and ProcessInstancesPerStatus_SpecificProcess composite controls. Consider the ProcessSelection composite control as Source and the ProcessInstancesPerStatus_SpecificProcess composite control as Target.
 - b. Drag the ProcessSelection and the ProcessInstancesPerStatus_SpecificProcess composite controls on to the XForm designer.
 - c. Right-click the ProcessSelection composite control on the XForm designer and select Configure Drill Down. The Message Map dialog box opens.
 - d. Map the processName of the Source to input parameter ProcessName (InputParameters > ProcessName) of the Target.
 - e. Click **OK**.
The Message Map dialog box closes.
4. Click  (Preview).
The Preview window opens.
 - a. Select the required process name from Process.
 - b. Select a required time frame and period.
 - c. Click  (Refresh).
All the Process-Level Views for the selected process appear.

The drill-down from the process-level views are configured.

Standard views

AppWorks Platform BAM provides the following Standard Views that are available as composite controls and can be **reused as composite controls**:

- Operational-level Views: Aids in monitoring performance of all processes in an organization.
- **Process-level Views**: Aids in monitoring performance of a specific process in an organization.
- **Activity-level Views**: Aids in monitoring performance of a specific activity of a process in an organization.
- **User-level Views**: Aids in monitoring performance of a user to whom the activity is assigned.
- Instance-level Views: Aids in monitoring performance of a specific instance of a process in an organization.

Configuring drill down from operational-level views

Use the composite controls provided in the corresponding sub-folders at /Cordys/WCP/Controls/runtime/bam/standardviews/.

Configure a drill-down from Operational-Level views to drive from one view to another while making one of the content panes as the Source and other content as the Target.

To configure drill down from operational-level views:

1. All the Operational-Level Views can be reused as composite controls. In the Untitled Composite Control - Composite Control that appears, make sure that you select the composite controls from AppWorks Platform Business Activity Monitoring folder.
2. You can define drill-down from Top X Processes for Status view using `TopXInstancesPerStatus_AcrossProcess` (available at `/Cordys/WCP/Controls/runtime/bam/standardviews/operationaldashboard/`) composite control to all corresponding Process-level view composite control. Consider the `TopXInstancesPerStatus_AcrossProcess` composite control as Source and `ProcessInstancesPerStatus_SpecificProcess` (available at `/Cordys/WCP/Controls/runtime/bam/standardviews/processdashboard/`) composite control as Target as an example of a drill down scenario from operational-view to the process-level view. Perform the following to define such a drill-down:
 - a. In the project content tree, reuse the `TopXInstancesPerStatus_AcrossProcess` composite control and `ProcessInstancesPerStatus_SpecificProcess` composite control, related to process_level view. The respective composite controls are added to the project content tree.
 - b. From the project content tree, drag the `TopXInstancesPerStatus_AcrossProcess` and `ProcessInstancesPerStatus_SpecificProcess` composite controls on the XForm designer.
 - c. Right-click the `TopXInstancesPerStatus_AcrossProcess` composite control on the XForm designer and select the **Configure** drill down. The Message Map dialog box appears.
 - d. Map the output parameter `Name` (`OperationalData > tuple > old > GetTopXProcessesByStatus > Name`) of the Source to the input parameter, `ProcessName` (`InputParameters > ProcessName`) of the Target.
 - e. Click **OK**. The Message Map dialog box closes.
 - f. Click  (Preview). The Preview window appears, displaying a preview of the selected XForm.
 - g. Similarly you can add `TimeFrame` composite control as runtime reference and use it as a Source. The output parameters (`startTime` and `endTime`) of `TimeFrame` composite control can be mapped to input of target (`FromDate` and `ToDate`), any of the composite controls related to operational-level views.

The drill-down from the operational-level views are configured.

Configuring drill down from instance-level views

- The `TopXInstancesByLeadTimeForStatus_SpecificProcess` composite control available at the location `/Cordys/WCP/Controls/runtime/bam/standardviews/` has been deprecated. Instead; it is recommended to use the composite control available at `/Cordys/WCP/Controls/runtime/bam/standardviews/instancesdashboard/` with the

same name.

- Similarly, the ProcessSelection composite control available at the location `/Cordys/WCP/Controls/runtime/bam/genericviews/` has also been deprecated. Therefore, it is recommended to use the composite control available at `/Cordys/WCP/Controls/runtime/bam/genericviews/processdashboard/` with the same name.

Configure a drill-down from the Instance-Level Views to drive from one view to another while making one of the content panes as the source and the other as the target.

To configure drill down from Instance-Level views:

1. The Instance-Level Views can be reused as composite controls. In the Untitled Composite Control - Composite Control that appears, ensure that you select the composite controls from the AppWorks Platform Business Activity Monitoring folder.
2. You can define drill-down from Instance-Level View to the Graphical view of the process. Perform the following to define the drill-down:
 - a. From the project content tree, reuse the **TopXInstancesByLeadTimeForStatus_SpecificProcess** available at `/Cordys/WCP/Controls/runtime/bam/standardviews/instancesdashboard/`, **ProcessInstanceGraphicalView**, the **ProcessInstanceActivities** and the **ProcessInstances** composite controls. Ensure that you select the TopXInstancesByLeadTimeForStatus_SpecificProcess composite control from the AppWorks Platform Business Activity Monitoring folder, and select the ProcessInstanceGraphicalView, ProcessInstanceActivities, ProcessInstances composite controls from the AppWorks Platform Business Process Engine folder. The respective composite controls are added to the project content tree.
 - b. Drag the **TopXInstancesByLeadTimeForStatus_SpecificProcess** and the **ProcessInstanceGraphicalView** composite controls on the XForms designer.
 - c. Right-click the **TopXInstancesByLeadTimeForStatus_SpecificProcess** composite control on the XForms designer and select **Configure Drill Down**. The Message Map dialog box opens.
 - d. Map the output parameter **Name**, `InstanceData > tuple > old > GetProcessLoadHistory > Name` of the source to the input parameter `instanceID` of the target.
 - e. Click **OK**.
The Message Map dialog box closes.
 - f. Click  (Preview).
The Preview window opens, displaying a preview of the selected XForm.
 - g. In the XForm preview, click the required data point on the instance bar containing the `InstanceId` to retrieve the graphical view of the process. The graphical view appears, displaying the actual state of a process instance. The actual activity is highlighted to make its appearance in bold, and the execution path is displayed in a dotted line.

3. You can define a drill-down from the Instance-Level View to the Process-Instance view. Perform the following to define the drill-down:
 - a. Drag the **TopXInstancesByLeadTimeForStatus_SpecificProcess** and the **ProcessInstanceActivities** composite controls on the XForms designer. Consider the TopXInstancesByLeadTimeForStatus_SpecificProcess composite control as the source and the ProcessInstanceActivities composite control as the target.
 - b. Right-click the **TopXInstancesByLeadTimeForStatus_SpecificProcess** composite control on the XForms designer and select **Configure Drill Down**. The Message Map dialog box opens.
 - c. Map the output parameter **Name**, that is `InstanceData > tuple > old > GetProcessLoadHistory > Name` of the source to the input parameter `instanceID` of the target.
 - d. Click **OK**.
The Message Map dialog box closes.
 - e. Click  (Preview).
The Preview window opens. In the XForm preview, you can view details about different activities of the process instance.
4. You can drill-down from an Instance-Level View to a Process-Instance view. Perform the following to define the drill-down:
 - a. Drag the **TopXInstancesByLeadTimeForStatus_SpecificProcess** composite control and the **ProcessInstances** composite control, which is related to the process-level view. Consider the TopXInstancesByLeadTimeForStatus_SpecificProcess composite control as the source and the ProcessInstances composite control as the target.
 - b. Right-click the **TopXInstancesByLeadTimeForStatus_SpecificProcess** composite control on the XForms designer, and select **Configure Drill Down**. The Message Map dialog box opens.
 - c. Map the input parameters, **ProcessName** and **Status of the Source** to input parameters, **processName** and **status** of the Target respectively.
 - d. Click **OK**.
The Message Map dialog box closes.
 - e. Click  (Preview). The Preview window opens. To retrieve the process instance view, select status in an instance, and click the required data-point on the instance bar. Based on the selected instance status, the filtered view of process instances is displayed.
 - f. If you are viewing the Top X instances by lead-time for Waiting status, then click the process instance view link.
The process instance view appears, displaying the instances that have only the Waiting status.
5. You can drill-down from the ProcessSelection composite control, available at `/Cordys/WCP/Controls/runtime/bam/genericviews/processdashboard/` to specific process-level views to retrieve a list of processes.
Perform the following to define the drill-down:

- a. Drag the ProcessSelection and the TopXInstancesByLeadTimeForStatus_SpecificProcess composite controls on the XForms designer. Consider the ProcessSelection composite control as the source and the TopXInstancesByLeadTimeForStatus_SpecificProcess composite control as the target.
- b. Right-click the **ProcessSelection** composite control on the XForms designer, and select **Configure Drill Down**.
The Message Map dialog box opens.
- c. Map the **processName** of the source to the input parameter **ProcessName**, that is `InputParameters > ProcessName`, of the target.
- d. Click **OK**.
The Message Map dialog box closes.
- e. Click  (Preview).
The Preview window opens:
 - i. Select the required process name from **Process**.
 - ii. Select the required timeframe and period.
 - iii. Click  (Refresh). All the process-level views for the selected process are displayed.

Similarly, you can add the TimeFrame composite control as a run-time reference and can use it as a source. You can map the output parameters (startTime and endTime) of the TimeFrame composite control to input parameters of the target (FromDate andToDate) or any of the composite controls related to instance-level views.

The drill-down from instance-level views are configured.

Standard views composite control properties interface

The Business Measure Composite Control Properties Interface helps in configuring properties of composite control.

The fields on the Properties Interface of the ActivityCycleTime_SpecificActivity Composite Control are as follows.

Activity Name	Click  , and select the activity.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Period In	Select one of the following: Second(s), Minute(s), Hour(s), Day (s), Week(s), Month(s), Year(s).
Process Name	Click  , and select the business process.
Render View on load	Select if you want to render the view while the page loads. The

check box	check box is selected by default.
Show Header	Select the check box to give inputs at run-time.
Time Frame group box	Enter time period in the Last text box and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year (s).
Title	Activity Cycle Time appears by default.
X-Axis Caption	Period is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. If enough space is not available on the chart, the text of the labels will be trimmed and an ellipses (...) is added at the end and would show tool-tips for those labels. ■ The X-Axis Caption Style is not available for Pyramid and Pie charts. ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Cycle Time is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.

The fields on the Properties Interface of the ActivityCycleTimePerUser_SpecificActivity Composite Control are as follows.

Title	Cycle Time for Activity per User appears by default.
Period In	Select one of the following: Second(s), Minute(s), Hour(s), Day (s), Week(s), Month(s), Year(s).
Show Header check box	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
X-Axis Caption	User is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.

X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label <p>■ The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	Cycle Time is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.
Activity Name	Click  and select the activity.

The fields on the Properties Interface of the ActivityInstancesPerStatus_SpecificActivity Composite Control are as follows.

Title	Number of Activity Instances per Status appears by default.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
X-Axis Caption	Status is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label <p>■ The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>■ If the x-axis field value is lengthy, select Truncate or Wrap</p>

	to get a proper graph without any distortion.
Y-Axis Caption	No Of Instances is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last text box and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.
Activity Name	Click  and select the activity for which you want to display activity instances per status.

The fields on the Properties Interface of the ActivityTasksPerUserPerStatus_SpecificActivity Composite Control are as follows.

Title	No. of Tasks per User by Status appears by default.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Column 2D, Line 2D, Bar 2D.
X-Axis Caption	User is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label The X-Axis Caption Style is not available for Pyramid and Pie charts. <ul style="list-style-type: none"> ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	No. Of Tasks is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).

Process Name	Click  and select the business process.
Activity Name	Click  and select the activity for which you want to display activity instances per status.

The fields on the Properties Interface of CompletedVsScheduledTasksAcrossProcess_User Composite Control are as follows.

Title	Completed Vs Scheduled Tasks for User across Processes appears by default.
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Column 2D, Line 2D, Bar 2D.
X-Axis Caption	Period is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label The X-Axis Caption Style is not available for Pyramid and Pie charts. <ul style="list-style-type: none"> ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Number of Tasks is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
User	Click and select the user.

The fields on the Properties Interface of the CompletedVsScheduledTasksForActivity_User Composite Control are as follows.

Title	Completed Vs Scheduled Tasks for User for Activity appears by
-------	---

	default.
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Column 2D, Line 2D, Bar 2D.
X-Axis Caption	Period is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. <p>The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	Number of Tasks is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. Selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.
Activity Name	Click  and select the activity.
User	Click  and select the user.

The fields on the Properties Interface of ProcessActivitiesCycleTime_SpecificProcess Composite Control are as follows.

Title	Cycle time per Activity appears by default.
Period In	Select one of the following: Second(s), Minute(s), Hour(s), Day (s), Week(s), Month(s), Year(s).

Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
X-Axis Caption	Activity Name is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. ■ The X-Axis Caption Style is not available for Pyramid and Pie charts. ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Cycle Time is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.

The fields on the Properties Interface of ProcessActivitiesInstancesPerStatus_SpecificProcess Composite Control are as follows.

Title	No. of Instances per Activity by Status appears by default.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Column 2D, Line 2D, Bar 2D.
X-Axis Caption	Activity Name is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following:

	<ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. ■ The X-Axis Caption Style is not available for Pyramid and Pie charts. ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Number of Instances is displayed by default. You can edit them if required.
Render View on load check box	The Y-Axis Caption is not available for Pyramid and Pie charts.
Time Frame group box	Select if you want to render the view while the page loads. The check box is selected by default.
Process Name	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).

The fields on the Properties Interface of the ProcessCycleTime_SpecificProcess Composite Control are as follows.

Title	Process Cycle Time appears by default.
Period In	Select one of the following: Second(s), Minute(s), Hour(s), Day(s), Week(s), Month(s), Year(s).
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart.
X-Axis Caption	Period is displayed by default. Specify x-axis caption. You can edit them if required.
	The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically.

	<p>vertically.</p> <ul style="list-style-type: none"> ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. ■ The X-Axis Caption Style is not available for Pyramid and Pie charts. ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Cycle Time is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.

The fields on the Properties Interface of the ProcessInstancesPerStatus_AcrossProcess Composite Control are as follows.

Title	Aggregate Process Instances per Status appears by default.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart.
X-Axis Caption	Status is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. ■ The X-Axis Caption Style is not available for Pyramid and

	<p>Pie charts.</p> <ul style="list-style-type: none"> ▪ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	<p>No Of Instances is displayed by default. You can edit them if required.</p> <p>The Y-Axis Caption is not available for Pyramid and Pie charts.</p>
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).

The fields on the Properties Interface of the ProcessInstancesPerStatus_SpecificProcess Composite Control are as follows.

Title	Process Instances per Status appears by default.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart.
X-Axis Caption	Status is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	<p>Select one of the following:</p> <ul style="list-style-type: none"> ▪ Truncate: To display the truncated label ▪ Rotate: To display the Rotated label. Displays the labels vertically. ▪ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ▪ Stagger: To display the Stagger label. Displays the labels in multiple lines. ▪ Wrap: To display the Wrapped label. <p>▪ The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>▪ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	<p>Number of Instances is displayed by default. You can edit them if required.</p> <p>The Y-Axis Caption is not available for Pyramid and Pie charts.</p>
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.

Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.

The fields on the Properties Interface of the ProcessLoadForPeriod_AcrossProcess Composite Control are as follows.

Title	Process Load for Period appears by default.
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
X-Axis Caption	Period is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. ■ The X-Axis Caption Style is not available for Pyramid and Pie charts. ■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.
Y-Axis Caption	Cycle Time is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).

The fields on the Properties Interface of the ProcessLoadforPeriod_SpecificProcess Composite Control are as follows.

Title	Process Load for Period appears by default.
Group By	Select one of the following: Hour(s), Day(s), Week(s), Month(s), Year(s).
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart. Pie Chart appears by default.
X-Axis Caption	Period is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Rotate: To display the Rotated label. Displays the labels vertically. ■ Slant: To display the label in Slant. Displays the labels at 45 degrees angle. ■ Stagger: To display the Stagger label. Displays the labels in multiple lines. ■ Wrap: To display the Wrapped label. <p>The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	Number of Instances is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.

The fields on the Properties Interface of the TopXInstancesByLeadTimeForStatus_SpecificProcess Composite Control are as follows.

Title	GetTopXInstancesByLeadTimeForStatus appears by default.
No Of Instances	Enter the number of instances.
Period In	Select one of the following: Second(s), Minute(s), Hour(s), Day(s), Week(s), Month(s), Year(s).
Status	Select the status of the business process for which the Top X

	Instances by Lead time for Status are displayed, from one of the following: Aborted, Completed, Suspended, Terminated, Waiting.
Show Header	Select the check box to give inputs at run-time.
Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart.
X-Axis Caption	Instance ID is displayed by default. You can edit them if required.
X-Axis Caption Style	<p>The X-Axis Caption is not available for Pyramid and Pie charts.</p> <p>Select one of the following:</p> <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label <p>The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	<p>Lead Time is displayed by default. You can edit them if required.</p> <p>The Y-Axis Caption is not available for Pyramid and Pie charts.</p>
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).
Process Name	Click  and select the business process.

The fields on the the Properties Interface of the TopXInstancesPerStatus_AcrossProcess Composite Control are as follows.

Title	GetTopXProcessesByStatus appears by default.
No Of Processes	Enter the number of processes for which you want to display Top X Instances per status across process.
Status	Select the status of the business process for which the Top X Instances by Lead time for Status are displayed, from one of the following: Aborted, Completed, Suspended, Terminated, Waiting.
Show Header	Select the check box to give inputs at run-time.

Chart Type	Specify one of the following view type: Line Chart, Bar Chart, Pyramid, Pie Chart.
X-Axis Caption	Process Name is displayed by default. You can edit them if required. The X-Axis Caption is not available for Pyramid and Pie charts.
X-Axis Caption Style	Select one of the following: <ul style="list-style-type: none"> ■ Truncate: To display the truncated label ■ Slant: To display the label in Slant ■ Stagger: To display the Stagger label ■ Rotate: To display the Rotated label ■ Wrap: To display the Wrapped label <p>■ The X-Axis Caption Style is not available for Pyramid and Pie charts.</p> <p>■ If the x-axis field value is lengthy, select Truncate or Wrap to get a proper graph without any distortion.</p>
Y-Axis Caption	Number of Instances is displayed by default. You can edit them if required. The Y-Axis Caption is not available for Pyramid and Pie charts.
Render View on load check box	Select if you want to render the view while the page loads. The check box is selected by default.
Time Frame group box	Enter time period in Last and select one of the following unit of time: Hour(s), Day(s), Week(s), Month(s), Year(s).

The APIs for the Standard Views Composite Control are as follows.

getChartObject	This API returns the Chart object, on which the APIs of Chart can be invoked
refresh	Use the API to redraw the chart.
setAsynchronous(String)	Value can be "true" or "false". By default, the request is sent in asynchronous mode. To change the mode to synchronous, invoke the API with value "false" and invoke refresh API to redraw the chart.

The events for the Standard Views Composite Control are as follows.

ondataclick	Fires when clicked on a data point on the chart. Refer to the following sample code. <pre>function Form_InitDone(eventObject) { <svid>.addListener(ondataclicklistenerFn); }</pre>
-------------	---

```
function ondataclicklistenerFn(eventObject)
{
//eventObject.businessObject will give the businessobject as
a JSON object for the clicked datapoint
}
```

User level views

The following table describes the functionality of each user-level view that you can use to monitor an aspect related to an assigned user.

View	Description	Corresponding Composite Control
Completed Vs Scheduled Tasks for User for Activity	Provides a view of the workload for a selected user through comparison of the number of completed tasks as against the number of scheduled tasks along with the task backlog, for a specific activity and for the specified period in a graphical format.	CompletedVsScheduledTasksForActivity_User
Completed Vs Scheduled Tasks for User across Processes	Provides a view of the number of completed tasks as against the number of scheduled tasks for a selected user along with the task backlog, across processes, for the specified period in a graphical format.	CompletedVsScheduledTasksAcrossProcess_User

The time values displayed will be based on the time zone of the client browser in a 12-hour format.

Configuring drill down from user-level views

- The CompletedVsScheduledTasksForActivity_User composite control available at the location /Cordys/WCP/Controls/runtime/bam/standardviews/ has been deprecated. Instead, it is recommended to use the composite control available at /Cordys/WCP/Controls/runtime/bam/standardviews/userdashboard/ with the same name.
- Similarly, the UserSelection composite control available at the location /Cordys/WCP/Controls/runtime/bam/genericviews/ has also been deprecated. Hence, it is recommended to use the composite control available at /Cordys/WCP/Controls/runtime/bam/genericviews/userdashboard/ with the same name.

Configure a drill-down from the User-Level Views to drive from one view to another while making one of the content panes as the Source and the other as Target.

To configure drill down from User-Level views:

1. All the [User-Level Views](#) can be [reused as composite controls](#). In the Untitled Composite Control - Composite Control, make sure that you select the the composite controls from AppWorks Platform Business Activity Monitoring folder.
2. You can use the UserSelection composite control, available at `/Cordys/WCP/Controls/runtime/bam/genericviews/userdashboard/`, to get the list of users to drill-down to specific User-Level Views. Consider the CompletedVsScheduledTasksForActivity_User composite control, available at `/Cordys/WCP/Controls/runtime/bam/standardviews/userdashboard/`, as an example of the User-Level Views. Perform the following to define drill-down:
 - a. In the project content tree, [reuse](#) the UserSelection composite control and CompletedVsScheduledTasksForActivity_User composite controls. The respective composite controls are added to the project content tree.
 - b. From the project content tree, drag the UserSelection and CompletedVsScheduledTasksForActivity_User composite controls on the XForm designer. Consider UserSelection composite control as Source and the CompletedVsScheduledTasksForActivity_User composite control as Target.
 - c. Right-click the **UserSelection** composite control on the XForm designer and select Configure Drill Down. The Message Map dialog box appears.
 - d. Map the parameters of the Source, processName, activityID and userName to input parameters of the Target, ProcessName (InputParameters > ProcessName), ActivityID (InputParameters > ActivityID) and User (InputParameters > User).
 - e. Click **OK**.
The Message Map dialog box closes.
 - f. Click  (**Preview**).
The Preview window opens, displaying a preview of the selected XForm.
3. In the XForm preview, you can change the user-level views for the selected user by performing the following steps:
 - a. Click  corresponding to Select Process.
The Select Business Process dialog box opens.
 - b. Select the required process name and click **OK**.
 - c. Click  corresponding to Select Activity.
The Select Activity dialog box opens.
 - d. Select the required activity name and click **OK**.
 - e. Click  corresponding to Select User.
The Select User dialog box opens.
 - f. From User, select the required user name.

Similarly, you can add a TimeFrame composite control as the runtime reference and can use it as a Source. You can map the output parameters, (startTime and endTime) of TimeFrame composite control to the input parameters of the Target (FromDate andToDate) or any of the composite controls corresponding to user-level views.

Business measure composite control properties interface

The Business Measure Composite Control Properties Interface helps in configuring properties of composite control.

The fields on the Properties Interface of Business Measure Composite Control for Graph are as follows.

Title	Specify chart title. Title property is not applicable for Angular gauge and Linear gauge. You may use XForms label for the same.
View	<p>Specify one of the following view types:</p> <ul style="list-style-type: none"> ■ Line Chart: To display the view as Line chart ■ Bar Chart: To display the view as Bar chart ■ Pyramid: To display the view as Pyramid ■ Pie Chart: To display the view as Pie2D ■ Chart Stacked Bar Chart: To display the view as Stacked Bar Chart. A Stacked Bar Chart displays the results of multiple queries stacked on top of one another and usually used for showing relative quantity with respect to whole. For example: In an Order-to-Cash process, you would like to visualize Units in Stock and Units on Order for a Product. You can make a Business Measure with the following query: Select Avg(UnitsInStock) In Stock, Avg(UnitsOnOrder) OnOrder From Orders Group by ProductID. For visualizing the same you can make use of view type Stacked Bar Chart and the resultant view will show Units In Stock and Units on Order stacked on top of one another. ■ Combinational Chart: Display the view as Combinational Chart. Combinational chart displays different plots on the same y-axis. ■ Combinational Dual-Y Chart: Display the view as Combinational Dual Y Chart. Combinational Dual Y chart displays different plots on the two y-axes. <p>If you select Combinational Dual-Y Chart, then the Secondary Y-Axis Caption text box appears under Y-Axis Caption. Enter a caption for the Secondary Y-Axis.</p> <p>Line Chart appears by default.</p>
Render View on load	Render the view while the page loads.
Chart Details > Select X-Axis Fields	<p>Click . The Select X-Axis Fields dialog box opens. See Select X-Axis Field Interface for information on completing the fields. Select an X-axis field and click OK.</p> <ul style="list-style-type: none"> ■ If you want to view a Business Measure having multiple Group by clause, select multiple X-axis fields of the Business Measure and

	<p>visualize it in the Dashboard. The selected X-axis field name(s) appear in text box next to Chart Details > X-Axis Caption.</p> <ul style="list-style-type: none"> ■ You can select multiple X-axis fields only for Line Chart, Bar Chart, Stacked Bar Chart, Combinational Chart, and Combinational Dual-Y Chart view types. ■ If you want to configure drill-down to Process Monitoring Object Instances from the Business Measure, select group by fields as X-axis fields, and aggregate functions as Y-axis filelds. If you select the fields in reverse order, you get an error at run-time. <p>For example: In an Order-to-Cash process, say you would like to visualize Total Orders by CustomerGroup per Year. For this, you can build a Business Measure with the following query: Select Count(OrderID), CustomerGroup, from Orders group by CustomerGroup, datepart(Year, OrderDate). While selecting the X-axis fields, you can specify CustomerGroup as Primary X-axis and Year as Secondary X-axis. The resultant view will have data grouped based on Primary X-axis and Secondary X-axis with Primary X-axis shown as the label and Secondary X-axis as the legend.</p>
Chart Details > X-Axis Caption	<p>The selected X-Axis field name(s) appear in text box next to X-Axis Caption.</p> <p>The X-Axis Caption is not available for Pyramid and Pie charts.</p>
Chart Details > Select Y-Axis Fields	<p>Click . The Select Y-Axis Fields dialog box opens. See Select Y-Axis Field Interface for information about the fields that display, and fill in the appropriate fields. Select the Y-axis field and click OK.</p> <ul style="list-style-type: none"> ■ To view a Business Measure having multiple Group by clause, select multiple Y-axis fields of the Business Measure and visualize it in the Dashboard. ■ Selection of multiple Y-axis fields is supported for only the Line Chart, Bar Chart, Stacked Bar Chart, Combinational Chart, and Combinational Dual-Y Chart view types. It is not possible to select multiple Y-axis fields for the Pyramid Chart and Pie 2D Chart view types. <p>For example: In an Order-to-Cash process, say you would like to visualize Total Orders and Order Amount by Month. For this, you can build a Business Measure with following query: Select Count(OrderID) TotalOrders, Sum(OrderAmount) OrderAmount, Datepart(Month, OrderDate) Month From Orders Group by Datepart(Month, OrderDate).</p> <ul style="list-style-type: none"> ■ If you select the Combinational Chart view type, then, while selecting the Y-axis fields, you have to specify the view type for each field (OrderID, OrderAmount) from the available options (Line, Bar, and Area chart). The resultant view will plot both the fields on Y-axis.

	<ul style="list-style-type: none"> If you select the Combinational Dual Y Chart view type, select one field (OrderID) as the Primary Y-axis and other field (OrderAmount) as the Secondary Y-axis. Specify view types for them. The resultant view will plot, (OrderID) on Primary Y-axis (Left side) and (OrderAmount) on Secondary Y-axis (Right side).
Chart Details > Y-Axis Caption	<p>The selected Y-axis field name(s) appear in text box next to Y-Axis Caption.</p> <p>The Y-Axis Caption feature is not available for the Pyramid and Pie charts.</p>
X-Axis Label Style	<p>Select one of the following to specify the display style of the X-axis label:</p> <ul style="list-style-type: none"> Truncate: Display the truncated label. Rotate: Display the Rotated label. Displays the labels vertically. Slant: Display the label in Slant. Displays the labels at 45 degrees angle. Stagger: Display the Stagger label. Displays the labels in multiple lines. Wrap: Display the Wrapped label. If enough space is not available on the chart, the text of the labels will be trimmed and an ellipsis (...) is added at the end and would show tool-tips for those labels. <p>The X-Axis Label Style feature is not available for the Pyramid and Pie charts.</p>
Business Object Element	<p>Click . The Select Business Object Element dialog box opens. Select one of the output parameters as the Business Object element and click OK. The selected output parameter appears in the text box corresponding to the Business Object Element.</p> <p>The  icon is enabled only if the Is Transactional check box is selected.</p>
Time Frame	<p>By default, the defined time-frame appears in the Business Measure. If required, you can edit it.</p>
Input Parameter	<p>The value of the attribute for which you need dynamic input while designing the dashboard (as specified in Business Measure). You have to type in the input values, depending on the parameter and its data type.</p>

The fields on the Properties Interface of Business Measure Composite Control for KPI are as follows:

Gauge Details > Label	Type a label. The label denotes the description of the value. For example, Total_OrderAmount.
Gauge Details > Select Field	Click  . The Select Gauge Field dialog box opens. Select a gauge field and click OK .
Input Parameter	Refers to the value of the attribute for which you need dynamic input

	while designing the dashboard (as specified in Business Measure). Type the input values, depending on the parameter and its data type.
Range	Define ranges based on the business measure built. You can edit if required, and assign colors to the ranges using the 'Color Picker' option.
Render View on load check box	Select this option if you want to render the view while the page loads.
Show Legend	Select this option to view the legend information of the ranges and unit of measure. If you want to view only the guage, you must clear this option. This option is selected by default.
Time Frame	By default, the time-frame appears in the Business Measure. You can edit it as required.
Title	Specify chart title. The Title property is not applicable for Angular gauge and Linear gauge. You can use XForms label for the same.
Unit of Measure	Displays, by default, the unit of measure that you selected while defining the KPI. You can change the unit as required.
View	Specify one of the following as the View type: <ul style="list-style-type: none"> ■ Angular Gauge: Display the view as Angular Gauge ■ Linear Gauge: Display the view as Linear Gauge ■ Vertical Bullet: Display the view as Vertical Bullet ■ Horizontal Bullet: Display the view as Horizontal Bullet

The APIs for the Business Measure Composite Control are as follows:

setAsynchronous (String)	Value can be "true" or "false." By default, the request is sent in asynchronous mode. To change the mode to synchronous, set setAsynchronous to "false," and invoke the refresh API to redraw the chart.
refresh	Use this API to redraw the chart.
getChartObject	This API returns the Chart object ,on which the APIs of Chart can be invoked

The events for the Business Measure Composite Control are as follows:

ondataclick	Fires when clicked on a data point on the chart. Sample usage is as follows: <pre>function Form_InitDone(eventObject) { <bmid>.addListener(ondataclicklistenerFn); } function ondataclicklistenerFn(eventObject) {</pre>
-------------	---

	//eventObject.businessObject will give the businessobject as a JSON object for the clicked datapoint }
--	---

The time values displayed are based on the time zone of the client's browser in 12-hour format.

Using business activity monitoring

To use Business Activity Monitoring (BAM):

1. Configure the BAM service container
2. Load the BAM MDM Model application package

The detailed steps for configuring and loading follow.

To configure Business Activity Monitoring service container:

1. In the Explorer, click **System Resource Manager**.
The System Resource Manager window opens.
2. Under Service Containers, right-click the **Business Activity Monitor** service container and select **Properties**.
3. On the **General** tab, change **Startup Type** to **Automatic**.
4. Click the **BAM Connector > Monitoring Engine** tab.
5. In the Threads Configuration group, in **Event Processing Threads**, type the number of threads to be processed for the events.
These threads are used to process the events generated when a BPM is instantiated. By default, five threads are used in the event processing.
6. In the Threads Configuration group, in **Business Object Processing Threads**, type the number of threads to be processed for the business objects, that is either Process Monitoring Object or Business Event Response.
By default, 15 threads are used in the business objects processing.
7. Type the number of the business objects to be cached in **Business Object Cache Size**.
By default, 50 objects are cached.
8. In the **Profile Configuration** group:

E-mail address	Type the E-mail address. The email actions configured on KPI and Business Event Response of BAM are sent from the provided email address.
Display Name	Type the name. This is used as the display name when the mails are sent to the users.

9. In the Profile Configuration group, click  (File lookup) corresponding to Select proxy user for runtime execution.
The Organization Users dialog box opens.
10. Select the relevant user and click **OK**.
The contextual information, Web service operation, in both Process Monitoring Object (PMO) and Business Event Response (BER) is executed in the context of this user.
11. Ensure that this user has the required access, by assigning Roles from the required application, to execute the Web service operation defined in the contextual information of PMO or BER.
12. You can select a Proxy User to invoke Web service operations for:
 - The contextual information, Web service operation, in Monitoring Object and Business Event Response.
 - Rules of Business Event Response and KPI is fired with this user context.
 - The selected user must have access to the required application to invoke the Web service operations.
13. Save and restart the service container.
14. Right-click the **MDM** service container and select **Properties**.
15. On the General tab, change the **Startup Type** to **Automatic**.
16. Save and restart the service container.
17. Right-click the **MDM Hub Publisher** service container and select **Properties**.
18. On the General tab, change the **Startup Type** to **Automatic**.
19. Save and restart the service container.

To load BAM MDM Model application package:

1. In the Explorer, click **Application Deployer**.
An Application Deployer wizard opens.
2. Click **New**.
All the available applications are listed.
3. Right-click the **BAM MDM Model** application and click **Deploy** to install the application.
A Web page appears, listing the selected application and the dependencies.
4. Click **Next**.
5. Click the **MDM Model** tab and configure the MDM Hub publisher and MDM Spoke publisher service groups.
While deploying the application packages during installation, the MDM Service and MDM Hub publisher service groups are created.
6. Click zoom next to **MDM Hub Publisher**.
7. Select **MDM Hub publisher** and click **OK**.
8. Click **MDM Spoke Publisher** and click **Configure Service group** on the toolbar.
9. Select the **Service Configuration** tab.

10. Select database configuration from the **Select Database Configuration** list on the Repository tab.
The Repository database configuration selected must be the same as the one you selected in the Business Activity Monitor service container.
11. Select **Use different database** for Application option.
A new Application tab appears.
12. On the Application tab, select the database configuration from the **Select Database Configuration** list.
The Application database configuration selected must be the same as the one you selected in the Business Process Management service container.
The Business Process Management service container and Notification service container must always use the same database.
13. Click **Save**.
Service groups are configured and the Configure Service group button is disabled.
14. Click the **Database Schema** tab and select the database configuration from the **Select Database Configuration** list.
The selected database configuration must be the same as the one you selected on the Application tab in the above Step (2c).
15. Click **Next**.
16. Click **Deploy** to proceed with the installation.

In a primary-distributed setup, it is adequate to deploy the BAM MDM Model application package only in the primary server.

You must have the BAM service container in the organization to work with BAM.

All the BAM service containers in a cluster or a AppWorks Platform instance must always point to the same repository.

Chapter 5

Modeling cases

The AppWorks Platform case modeler offers an intuitive modeling environment to model and manage your business cases. It equips you with a set of tools that ease the entire process of modeling and managing a case. It helps you to model activities that do not necessarily follow any defined sequence as compared to a business process model and can execute in parallel. You can integrate various documents such as your business processes, human tasks, and apply the business calendar wherever necessary in your case model. The case modeling environment is where you design your case model.

To design your case model, use these procedures:

- [Creating a case model](#)
- [Designing a case model](#)
- [Setting the properties of a case model](#)
- [Designing a case model](#)
- Monitoring Case Instances (see *AppWorks Platform Administrator's Guide*)

Creating a case model

A case model helps you to model a company's complex cases involving activities that require human intervention and system based activities.

To create a case model:

1. Select a starting point and select  (Case Model).
You can either start [designing the case model](#) and save it later, or save it first and then design.
2. Click .

A new case model is created and added to the project content tree in Workspace Documents.

After you complete this task:

- [Publish](#) the case model.

Designing a case model

You can design a case model using the various case modeling constructs. An icon that is available on the case model toolbox represents each construct. See the [constructs used in a case model](#). The following procedure adds the constructs that are mandatory for a case model flow.

To design a case model:

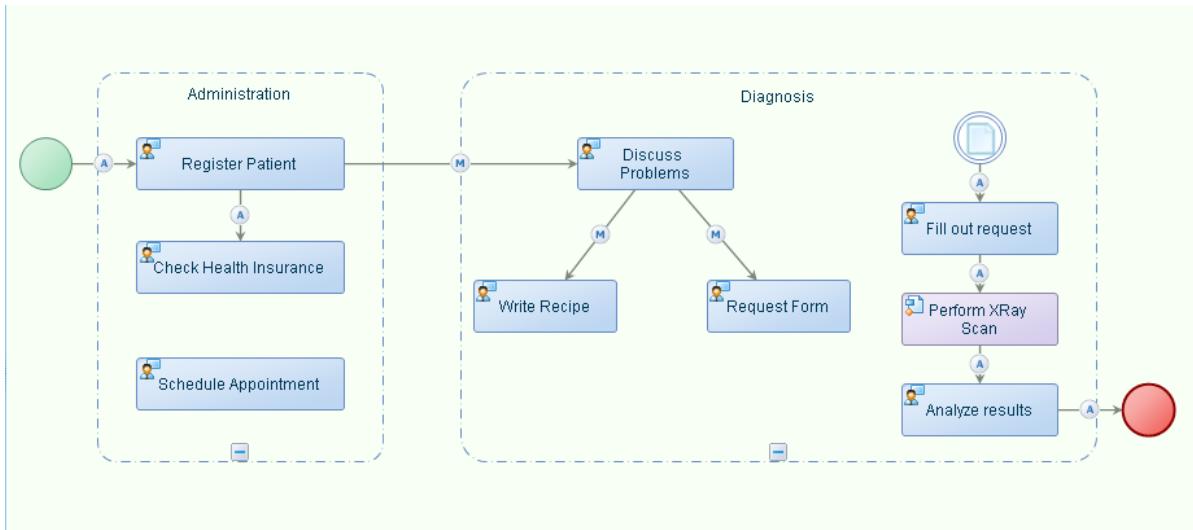
1. Select a starting point and click  to open a case model.
The case model appears in the case modeler.
 - By default, the grid view is enabled in the case modeler.
 - You can also use the quick draw constructs to accelerate the process modeling.
2. To handle the start of a case, drag  (Start Case) from the toolbox to the case modeler as shown in the example.
3. Drag  (Activity) from the toolbox to the case modeler. If the toolbox is not visible in the modeler, click  (Options) and select Toolbox.
4. To associate other documents (such as User Interface, Business Process Model, and Case Model) with the case activity, do **one** of the following:
 - Drag the <Document name> from the <Project> tree list in the **Workspace** tab onto the case model.
 - Drag the <Document name> from the <Document> group (User Interface, Business Process Model, or Case Model) in the **Insert** tab onto the case model.
 - Right-click the activity, select **Insert > <Document>** and select the appropriate document from the **Select a <Document>** dialog box. This adds the document to the activity in the case model. Depending on the document added, additional options to add more documents are displayed for the activity. Right-click the activity and select **Insert** to view these options.
5. Link the constructs by selecting the relevant connector icon from the toolbar as shown in the example below.
You can add other constructs required for your case model by dragging them to the case modeler.
You can align the connector text to the connector direction by right-clicking it and selecting **Align Text**.
6. Drag  from the toolbox to the modeler.

You have successfully designed a case model.

See [Examples of Various Case Models](#) for a view of cases using different constructs.

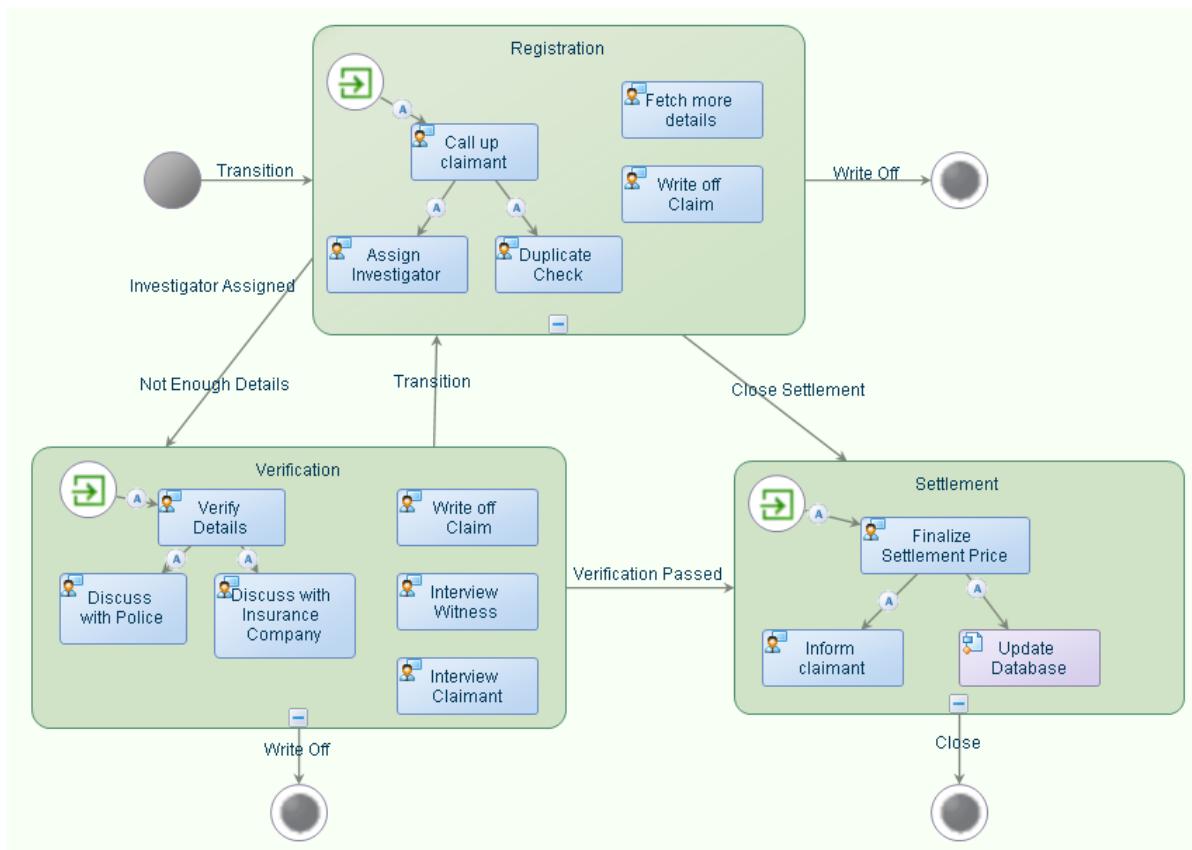
Sample case models

- Case model using the start and end case constructs - Patient health check



The example depicts activities that are involved when a patient consults a physician for physical health checkup. The various activities involved in Patient Health Check case model are grouped or clustered together based on their homogenous nature for easy processing. The physician performs parallel activities such as, Write Recipe and Request Form, while discussing with the patient. Activities or processes that do not need any human intervention are connected using the Automatic Follow-up flow connector; whereas, activities involving direct human intervention such as, Discuss Problems or Write Recipe, are connected using the Manual flow connector.

- Case model using initial and final state constructs - Handle complaints



The example depicts a complaint handling process wherein a claimant makes an insurance claim. The Handle Complaints case model depicts activities of homogenous nature grouped together into States such as Registration, Verification, and Settlement. Transition type flow connectors are used to depict transition from one State to another. Different options for connectivity of activities such as Automatic Follow-up and Manual are shown. Manual flow connectors are used to connect tasks involving human intervention and Automatic Follow-up flow connectors are used to depict consequent activities or tasks that are performed in a logical sequence. State Entry is used to represent events that are automatically raised when entering a State. Activities or tasks involving human intervention are represented using Task icons as shown for Discuss With Police or Discuss with Insurance Company activities. You can add supportive automated activities to States where partial automation of activities are involved. In the above case model, Update Database (a sub process) is an automated process which is added to the Settlement State. You can raise Events such as Close With Details or Not Enough Details for the Case while completing a task.

Next steps after you complete this task:

- [Setting the properties of a case model](#)
- [Setting the properties of a case activity](#)
- [Setting the properties of a state](#)
- [Setting the properties of a transition](#)
- [Publish the case model](#)

Setting the properties of a case model

Before you begin, create the following:

- [Functional Roles](#), [Users](#)
- [Create a business calendar](#)

After designing the case model, you must set its properties to make it executable in line with your business needs.

To set properties of a case model:

1. [Select a starting point](#) and click  to open a case model.
The case model appears in the case modeling environment.
2. Right-click in the case modeling environment to set the properties of the case model and select **Properties**.
The <Untitled Case Model> - Case Model properties pane appears. For more information, see [Case Model Properties Interface](#).
3. Click **General** to use a case overview form.
4. Click **Data** to use XML Schema Fragment and to create Case Identifiers for the case model.
5. Click **Duration** to add a business calendar to the case model or any of its activities.
6. Click **Case Variables** to add case variables.
7. Click **Work Assignment** to define work assignment to assignees.
8. Click **Work Distribution** to define distribution of work to case workers.
9. Click **Events** to create events for the case model.
10. Click **Attachments** to define the attachments definition and document types associated with that attachment definition.
11. Click **Annotation** to type additional notes or comments on the case model.

You have set the properties of the case model and it is ready to be executed. Next, see [Setting the properties of a case activity](#)

Case model properties interface

The Properties - Case Model pane helps in viewing and setting the properties of a Case model.

General tab

Name	Name of the current Case model. This is a mandatory field.
Description	Provide a description for the Case model.
Root State Name	All the follow-up activities defined in the model that are not associated to any specific user-defined state are considered to be part

	<p>of this state. This is an editable field and you can modify the default name provided. The state name provided here can be selected by the Inbox users to list the follow-up activities that are not associated to any specific user-defined state.</p>
Priority	<p>When multiple case models execute, the case instances corresponding to these case models are created. The priority displays the various levels of the case model, which assumes importance in the run-time. Using the Priority list, you can select the priority level High, Medium, or Low and provide the priority of the execution. If the priority is not set, the case instances execute in FIFO (First-In, First-Out) order. During the execution, the case model with High priority will take precedence over the case models with lower priority levels.</p>
Case Overview Form	<p>A Case overview form is a User Interface (UI) that allows you to view the data related to a case. In the UI, the information related to the case data is displayed in the case Inbox and Case Instance Monitoring (CIM). When a Case overview form is configured for any case model, all the users working on that case model will be able to access the case overview form. Select the required overview form to attach to the case model. Also, you can add a New User Interface or External User Interface or add an existing one from the Select a Case Overview Form window. In the case overview form, use the SOAP API GetCaseData to access the data associated to a case instance and populate the controls in the Case overview form.</p>
Applicability Service	<p>Often, the case workers or knowledge workers need to take decisions to select the most applicable sub-set from the next set of follow-up activities, based on the context and data available for a case. In such a scenario, it becomes necessary for a case worker to consider all the possible options and exceptions available for executing a task. The Applicability Service feature helps a case worker to accomplish tasks easily, while considering all the possible options and exceptions by implementing the required business logic on a Web service. When the Applicability Service feature is selected for the state, it becomes applicable for all the activities within the state. When the activities within the state are executed, based on the work option or exception, Applicability Service is triggered. For more information on configuration and related usage, see Using the Applicability Service in a Case Model.</p> <ul style="list-style-type: none"> ▪ Select Business Process or Web Service from the Applicability Service list. ▪ Click to browse and select the related business process or Web service. <ul style="list-style-type: none"> • If Business Process is selected, only the process models that

	<p>implement the contract shared are displayed. Implementation of such business processes are based on Contract First Development option with the WSDL that is shared.</p> <ul style="list-style-type: none"> If Web Service is selected, only the services that implement the WSDL are considered.
--	--

Data tab

Use XML Schema Fragment	A data model gives the structure of the information in the case model. Select the required XML schema fragment for the case to add a data model to the case model. By using the <code>CreateCase</code> SOAP API, you can trigger the Case model and pass the instance of the data model.
Case Identifiers	<p>A Case identifier is a business attribute that identifies or searches for a case in the Inbox. For example, the Customer ID or Order ID can be used as the business attributes by the case administrator to search for the case. The case data acts as the source for configuring the case identifiers.</p> <p>Click  to add a case identifier and provide the following details:</p> <ul style="list-style-type: none"> Name - Name of the case identifier. Type - Select the type of case identifier: String, Numeric, or Date. <ul style="list-style-type: none"> String Supports the storage of string values. Numeric Supports the storage of integer values only. Date Supports the storage of date in the yyyy-mm-ddThh:mm:ss.ms format; for example: 2012-10-15T05:01:42.951 XPath - Provide the XPath expression of the case identifier.

Duration tab

Business Calendar	<p>Business Calendar allows you to associate the timely business activities with the case model.</p> <p>Select the required Business Calendar from the available list and attach to the case model. If you want to continue with the default calendar, that is 24*7, do not provide any changes.</p>
Duration Type	
Escalate to	<p>Escalate to allows you to send notifications to the concerned member working on the case so as to ensure that the case will be completed at the earliest possible time. Select the member to whom you wish to send the notification from the list mentioned below.</p> <ul style="list-style-type: none"> Manager of the user

	<ul style="list-style-type: none"> ▪ User defined by case variable ▪ User who created the case <p>The escalation feature in the case model notifies the receiver. It will not support the re-assignment of a task.</p>
User	<p>To allocate an appropriate case worker to a case, select a case worker.</p> <p>This option is enabled only if the User defined by case variable check box is selected in the Escalate to list.</p>

Case Variables tab

You can use a case variable as a parameter in a case model. It has a type and a default value.

A case model can have zero or more variables and collectively define exactly the parts of the model that can be changed. You can create your own case variables on any case model by using the **Case Variables** tab on the case model properties page.

Case variables help case workers to regulate case execution behavior at runtime, by delaying the specification of certain case element values, until the instance is actually executed. Many decisions are made during design time by the case worker, for example; activities, norm times, business calendars and so on. However, some properties have to be decided by the case worker at runtime. To facilitate this process, the case designer can define custom case variables with default values. They can be set by the case worker only while creating a case or updating a case variable Web services at runtime of the Business Process Model.

To add a case variable:

1. Click  and provide the following details.

Name	Provide a name for the case variable.
Type	<p>Define the case variable type as any one of the following:</p> <p>Team - Select Team when the team which must perform the human task can change during the course of implementation or at runtime</p> <p>User - Select User when the user who must perform the human task can change during the course of implementation or at runtime.</p> <p>Role - Select User when the user who must perform the human task can change during the course of implementation or at runtime.</p> <p>Worklist - Select User when the user who must perform the human task can change during the course of implementation or at runtime.</p> <p>Distribution Algorithm - Distribution Algorithm or Scheduling algorithm suggests the handling of requests within a queue. By default, the Round Robin algorithm is implemented. It ensures that case workers</p>

	<p>complete a task one after another in a sequential manner. It also assigns time slices to each process in equal portions and in circular order, handling all the processes without priority. So, if a task is not complete in one time slice, it will be executed in the next cycle. However, you can implement your own scheduling algorithm too.</p> <p>Duration - Duration case variable is used at the activity level to define its own execution time. There can be different case variables to define duration for different activities. Provide the duration for the completion of all the activities within the case model in the Set Duration dialog box.</p> <p>String - Any value that can hold alphanumeric content. It holds the value that is used to instantiate the sub-process models.</p> <p>Default Value - Click  to select a default value. Based on the selection of Type, the corresponding default dialog box appears. Select the required Case Variable Type value as described:</p> <ul style="list-style-type: none"> ▪ User - Select a user from the Select a User dialog box. You can also use Find a User option to quickly find a user. ▪ Role - Select a role from the Select a Role dialog box. ▪ Distribution Algorithm - Select the required algorithm from the Select a Dispatch Algorithm dialog box. ▪ Duration - Provide the duration in Set Duration in days, hours, and minutes as required. ▪ String - Provide the required value from the look up provided. For Team (known as Organizational Unit) or Worklist, type the name of the Organizational Unit or Worklist that was created by the administrator using the Identity dashboards. For information about Identity dashboards, see the <i>AppWorks Platform Low-Code Design Guide</i>.
Description	<p>Provide a description for the case variable. This provides more flexibility because the case worker can now make last minute changes, when creating a specific case, by overriding the default case properties. For example, a case worker can decide that staff-B must work on a certain activity instead of staff-A.</p> <p>This tab appears at the case model view but not for activity view.</p>

Work Assignment tab

Assignee Type	<p>Based on your business requirements, you may have a single case worker, multiple case workers, or teams working on the case. Therefore, it is important to select an assignee type to clearly indicate who must work on the current case.</p> <p>Click  to add the type of assignment and to add users or teams so</p>
---------------	--

<p>that you may assign them the activities of the case as follows:</p> <ul style="list-style-type: none"> ■ Team - The users in the specified team who must work on this case or activity. ■ Role - The set of users with this role who must work on this case or activity. ■ Worklist - The list of tasks on which the teams are working. ■ User - The specific user. When you select this value, you must create or select a case variable that contains the value of a role, team, or user that specifies the users. <p>If the Static Value option is selected for Team (known as Organizational Unit) or Worklist, then type the name of the Organizational Unit or Worklist that was created by the administrator using the Identity dashboards. For information about Identity dashboards, see the <i>AppWorks Platform Low-Code Design Guide</i>.</p>

Work Distribution tab

Work distribution is a custom algorithm that you must define to distribute the work among the case workers. Select one of the following work distribution algorithm:

- **Use System Default** - The default settings of the system decides the work distribution.
- **Static (decide when creating the case) > Algorithm** - Select the algorithm to be used to distribute work amongst the case workers.
- **Dynamic (decide when creating the case) > Case variable to be used** - Select the case variable to be used to distribute work amongst case workers.

Events tab

Event Name	Name of the event that you have created. At times, you may want to define your own event types that are specific for the case model. After you create the event types, you can use them at several places. You can use an event to describe what can happen when certain situations like receiving a document, creation of a case, or completion of an activity occur. On an activity, you can also specify the event it can raise. You can use an event to trigger a transition between states. For example, Event Name: OrderApproval. <ul style="list-style-type: none"> ■ Click  to add an event.
Description	A meaningful description of the event created.

Attachments tab

The access control list (ACL) or permissions (Read, Write, Delete) defined in the case model property sheet are applicable to all the tasks in the respective case model.

Attachment Location	Optional. Attachment Location is useful if case model designers want
---------------------	--

	<p>to store the attachments in the custom specified locations. Case designer can chose any one option mentioned below.</p> <ul style="list-style-type: none"> ■ Default: This is selected by default and the attachments are stored in the default location. ■ Custom: Select this option to specify the custom location and chose the location type from the select box. <ul style="list-style-type: none"> • Static: Specify the static location. • Dynamic: Specify the path from the case message map. (case instance properties or case data). <p>By default all the attachments are uploaded to a standard location on document store if the custom location is not specified. This default location evaluates as per the following pattern:</p> <pre>casemanagement/<Case Model Name>/<Case Instance Id>/<AttachmentDefinitionName></pre> <p>For example:</p> <pre>casemanagement/LoanApplication/5337b252-d28c-4607-a0df-59654c5ebf4d/LoanDocument</pre>
Label Name	Select the check box to enable the case worker to view the attachment in the Inbox, modify the contents in the attachment, add another attachment, or delete the attachment.
Supported Document Types	<p>Case models use applicable or supported case documents. The Supported Document Types displays a list of all the existing document types that new attachment definition supports.</p> <ul style="list-style-type: none"> ■ Click  to select a document from the list that appears in the Select a Mime Type window. ■ If a required document type is not available in the list, select Any file type.
Read	Select the check box to enable the case worker to download the attached document in the Inbox.
Write	Select the check box to enable the case worker to download the attachment, modify it, or add a new attachment to the task in the Inbox.
Delete	Select the check box to enable the case worker to view the attachment in the Inbox, modify the contents in the attachment, add another attachment, or delete the attachment.

Annotation tab

Annotation	Provide any additional notes or comments on the case activity.
------------	--

Setting the properties of a case activity

After designing the case model, you must set its activity properties to make it executable exactly as you intended and in line with your business needs.

To set the properties of a case activity:

1. From Workspace Documents (Explorer) view, expand <Project> and click  to open an existing case model.
The case model appears in the case modeling environment.
2. Right-click an activity and select **Properties**.
The <Activity> - Activity Properties pane appears. For more information, see [Case Activity Properties Interface](#).
3. Click **General** to set the properties of the case activity.
4. Click **Annotation** to type additional notes or comments on the case activity.

You have set the properties of the case activity. Next, see [Executing a case model](#).

Case model activity properties interface

The Activity Properties - New Activity pane helps in viewing and setting the activity level properties in a case model.

The fields on the Activity Properties - New Activity Pane are as follows.

General tab

Description	<p>Provide a description for the case activity that you might want to reuse later.</p> <p>This field displays the Description of the Sub Case if it is for a Sub Case activity or the Name of the Sub Process if it is for a Sub Process activity.</p>
Priority	<p>Displays the various priority levels of the case activity in a list that assumes importance at runtime. When the case model activities are delivered as tasks to the Inbox, the tasks will have the appropriate priority column set based on this property. You can select one of the following options to set priority for the case model activity:</p> <ul style="list-style-type: none"> ■ Same as Main Case - Priority is same as defined in the main case model. ■ Same as Sub Process - Priority is same as defined in the linked business process. ■ Same as Sub Case - Priority is same as defined in the linked sub case model. ■ Other - To specify the execution priority that is different to the main process and the sub process, select one of the following:

	<ul style="list-style-type: none"> • Static - You can select any one of the priority levels: Highest, High, Normal, Low, and Lowest. • Case Variable - If you want to dynamically set the execution priority via a case variable of type string. <p>By default, the priority level property set at case model level is inherited by all the activities of the case model. You can override the default priority value at the activity level using this property.</p> <p>The option Same as Sub Process is only visible for a Sub Process activity and the option Same as Sub Case is only visible for a Sub Case activity.</p>
This activity must be performed (required activity)	<p>Select the This activity must be performed (required activity) check box to ensure that the current activity must be performed.</p> <p>By default, the activities of type automatic follow up are planned during case execution. However, the case workers still have an option in the runtime to remove such activities from execution through the follow up view of Inbox. Setting this property will disable it and enforce the case workers to perform the action to complete it.</p> <ul style="list-style-type: none"> ▪ This property is applicable only for activities of type Automatic follow up. ▪ State transition does not occur if there are any pending activities with the This activity must be performed (required activity) option selected within the current state.
Business Process Model	<p>This field is visible only when there is a sub-process activity within the case model.</p> <ul style="list-style-type: none"> ▪ Click  to browse and select an existing business process model.
Case Model	<p>This field is visible only when there is a sub-case within the case model.</p> <ul style="list-style-type: none"> ▪ Click  to browse and select an existing case model.
Instance ID	<p>The associated process will be instantiated with the value present in the associated Case Variable. Ensure the Case Variable of type 'String' is the selected variable. Also, ensure that the value for the variables selected is unique for every process instance that is created and that it does not exceed the limit of 50 characters.</p>
Wait until Sub Process is Finished	<p>This field is visible only when there is a sub-process activity within the case model.</p> <ul style="list-style-type: none"> ▪ If you select this check box, the case model waits until the sub-process is finished. ▪ If you clear this check box, the case model starts the sub-process

	<p>and continues with the next activity.</p> <p>This option is disabled if the sub-process is a short-lived process or a page flow process.</p>
--	---

Workflow Model tab

Delivery Model	<ul style="list-style-type: none"> ■ Click  to select an existing delivery model for the User Interface or External User Interface from the Select a Delivery Model window. ■ To select a delivery model, you must first create a delivery model. If a delivery model has both the Inbox Model and Email Model enabled, the Use Inbox Model and Use Email Model check boxes appear when the delivery model is attached to the task. ■ When there is only an Inbox model or an Email model enabled for the delivery model, the corresponding check box is enabled, that is, only the Use Inbox Model or Use Email Model check box appears. ■ However, if a delivery model does not have any of the Inbox model or Email model enabled, these check boxes do not appear. Select the appropriate check box that appears for the delivery model. ■ You can configure Inbox and or Email models for a Team and a Work list as well. A task can have both Inbox and Email models configured on it. ■ The Use Inbox Model option is not visible for a Sub Case activity or a Sub Process activity.
----------------	---

Duration tab

This tab is not visible for a Sub Case activity or a Sub Process activity.

Business Calendar	<p>Enables you to select an available business calendar by browsing the list from the Select a Business Calendar dialog box.</p> <p>Business calendar is used to calculate the due date durations for an activity. In runtime, when an activity is past the due date, the activity will be marked in red color in the Inbox and if the escalations are configured, they will be triggered. By default, business calendar configured at case model level will be applicable for all the activities of the case model. You can override the default business calendar at an activity level using this property.</p>
Duration	<p>Specify the duration in terms of Days, Hours and Minutes, during which an activity needs to be completed by the case worker. You can also provide the duration dynamically by configuring case variables.</p>

Escalate to	When the duration specified for the task exceeds maximum duration, you can escalate it for further action to any one of the following: <ul style="list-style-type: none"> ■ Manager of the user: Manager of the user's principal team will receive the escalation notification. ■ User defined by case variable. ■ User who created the case.
User	Select the case variable mapped to User who would receive the escalation notification. This option is enabled only when the option User defined by case variable is selected from Escalate to list.

Work Assignment tab

This tab is not visible for a Sub Case activity or a Sub Process activity.

Assignee type	<p>Type of assignment you would want to use to allocate work to various targets.</p> <p>Select an Assignee type from the list to add target to the case activity, so that you can assign the case activities to them, as described below:</p> <ul style="list-style-type: none"> ■ Team: The users in the specified team who should work on this activity. ■ Role: The set of users with this role who should work on this activity. ■ Worklist: The list of teams associated to work list. ■ User: When you select this value, you must create or select a case variable that should contain the value of a user DN. <p>After selecting an assignee type from the above:</p> <ol style="list-style-type: none"> 1. Select Static Value or Case Variable, and click  to select the required value for the selected assignee type. 2. If the Static Value option is selected for Team (now known as Organizational Unit) or Worklist, then type the name of the Organizational Unit or Worklist that was created by the administrator using the Identity dashboards. For information about Identity dashboards, see the <i>AppWorks Platform Low-Code Design Guide</i>.
---------------	--

Authorization tab

This tab is visible only when the **Assignee Type** in the **Work Assignment** tab, as described above, is **Team** or **Worklist**.

Roles	Typically, in an organizational or team set up, different users perform different tasks. When you want only a specific member, members of
-------	---

	<p>your team, or worklist to view and/or execute a task, you can provide the required access permissions in the Authorization tab. Depending upon the selected Assignee Type (in the Work Assignment tab), which can either be a Team or a Worklist, you can select specific roles for the task on hand from the Team or Worklist.</p> <p>To select a specific role or roles do the following:</p> <ol style="list-style-type: none"> 1. In the Roles list box, click  . The Select Role window opens. 2. Select the required role. The selected role appears in the Name column.
Permissions	<p>In the Permissions pane, specific rights or permissions for performing a task to specific roles can be provided.</p> <p>By default, whichever role remains highlighted or selected in the Roles > Name section, the Permissions is visible only for that specific role. To view Permissions for each individual role, select the required role by clicking it. Access rights permission to a specific role (s) consists of the following rights:</p> <ul style="list-style-type: none"> ■ Change Due Date: Grant permission to a specific role to change the due date for the activity. ■ Execute: Specific role(s) to perform this task. ■ View: Specific role(s) to only view the current task but not perform any operations on it. ■ A role that does not have either Execute or View permissions will get the task in the Case Inbox but cannot view or execute the task. To grant permission to a specific role, click the role and select the relevant check box in the Permissions pane. ■ If you want to remove the Execute or View permissions of a particular role, select the check box against the required role and click .

Work Distribution tab

This tab is not visible for a Sub Case activity or a Sub Process activity.

Select one of the following work distribution options:

- **Use System Default:** The default settings of the system determine work distribution.
- **Static (decide now) > Algorithm:** Select the algorithm to be used to distribute work amongst the case workers.
- **Dynamic (decide when creating the case) > Case variable to be used:** Select the case variable to be used to distribute work amongst case workers.

For information on creating and working with dispatch algorithms, see [working with Dispatch Algorithms](#).

Raised events tab

This tab is not visible for a Sub Case activity or a Sub Process activity.

Name of the event that you have raised or created. By selecting the **Can be raised** option, you can trigger the selected events when the activity is opened from the Inbox. Only those events that have this option selected are displayed in the **Events** tab and can be triggered for that particular activity.

Attachments tab

- This tab is not visible for a Sub Case activity or a Sub Process activity.
- The access permissions that are set at the model level can be over-written here (at the activity level).

All the attachment definitions that are applicable on the case model are displayed in this tab. The access levels set on these attachments are inherited by the activities by default. You can select the required attachment definition and modify the access levels as explained below.

Inherit	Select the check box to inherit the default permissions provided to the case worker for attaching a document. These permissions are set while modeling the case and they can be overwritten at the activity level. To over-write the default permissions, clear the Inherit check box and select the relevant permissions.
Read	Select the check box to enable the case worker to download the attached document in the Inbox.
Write	Select the check box to enable the case worker to download the attachment, modify it, or add a new attachment to the task in the Inbox.
Delete	Select the check box to enable the case worker to view the attachment, modify the contents in the attachment, add another attachment, or delete the attachment in the Inbox.

Links tab

URL	Type the URL of the location where the work instructions for tasks and case are made available for the case worker to view and implement.
Description	Description of the URL.

Annotation tab

Annotation	Provide any additional notes or comments on the case activity.
------------	--

Case activities interface

This topic describes the tabs and fields that appear on the Case Activities interface and operations that can be performed.

Overview tab

The Overview tab displays general information of the selected case instance.

Overview tab - Groupbox Header Info

Instance ID	ID of the selected case instance. For example, 46996aa4-241c-49f9-a1e8-9729f9f0506d.
Status	Status of the case instance. For example, NEW.
Last Modified On	Time at which the case activity was last modified.
Start Time	Start time at which the flow of control has moved to the case activity.

Overview tab - Groupbox Case Info

Case Info	Case data in XML format.
-----------	--------------------------

Case Activities tab

The Case Activities tab displays a list view of all activities in the selected case instance with the following information:

Activity	General status of each activity. For example, New Activity.
State	State of each activity. For example, default state.
Status	Status of each activity. For example, Released.
Planned On	Day, date, and time at which an activity was planned. For example, Monday, November 10, 2008 3:48:33 PM.
Started On	Start day, date, and time at which the flow of control in the case model has moved from previous activity to the current or selected activity. For example, Tuesday, November 11, 2008 3:48:33 PM.
Acted On	Day, date, and time at which the case worker performed the activity or acted upon it. For example, Wednesday, November 12, 2008 3:48:33 PM
Acted By	Name of the case worker who acted upon an activity. For example, Luca.
Activity Type	<p>Activity type.</p> <p>An activity in a case model can be any of the following:</p> <ul style="list-style-type: none"> ■ HUMANTASK: This denotes an activity that requires human intervention. Click anywhere on this row to drill down to view task details. ■ BPMTASK: This denotes an activity that is independent of human intervention and processes automatically. Click anywhere on this row to drill down to view the activity details. ■ CASETASK: This denotes an activity of a case model that may or may not need human intervention. You cannot drill down further for a CASETASK.

Performing Operations in the Case Activities tab view

To obtain a view of details and history of the activities in the case instance, you can perform the following drill-down operations in the Case Activities tab.

Show Details	Details of the selected case activity. To drill-down and view the details of an activity: <ul style="list-style-type: none">▪ Right-click an activity and select Show Details.
Show History This option is visible only when the Activity Type is Manual(Task)	History of the selected case activity. To drill-down and view the history of a manual activity: <ol style="list-style-type: none">1. Right-click an activity and select Show History.2. Alternatively, select an activity and click . The task view appears displaying relevant history. <p>This option is not visible when the Activity Type is Sub Case or Sub Process.</p>

Setting the properties of a state

After designing the case model, you must set the properties of the States within the case model to make it executable exactly as you intended and in line with your business needs.

To set the properties of a state:

1. From Workspace Documents (Explorer) view, expand <Project> and click  to open an existing case model.
The case model appears in the case modeling environment.
2. Double-click  (State) and select Properties. Or, right-click  (State) and select Properties.
The <State> - State Properties pane appears. For more information, see [State Properties Interface](#).
3. Click **General** to provide a description and annotation for the State.
4. Click **Duration** to associate the State with a business calendar and to specify the duration of the State.

You have successfully set the properties of the State.

Setting the properties of a transition

Transitions provide various configuration options when a case instance moves from one state to another state. A state can have multiple incoming and outgoing transitions. A transition can be triggered by either an event being received or when specific or all planned activities in a state are completed.

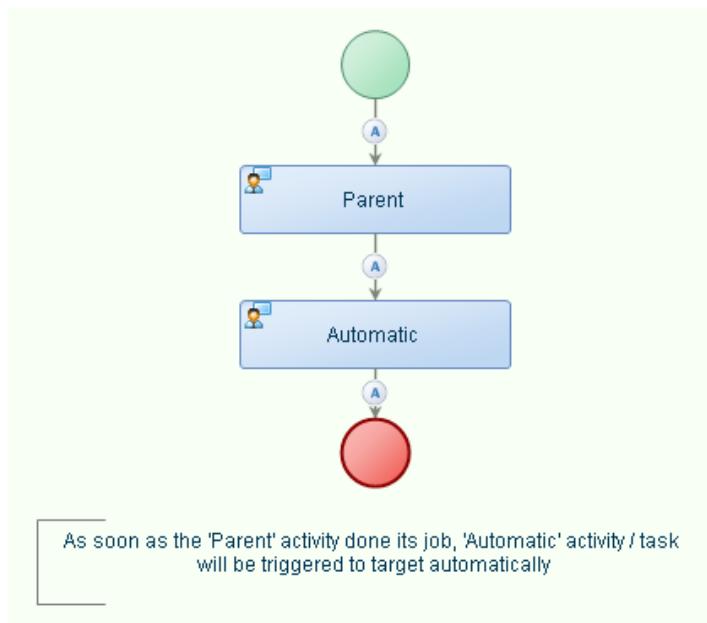
To set the properties of a transition:

1. From Workspace Documents (Explorer) view, expand <Project> and click  to open an existing case model that has states.
The case model appears in the case modeling environment.
2. Double-click  (Transition) and select Properties. Or, right-click  (Transition) and select Properties.
The <Transition Name> (from <Source state> to <Target state>) - Transition Properties pane appears. For more information, see [Transition Properties Interface](#).

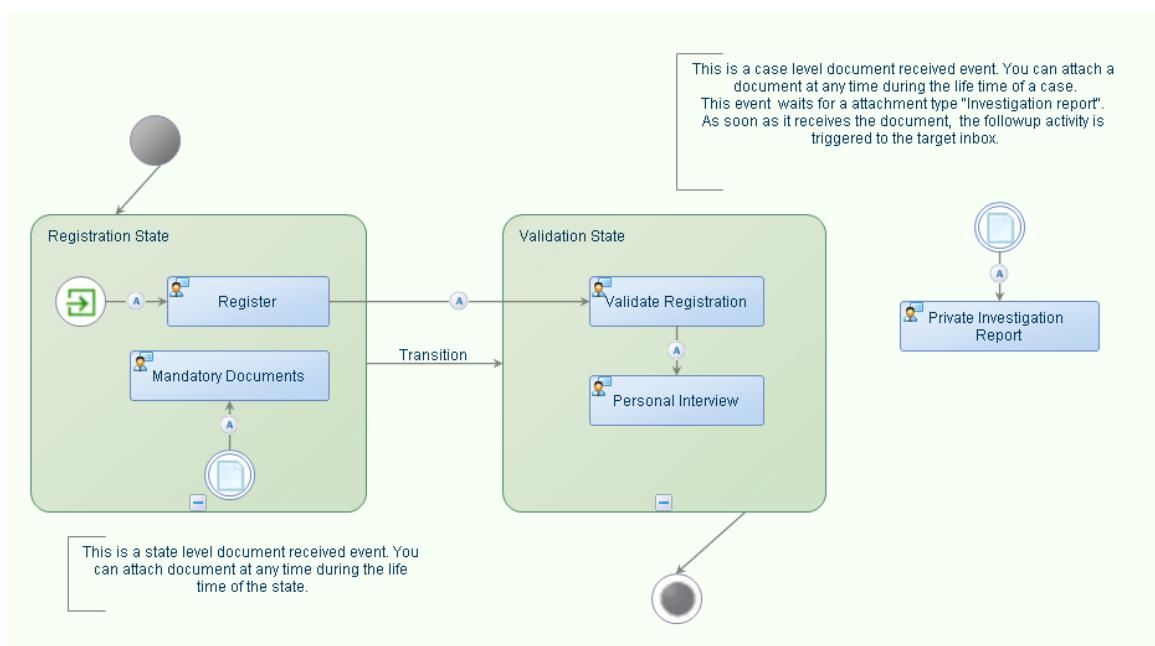
You have successfully set the properties of the Transition connector.

Examples of various case models

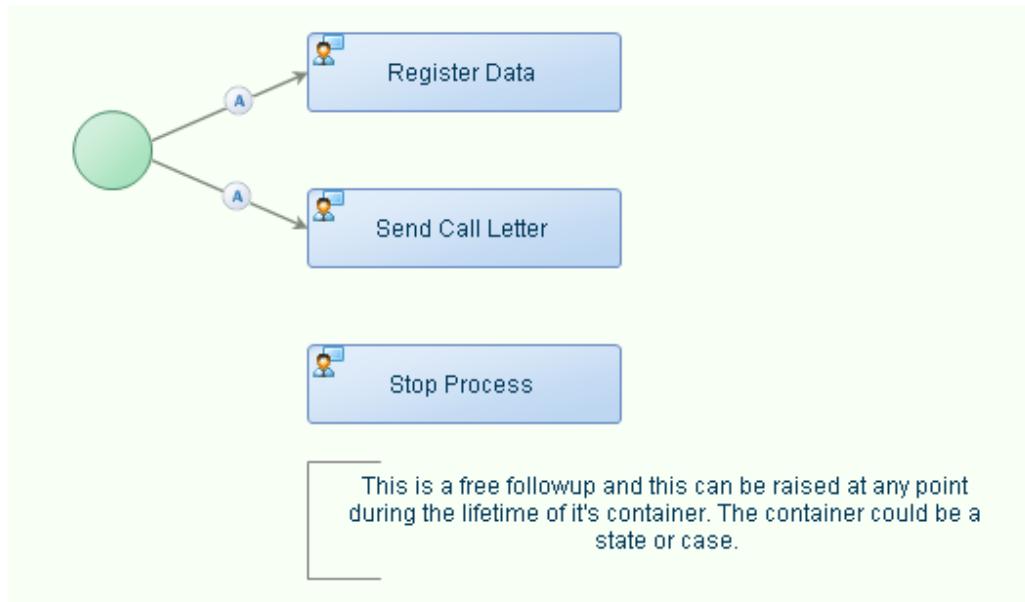
Using automatic follow-up in a case



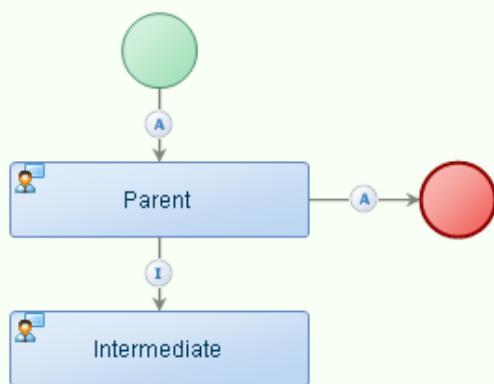
Using document received event in a case



Using free follow-up in a case

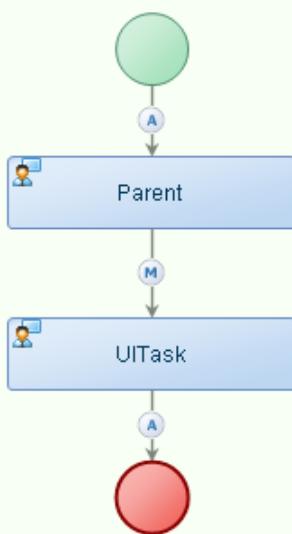


Using intermediate follow-up in a case



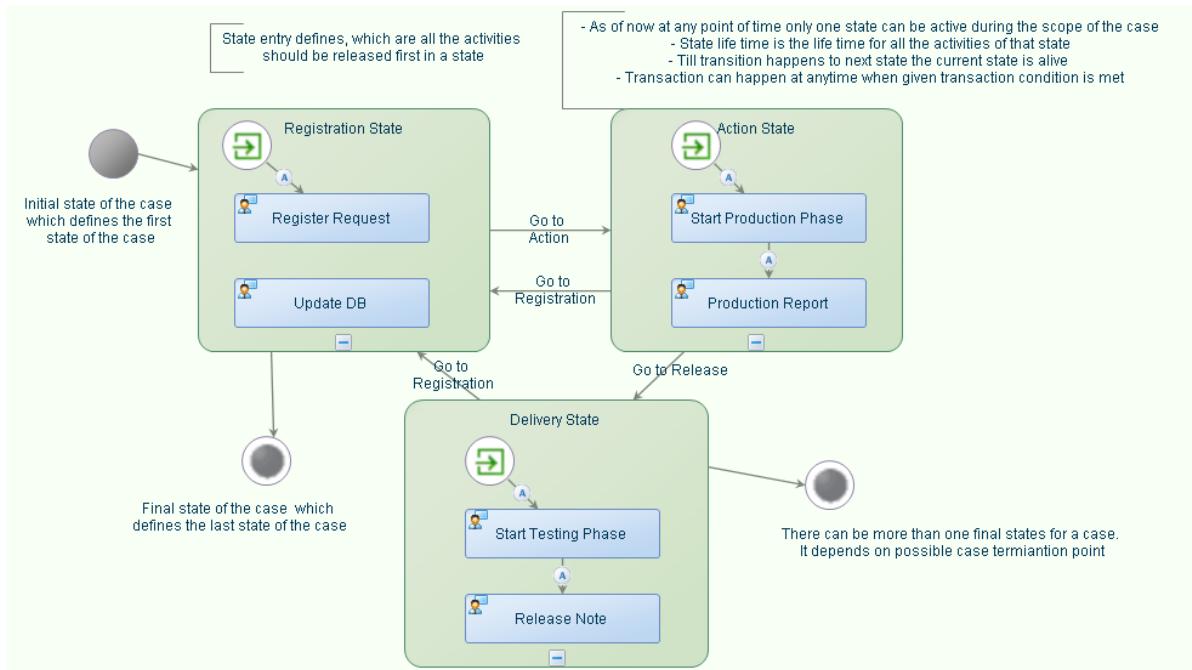
Here, the parent activity triggers the intermediate follow-up. Till the intermediate activity is complete, the parent activity will not appear in the Inbox. This also means that the parent activity is completed only after the intermediate task is completed.

Using manual follow-up in a case

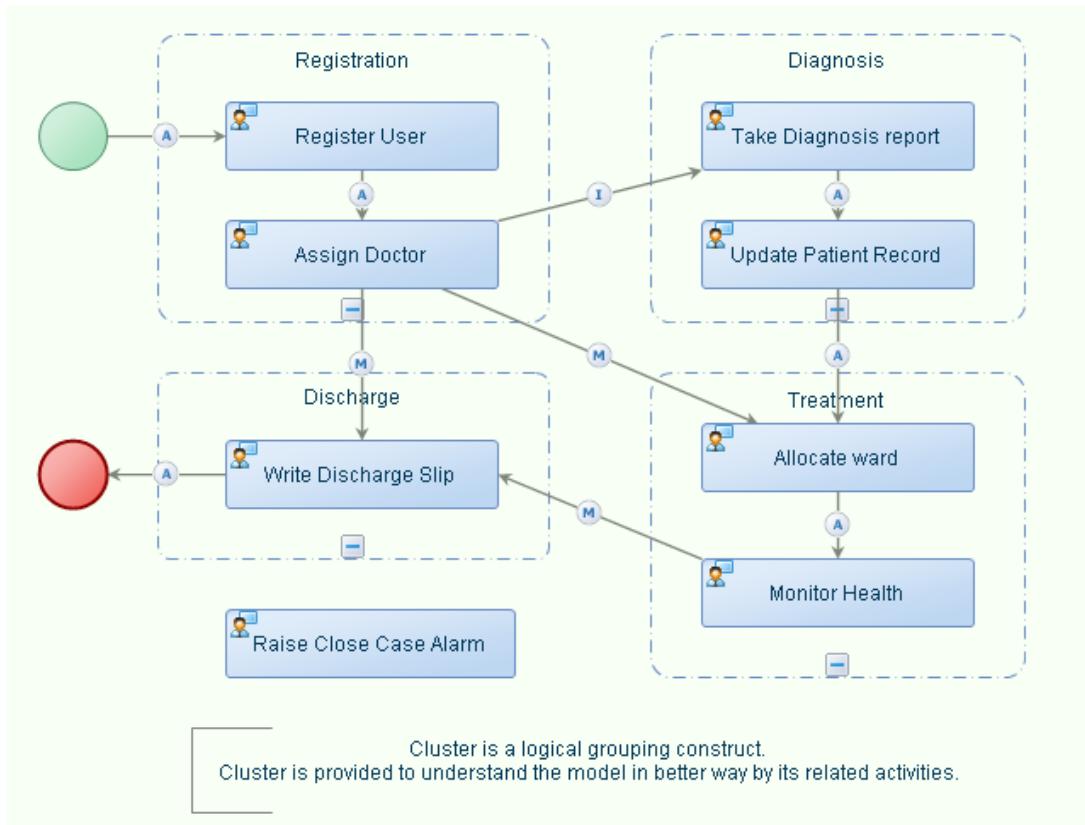


Before completing the task, the parent activity should select the manual followup activity / task that should be executed.

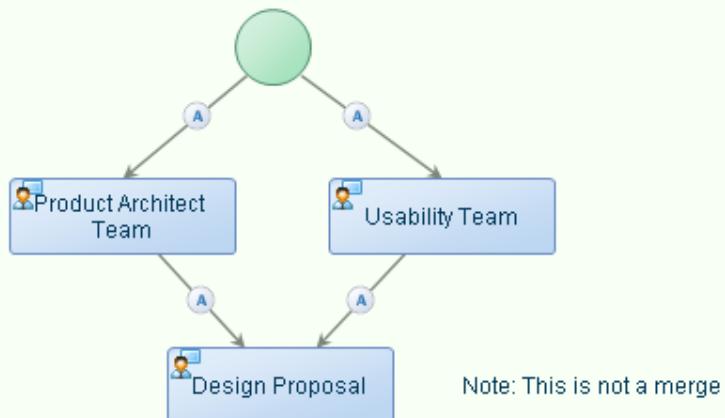
Using states in a case



Using cluster in a case

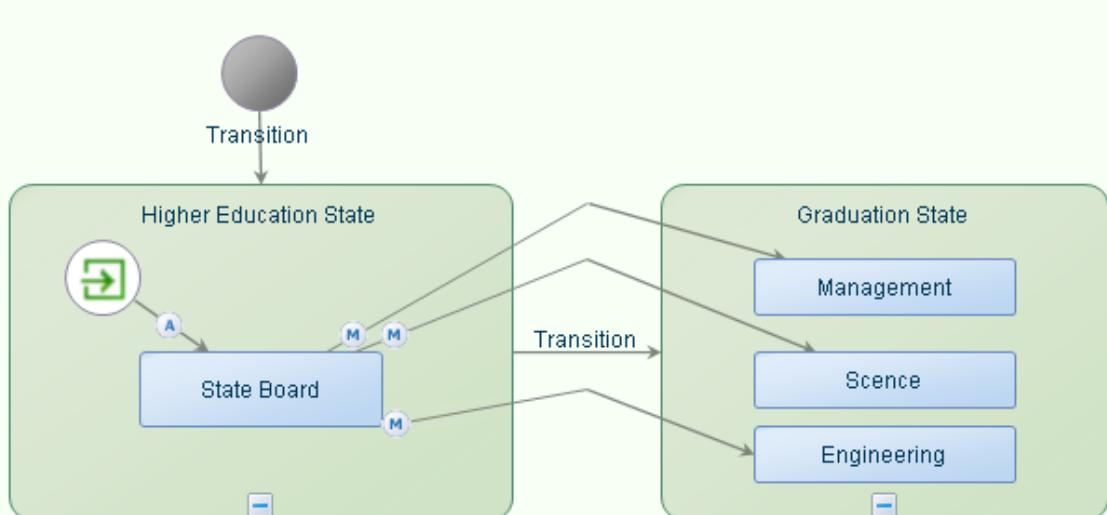


Using multiple plans for same task in a case



In the above model, Design proposal is released twice.
Product Architect team gives a design proposal & Usability team also gives a design proposal

Using pre-planning across states



You can plan your next state of activities much earlier. You can plan any activity across states
They remain in planned state until you move to that state.
When transition happens to that state, all your planned activities will be in your inbox

Executing a case model

Before you begin:

- You must [Validate](#) and [Publish](#) a Case Model.

Executing a Case Model creates an instance of the Case Model at runtime. The associated tasks of the Case instance are delivered to the Case Inbox. Case workers can access these tasks and complete them. The following procedure executes a Case Model from the CWS Workspace.

Note: You cannot pass Case data or Case variable details when the Case Model is executed from the Collaborative Workspace and Case Model editor. To pass Case data and Case variables while triggering a Case Model, use the [CreateCase] SOAP API. For more information, see [Triggering a case model from an XForm](#).

To execute a state model:

1. [Select a starting point](#) and click  to open a Case Model.
The Case Model is displayed in the Workspace Documents.
2. In the Workspace Documents, go to <**Solution**> > <**Project**>, right-click the <**Case Model**> and select **Execution** > **Run**. Or, right-click in the Case Modeling environment and select **Execution** > **Run**.

The Case Model is executed and a confirmation message displays indicating that an instance of the Case is successfully created. If you want to view the instance related details, select Show Case Instances.

Case instance specific properties

When a Case model is instantiated, a property is instantiated in the Message Map to provide instance-specific properties. These properties are collectively known as instance properties. They are used to retrieve information about the Case instances at runtime and can be used in Case definitions.

XML structure of instance-specific properties

The XML structure of instance-specific properties are as follows:

```
<instanceProperties debugMode="true">
  <source type="CASE">00215A63-1B7C-11E2-EAC6-E35CBC02BFED</source>
  <currentstate id="RootCaseModelState">Default State</currentstate>
  <identifier>00215A63-1B7C-11E2-EAC6-E32BDCFF3FED</identifier>
  <name>TestCaseModel</name>
  <modelid>{5D8D56E3-523A-47E1-9118-F24520F58201}</modelid>
  <modelrevision>00215A63-1B7C-11E2-EAC6-E32BDCFE5FED</modelrevision>
  <currentuser>cn=jdoe,cn=organizational
users,o=system,cn=cordys,o=vanenburg.com</currentuser>
  <lastmodified>1357280545980</lastmodified>
  <space>organization</space>
```

```

<instantiationuser>cn=JDoe,cn=organizational
users,o=system,cn=cordys,o=vanenburg.com</instantiationuser>
<startTime>1267526084395</startTime>
<organization>o=system,cn=cordys,o=vanenburg.com</organization>
<priority>3</priority>
<rootinstance>00215A63-1B7C-11E2-EAC6-E32BDCFF3FED</rootinstance>
<rootinstancetype>CASE</rootinstancetype>
<activityinfo>
    <PriorActivityInstanceDetails>
        <Name>TestSubCaseModel</Name>
        <Id>00215A63-1B7C-11E2-F840-C8A87E477F09</Id>
        <InstanceId>00215A63-1B7C-11E2-F859-F4830FA9BFBF</InstanceId>
        <Type>CASETASK</Type>
        <Status>COMPLETED</Status>
    </PriorActivityInstanceDetails>
</activityinfo>
<stateinfo>
    <PriorStateInstances>
        <State>
            <Name>Validation</Name>
            <Id>001CC435-DD29-11E3-EB97-B174AB7D9289</Id>
            <Parent>001CC435-DD29-11E3-EBCD-55E343EE1024</Parent>
        </State>
        <State>
            <Name>Registration</Name>
            <Id>001CC435-DD29-11E3-EBCD-55E343EE1024</Id>
            <Parent>RootCaseModelState</Parent>
        </State>
    </PriorStateInstances>
<ActiveStates>
    <State>
        <Name>Confirm</Name>
        <Id>001CC435-DD29-11E3-EBCD-5A886CD024</Id>
        <Parent>001CC435-DD29-11E3-EBCD-5A886CE07024</Parent>
    </State>
    <State>
        <Name>Approval</Name>
        <Id>001CC435-DD29-11E3-EBCD-5A886CE07024</Id>
        <Parent>RootCaseModelState</Parent>
    </State>
</ActiveStates>
</stateinfo>
</instanceProperties>

```

Elements in the XML structure of instance-specific properties

The following table describes the elements in the XML structure of instance-specific properties.

ActiveStates	<p>Details of current active states. A state in which the Case instance exists presently is considered active. Valid values:</p> <ul style="list-style-type: none"> ▪ State/Name: Name of the active state.
--------------	---

	<ul style="list-style-type: none"> ▪ State/Id: Unique identifier of the related state as available in the design time. Used in scenarios where applications want to drill-down to the details of a particular state. ▪ State/Parent: Unique identifier of the parent of the state as available in the design time. It can be used in scenarios where hierarchy of the state is required.
currentstate	(Deprecated) State of the Case instance Property stateinfo/ActiveStates/State/Name can be used to get current state of the Case Instance.
currentuser	LDAP DN of the user or role that last triggered the Case instance. Note: Initially, the DN is the same as the value in the instantiationuser element. However, if the Case instance is triggered again by an intermediate user or role, that user or role is the owner for further operations in the process. For example, if a Case instance sends a message to a participant and is waiting for a response, the participant in turn sends back a response that triggers the same Case instance. The owner for all the future operations with the Case instance is the participant whose response triggered the Case instance.
id	Unique identifier of the current state of the Case instance.
identifier	Unique identifier of the Case instance.
instantiationuser	LDAP DN of the user or role that instantiated the Case instance.
lastmodified	The time when the Case instance was last modified. The format of lastmodified is usually in 13 digits, where the last three digits are milliseconds. For example, if the startTime is 1267526084498, 498 indicates milliseconds and need not be converted to a valid AppWorks Platform or UTC format.
modelid	Unique identifier of the Case model.
modelrevision	Custom Case instantiation type. It can be used by applications to specify their identity for better Case monitoring and data filtering.
name	Name of the Case.
organization	DN of the organization in which the Case instance is created.
PriorActivity InstanceDetails	In a Case model, you can now map the previous activity details as a part of the pre-assignment of activity. This option is available for the activities that are connected through Automatic, Manual, or Intermediate follow-up modes and if both the previous and current activities are in the same State. This information is completely transient, and is not stored. This information is only available in the message map. The following are the details: <ul style="list-style-type: none"> ▪ Name: Description of the previous activity.

	<ul style="list-style-type: none"> ■ Id: Unique identifier of the previous activity as available in the design time. ■ InstanceId: Unique identifier of the previous activity instance as available in the runtime. ■ Type: Type of the previous activity. Valid values: <ul style="list-style-type: none"> • HUMANTASK - If the activity is associated to a User Interface. • BPMTASK - If the activity is associated to a BPM model. • CASETASK - If the activity is associated to a Case model. ■ Status: Status of the previous activity. The possible values are: <ul style="list-style-type: none"> • COMPLETED - When the previous activity has already been completed. • SUSPENDED - When the current activity is triggered as a part of the intermediate follow-up of the previous activity.
priority	Execution priority of the Case, where the priority ranges from 1 to 5; 1 corresponding to low and 5 corresponding to high.
PriorStateInstances	<p>Details of the previous active states. When a transition is considered from one state to another state, the current state and its parent states are considered as previous states. This information can be used in scenarios where an application wants to traverse back to the previous states from the current active state.</p> <p>A state in which the Case instance exists presently is considered as active.</p> <p>PriorStateInstances stores the details of all the completed states in the most recent state transition. For a single transition, the state that exits first is added to the first child of the PriorStateInstances node followed by the next state and so on. When a state has sub-states, all the sub-states are exited before the parent state exits. The parent state information is always the last state in the PriorStateInstances node.</p> <p>Consider the following scenarios:</p> <ul style="list-style-type: none"> ■ ParentState1 has two nested sub-states: SubState1 followed by SubState2. When ParentState1 exits, both the sub-states exit before the ParentState1; that is, SubState1 exits first, followed by SubState2 and ParentState1. The same order is followed while adding the details in the PriorStateInstances node. ■ ParentState1 has SubState1, and SubState1 in turn has a nested SubState2. When ParentState1 exits, SubState2 exits first, followed by SubState1 and ParentState1. The same order is followed while adding the details in the PriorStateInstances node. <p>The following are the details available in PriorStateInstances:</p>

	<ul style="list-style-type: none"> ▪ State/Name: Name of the state that was previously active. ▪ State/Id: Unique identifier of the related state as available in the design time. It can be used in scenarios where applications want to drill-down to the details of a particular state. ▪ State/Parent: Unique identifier of the parent of the state as available in the design time. It can be used in scenarios where hierarchy of the state is required.
rootinstance	Displays the Process instance ID if the rootinstancetype is PROCESS and displays the Case instance ID if the rootinstancetype is CASE .
rootinstancetype	Source type of the root of the instance. It can be a Process or Case model.
source	Parent process instance ID from which the current Case instance is instantiated. This information helps in scenarios where a Case instance must be aware of the parent process that instantiated it.
space	Place from which the Case instances are executed and displayed in the Case Instance Manager .
startTime	Time when the Case instance was instantiated. The format of startTime is usually in 13 digits, where the last three digits are milliseconds. For example, if the startTime is 1267526084395, 395 indicates milliseconds and need not be converted to a valid AppWorks Platform or UTC format.
type	<p>Application type that created the instance. The possible values are:</p> <ul style="list-style-type: none"> ▪ Web service - The default value when a Case instance is created using the CreateCase API. You can also customize the value as a part of the parameter source in the CreateCase API. ▪ CASE - The value when a Case instance is triggered from another Case instance. ▪ PROCESS - The value when a Case instance is triggered from another BPM instance.

Attaching files to a case

You can attach various file formats to case models, case instances, and the related case activities displayed in the AppWorks Platform Inbox. While modeling the cases, you can define the types of documents that can be attached to a case instance or an activity and set the access permissions on the attachment. Based on these attachment definitions and permissions, you can view or manipulate the attachments of case instances or activities. For example, if you provide only read access to an attachment at the activity level, user can only view the contents in the attachment but cannot update or delete the contents in that

attachment. Users can upload only those file formats that are defined in the attachment definition of an activity.

Before you begin:

By default, attachments in AppWorks Platform are stored in Document store, which is part of AppWorks Platform Repository Service Container. When the Web server and AppWorks Platform Repository Service Container are working on the same computer, the following properties point to the locations as indicated:

- com.eibus.web.tools.upload.UploadWritePath- points to **<AppWorks Platform_installdir>contentuploadcontent**
- com.eibus.web.tools.download.DownloadReadPath- points to **<AppWorks Platform_installdir>contentdownloadcontent**

Ensure that all users working on document store have write permissions on the **<AppWorks Platform_installdir>contentuploadcontent** and the **<AppWorks Platform_installdir>contentdownloadcontent** folders. For information on Document Store, see Document Store in the AppWorks Platform Administration Guide on My Support.

You can customize the tabs displayed in the Inbox, tasks, or activities, to enhance or minimize the number of items displayed. This will reduce the clutter as well as increase the productivity

To customize the tabs or to enhance or minimize the number of items displayed:

1. If you are unable to see the Attachments tab, click on the titlebar.
All the tabs that are displayed on the interface are shown in the list as selected. The tabs that are not displayed on the interface are shown as unselected.
2. Select **Attachments** option from the list.
3. To close any of these tabs, clear the check box of the tab you want to close.

To attach files to a case:

1. On the Welcome page > My Applications, click  (My Inbox).
The My Inbox window opens.
2. Click the **Case(s)** tab in the left pane of My Inbox.
All the case models are displayed below.
3. Click the required <case model>.
All the case instances to which the user is associated are displayed in the right pane.

To attach a file to a case instance:

1. Select the required case instance and click  on the Inbox task toolbar.
2. Alternatively, you can do the following:
 - a. Double-click the required case instance.
The Case Instance Overview window opens.

- b. Click the **Attachments** tab.
All the attachments that are already attached to that case model are displayed.
- c. Click  on the toolbar.
The files attached to a case instance are displayed across all the case activities related to that instance.

To attach a file to the case activity:

1. Click the required case instance to view all the released activities related to it and do any of the following:
 - Select the required activity from the Related Activities pane that appears and click  on the Inbox task toolbar.
 - Alternatively, you can do the following:
 - Double-click the required case activity in the Related Activities pane.
The Case Activity Overview window opens.
 - Click the **Attachments** tab.
If there are any attachments that are already attached to the case model, they are displayed here.
 - Click  on the toolbar.
The Add Attachment dialog box opens.

Based on the access permissions set on the attachment while modeling the case, the user can view or update the files that are displayed. For instance, if the attachment has only Read permission, the user can only view the existing files but cannot attach any new file to the activity. If the attachment has both Read and Write permissions, the user can attach a new file as well as modify the existing file and reload it. If the attachment has Read, Write, and Delete permissions, the user can delete the file as well.

2. Select the **Attachment Type**.
This Attachment Type is defined while modeling the case.
3. Click  next to the Document field and select the type of document to attach to this activity.
4. Provide a meaningful **Description** to the attachment.
5. Click **OK** to close the dialog box.
The attachments are displayed in the Case Attachments section.

The attachment is added to the case activity.

To view the files attached to an activity:

- Select the required activity and click  on the Inbox task toolbar.

All the attachments related to the activity are displayed.

Chapter 6

Working with a business calendar

Business calendars represent the operating hours of a business. A business calendar specifies a time zone and a set of time period rules. The time zone indicates your local time zone by default. The time period rules determine the days, dates, working hours (available for business activities), and non-working hours (unavailable for the business activities). You can use the AppWorks Platform business calendar to:

- Create a business calendar
- Define the business calendar
- Check user availability for a task
- Calculate due dates
- Calculate actual hours spent on a task
- Calculate dispatch logic

The business calendar also helps you:

- Avoid setting due dates on holidays or weekends
- Manage timeouts
- Avoid unexpected delays

Apart from checking the availability of users, you can use the business calendar to calculate time available for the business. When specifying the periods that business events are to take place (such as a message being sent or a particular task instance becoming overdue), you may want to express intervals in business time by including it in the business calendar.

Using the AppWorks Platform business calendar, you can configure either a static or a dynamic business calendar. You can define, manage, and track your business calendar using any of the following options:

- **Default Week Calendar:** Define your week day or business week pattern including the business time.
- **Calendar Exceptions:** Use this feature to specify and apply exceptions such as holidays, leaves, and non-working days to your business calendar.
- **Time Zone Settings:** Synchronize your business calendar across different time zones and countries. Specifying the time zone is important as the start time and end time are specified in the local time. The time zone will convert the local time to the corresponding

coordinated universal time (UTC-GMT) and it also supports daylight saving time. When you create a new business calendar, the time zone set in the Welcome page preferences will be shown by default. If you have not set any preference for the time zone in the Welcome page preference, then by default the server time zone option is shown.

Creating a business calendar

You create and add a business calendar to your project or attach it to your business process model as part of your project or solution.

To create a business calendar:

1. Select a starting point and select  (Business Calendar) to create a business calendar. The Untitled Business Calendar - Business Calendar dialog box opens with a view of the Default Week Calendar tab.
2. You can start defining your business week pattern, calendar exceptions, and set required time zone and save it later or save it first and then define the business calendar.
3. Click **Save**.

A business calendar is created and added to your project content tree.

Defining a business week pattern

Before you begin:

- [Create](#) a business calendar.

To define a business week pattern:

1. [Select a starting point](#) and click  (Business Calendar) to open an existing business calendar. By default, the Default Week Calendar tab view of the business calendar appears.
2. Enter the **Name** and **Description** for your business calendar.
3. Click  (Insert) to define your working day. A list appears in the Day column.
4. Choose the required day from the list to make it as a working day.
5. Click in the **Start Time** column and enter the start time in HH:MM format for the selected working day.
6. Click in the **End Time** column and enter the end time in HH:MM format for the selected working day.
Your business week pattern is defined.
7. Repeat Step 3 through Step 6 to define your entire business week.
8. You can plan multiple periods for a single workday; for instance, you can define one period from 9:00 until 12:00 and another period from 13:00 until 17:00 on Monday.

9. You can define exceptions to your default business week pattern and link the exceptions to your business calendar using the Calendar Exceptions tab.

Multiple segments should not overlap for periods on a single day.

The time that you enter in this screen is interpreted using the time zone defined at [Time Zone Settings](#).

Creating an exception list to the business calendar

Before you begin:

- [Define](#) your default business week calendar.

While you define the default working week pattern using the Default Week Calendar, you might want to define exceptions to this for a specific date(s). As an example, consider Christmas when most businesses are closed for a day or two. Another exception might be some additional overtime hours on a specific date. Exceptions are defined on an exception list, which is linked to a business calendar.

A business calendar can have only one exception list.

To create an exception list to the business calendar:

1. [Select a starting point](#) and select  (Calendar Exceptions) to create an exception list to a business calendar.
The <Untitled Calendar Exceptions - Calendar Exceptions> wizard opens.
2. Enter the **Name** and **Description** for your calendar exception and click **Save**.
The Save Document dialog box opens.
3. Click  (Save in Folder).
The Select Folder dialog box opens.
4. Select a folder or project and click **OK**.

Your Calendar Exceptions is attached to your project. You can now start defining the exceptions to your business calendar.

Note: The [exception list must be linked to a business calendar](#) to make it active.

Importing calendar exceptions

Before you begin:

- [Create](#) a business calendar.
- [Create](#) a calendar exception list.

Importing calendar exceptions and linking those exceptions to your business calendar enables easy access to existing calendar exceptions on the Internet, your local system, or any other machine and re-use. iCal is one of the popular calendar formats for sharing calendar information. You can import calendar information stored in this format into an

exception list, and you can import existing calendar exceptions into your calendar exceptions.

To import calendar exceptions:

1. Download an .ical file from the web and save it to your file system.
2. [Create a project in CWS](#).
3. In the Workspace Documents window, do any one of the following:
 - [Create](#) the Calendar Exceptions document to which you wish to import the .ical file.
 - Select <project> > <Calendar Exceptions> to which you wish to import the .ical file.The Calendar Exception window opens.
4. Click  (Import Exceptions).
The iCal Import Wizard opens.
5. In the **Location** group box of the iCal Import Wizard, click  (Lookup) for the .ical file to browse and import the .ical file to the Calendar Exceptions.
6. Define the **From date** and **To date** to specify the date range within which the exceptions need to be imported from the .ical file.
7. Click **Finish** to save the exceptions to your Calendar Exceptions.
The calendar exceptions are imported from the .ical file and are available to you to [link them to your business calendar](#).

You have successfully imported the calendar exceptions to the Calendar Exceptions document.

Defining exceptions to the business calendar

Before you begin:

- [Define](#) your default business week calendar.
- [Create](#) an exception list for your business calendar.

You can define exceptions to your business calendar by [importing exceptions from an iCal formatted file](#) or by performing the steps listed below.

To define exceptions to your business calendar:

1. [Select a starting point](#) and click  Calendar Exceptions) to open an existing calendar exception.
2. Do any one of the following:
 - Click  (Import) to import an existing iCal.
The iCal Import Wizard - Calendar Exceptions dialog box opens.

- Click  (Lookup) to select and import an existing iCal from the system where AppWorks Platform is installed.
- Select the required date range and click **Finish**.
The selected iCal is imported to the Calendar Exceptions list. See [the iCal Import Wizard Calendar Exceptions Interface](#) for information on the fields in the iCal Import Wizard - Calendar Exceptions dialog box.

- Click  (Insert) and enter Date.
 - Select the **Non working day** option.
 - Enter **Start Time**, **End Time**, and an appropriate **Description**.
 - After selecting a non working day, the Start Time and End Time fields are disabled. The Start Time and End Time can be entered only if it is a working day.
 - You can now [link them to your business calendar](#).
- You can specify the Date by typing it in or by clicking the calendar icon.
- You can define only one exception for a specific date.
- When you define an exception for a specific date, the default working pattern for that date is ignored.
- The time that you enter on this screen is interpreted using the time zone defined [at Time Zone Settings](#).

iCal import wizard calendar exceptions interface

The following table describes the various fields on the iCal Import Wizard - Calendar Exceptions dialog box.

Date range to import > From date	Click  to select a start date of the exceptions from the existing iCal that you are importing.
Date range to import > To date	Click  to select an end date of the exceptions from the existing iCal that you are importing.
Location > File	Click  to select and import an existing iCal from the system where AppWorks Platform is installed.

Linking calendar exceptions to the business calendar

Before you begin:

- [Define](#) your calendar exceptions.

After you create and define your calendar exceptions, you must link these exceptions to your business calendar to incorporate them in your business calendar.

A business calendar can have only one exception list.

To link calendar exceptions to the business calendar:

1. Select a starting point and click  (Business Calendar) to open a business calendar.
The Business Calendar window opens.
2. Select the **Calendar Exceptions** tab.
The Calendar Exceptions tab view opens.
3. Select the **Use Exception List** option.
The Exception List field is enabled.
4. Click  (Lookup) associated with the Exception List field.
The Select a Calendar Exception dialog box opens displaying all available calendar exceptions.
5. Select a required calendar exception.
The exception list from the selected <calendar exception> appears in your business calendar view.
6. Click **Save**.

The selected calendar exception is now linked to your business calendar

Setting time zone to a business calendar

Before you begin:

- Create a business calendar.

To set the time zone to a business calendar:

1. Select a starting point and select  (Business Calendar) to open a business calendar.
The business calendar window opens.
2. Click the **Time Zone Settings** tab.
The Time Zone Settings view opens.
3. Select the required time zone from the list.
4. Click **Close**.
The Confirm Message dialog box opens.
5. Select **Save** to save the selected time zone settings to your business calendar.

This also completes the creation of a business calendar.

Publishing a business calendar to an organization

Before you begin:

- [Create](#) a business calendar.
- Optional. Create, define, and link a calendar exception list to the business calendar.

To use a business calendar (within a business process), you need to publish your calendar to an organization. Perform the following steps to publish a business calendar and make it available for use.

To publish a business calendar to an organization:

1. From the Workspace Documents (Explorer) view, expand the project where the business calendar you want to deploy is available.
2. Right-click the business calendar and select **Publish to Organization**.

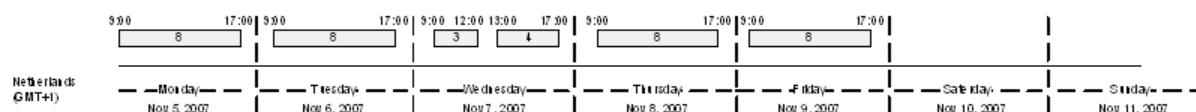
The business calendar is deployed to your organization and is ready for use.

Using a business calendar

When using a business calendar for planning, the actual number of open hours and the closed hours is taken into account. Different businesses have varied open hours. Therefore, when planning a task, for example, which takes 10 hours, it is important to know exactly how many hours can be spent in fulfilling this task per day. Many businesses or enterprises have the standard eight working hours per day. However, business is closed on some days because of holidays. A business calendar registers the exact open hours for an organization. When planning a task, activity, and duration, the actual open hours can be considered to calculate, for example, the end date and time of a task.

Planning in hours and minutes

Consider the following example of a business calendar that is open from Monday through Friday from 9:00 A.M. to 17:00 P.M., except for Wednesday. The open hours on Wednesday are from 9:00 A.M. to 12:00 noon and from 13:00 P.M. to 17:00 P.M. This business calendar is depicted in the following diagram.

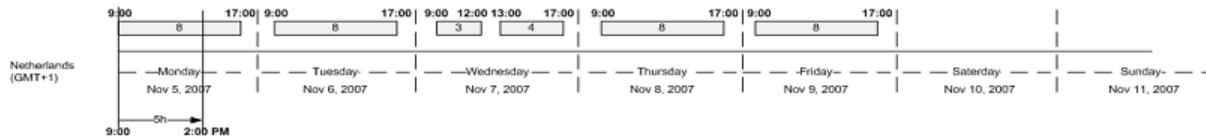


With this context in mind, consider the following examples.

Example 1

Assume that a task:

- Needs 5 hours for completion.
- The task starts on November 5 at 9:00 A.M. (as per the Dutch time zone (GMT+1)). This is depicted in the following diagram.



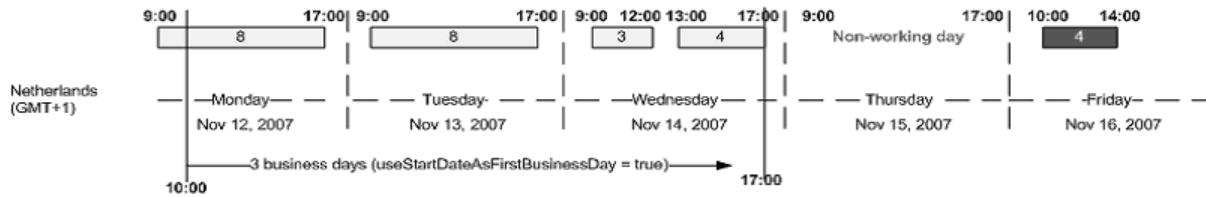
In this scenario, the task will end on November 7 at 4:05 P.M. (GMT+1).

Planning in business days

Besides planning in actual business hours and minutes, people sometimes tend to define a certain time span in a less accurate term and call it a 'business day'. A business day is any day that has a business capacity greater than zero. As an example, consider an organization that promises the customer that an answer will be provided within three business days. In such cases, the actual number of hours and minutes is not relevant but closed business hours such as weekends and holidays are certainly relevant. By default, a planned task will end at the end of the business day.

Assume that a task:

- Needs 3 business days for completion.
- The task starts on November 12 at 10:00 A.M. (as per the Dutch time zone (GMT+1)). This is depicted in the following diagram.

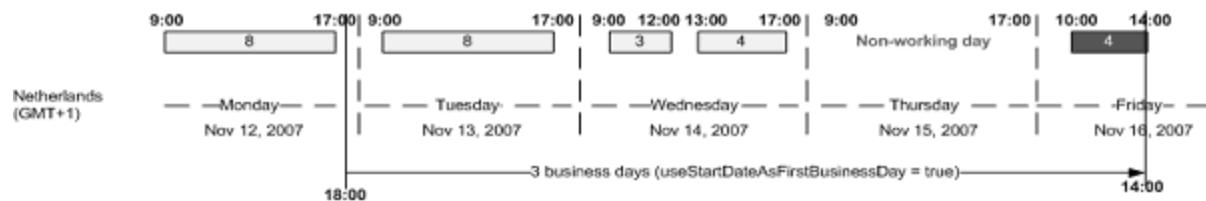


In the scenario:

- 12th November is a business day because it has a business capacity greater than zero (8 business hours)
- 13th November is a business day because it has a business capacity greater than zero (8 business hours).
- 14th November is a business day because it has a business capacity greater than zero (8 business hours).
- When you plan a task for three business days that starts on 12th November, the end date for completion of the task will be 14th November. By default, when you plan in business hours, the task ends at the end of the business day. On this day, the latest business hour is 17:00 P.M. (GMT+1). This task will end on 14th November at 17:00 P.M. (GMT+1).

Assume that a task:

- Needs 3 business days for completion.
- The task starts on 12th November at 18:00 P.M. (as per the Dutch time zone (GMT+1)). This is depicted in the following diagram.



In the scenario:

- 12th November is a business day because it has a business capacity greater than zero (8 business hours), but the task start after the business is closed so this day has no business capacity left and is ignored.
- 13th November is a business day because it has a business capacity greater than zero (8 business hours). This is counted as the first business day.
- 14th November is a business day because it has a business capacity greater than zero (8 business hours). This is counted as the second business day.
- The third business day is expected to be 15th November. However, a calendar exception (see Define exceptions to the daily week pattern) has been defined for this day and the day is marked as a non-working day. This means that this day has no capacity. So we move over to the next day.
- The third business day is now expected to be 16th November. Another calendar exception has been defined for this day. The default working hours have changed to 10:00 to 14:00. So, this day has a business capacity greater than zero (4 business hours) and the task will end at the end of this business day which is 16th November 14:00 P.M. (GMT+1).

Using a business calendar in a business process model

Before you begin:

- [Create a Business Calendar](#)
- [Define Business Week Pattern](#)
- [Define Exceptions to the Business Calendar](#)
- [Set Time Zone to the Business Calendar](#)

After you create a business calendar, define the business week patterns and exceptions, you need to use it in a business process model. The setup activities help to set the period in which each activity needs to be completed thereby, ensuring that all the activities of a business process are completed in time. Associating a business calendar with a business

process also helps you to identify the spill over time for completion of activities and ensures that non-working days are excluded.

To use a business calendar in a business process model:

1. In Workspace Documents (Explorer), expand <Project> and click  (Business Process Model) to open an existing business process model.
2. Right-click the business process model whose properties you want to set and select **Properties**.
Alternatively, double-click in the business process modeler or press the keyboard shortcut F8 to view the properties.
The Properties - Business Process Model - <Description> pane appears.
See [Business process model properties interface](#) for more information on the fields on the Properties - Business Process Model pane.
3. Click the **General** tab.
4. Select the **Use Business Calendar** option.
The Business Calendar field appears.
5. Click  (browse).
The Select a Business Calendar dialog box opens.
6. Select an existing business calendar.
7. Click **Save**.

You have successfully used the business calendar in a business process model.

Chapter 7

Modeling data

Modeling data facilitates creation and management of business objects that are used in a business process or a similar context. AppWorks Platform offers various data modeling techniques to manage and transform business data into Web services.

AppWorks Platform uses Object Templates to create and manage data that is independent of any external relational database. An Object Template is based on the XML Schema Definition of the business object. It is used to create Rules and Decision Tables that operate within a business process.

Data Transformation Model is another option to model data. AppWorks Platform facilitates the [transformation of data](#) from a source application to a target application, from one format to another. These models can be used inside the business process as a Web service or can be used to transform data received as an incoming message into the business process.

When directly dealing with relational data, AppWorks Platform provides the facility to create Database Metadata and store it in the form of data models. In this context, WS-AppServer, which functions as an independent processing layer between a relational data source and a UI client, reads the database details through Database Metadata, and creates data models.

Creating an XML schema

XML schema is a set of schema fragments that define the structure of an XML instance. In AppWorks Platform, with the help of XML Schema Definition (XSD), you can interpret the XML format of all types of documents. It acts as a common platform for various types of documents to exchange the document metadata. For example, in AppWorks Platform, you can use the message mapping feature in a Business Process Model and determine the XML content by reading the XSD in the WSDL for a Web service.

To create an XML schema:

1. [Select a starting point](#) and click  (Schema Definition Modeler).
The Schema Wizard opens.
2. Provide the **Name** and **Description** of the schema.
3. Provide the XML schema. Do one of the following:

- By default, **Enter schema or Instance** is selected. Provide the relevant schema XSD or schema XML in **Enter schema**.
- To import the schema from an external URL, select **Import from URL**.
 - Provide the relevant URL of the schema in **Schema URL**.
For example:
`http://www.ws-i.org/SampleApplications/SupplyChainManagement/2002-08/RetailOrder.xsd.`
 - Select **Authentication Details** if you want to authenticate the schema for selected users only. These authentication credentials enable the user to read the schema.
 - Provide the authentication credentials.
 - Based on the type of authentication, provide the username.
 - Provide the relevant password in the text box that is displayed.

The different types of authentication are as follows:

Basic	The server authorizes a request only after validating the username and password.
Digest	The server authorization is similar to that of Basic authentication, but the password is not sent in clear text.
NTLM	NTLM authentication protocol is implemented in various Microsoft networks.
SAML 1.1/Basic	The method of authorization combines Basic authentication and SAML 1.1 authentication.

If the imported schemas exist in the workspace, they will be updated with the latest content. If they are not found in the workspace, the newly created schemas will be stored in the `importedschemas` folder under the project.

If the schema contains imported schemas, they will be shown at the following two locations:

1. Within a folder under the `importedschemas` folder.
 2. Within the XSD reference under the parent schema.
4. Click **Finish**.
- The created XML Schema is added to the project. For more information, see [schema created from an external URL](#).

The XML Schema is created. You can use the schema as a base for creating the related schema fragments.

Creating XML schema fragments

A schema fragment consists of elements, complexType or simpleType of an XML schema. You can use these schema fragments in creating rules, decision tables, action templates, condition templates, or data transformations.

To create XML schema fragments:

1. From the Workspace Documents (Explorer), right-click the required XML schema and select **Add Schema Fragment**. Alternatively, open the schema view, click (View Children) on the tool bar, and click in the Add Children pane that appears. The Schema Fragment - Schema Fragment wizard opens.
2. Type required information in the **Name** and **Description** fields of the Schema Fragment - Schema Fragment wizard and click **Finish**. The XML Schema Editor opens.

To create or edit a schema fragment using the Text tab:

You can create a schema fragment in one of the following ways:

1. Using the Text tab (XSD representation of the schema fragment), click the **Text** tab.
2. Type the schema for the template.
See the example.
3. Click the **Instance** tab to view the XML representation of the schema fragment.
4. You can synchronize the Tree, Instance, and Text views of the XML Schema; the changes you make on one tab are reflected on the other two tabs automatically.
While synchronizing, the schema editor ensures that the XML Schema you create is valid. An alert is shown when an element is not in the required order or the XML schema provided is not as per the standards.
5. Click **Save**.

The schema fragment is created and is attached to the XML schema.

To create or edit a schema fragment using the Tree tab:

1. Using the Tree tab (graphical representation of the schema fragment), click the **Tree** tab.
2. In the Schema Tree pane, right-click the element, select **Add** and select the required child element.
3. Specify the necessary properties for the element in the Properties section.
Refer to the image provided in the XML Schema Editor topic.
4. Click the **Instance** tab to view the XML representation of the schema fragment.
5. You can synchronize the Tree, Instance, and Text views of the XML Schema; the changes you make on one tab are reflected on the other two tabs automatically.
While synchronizing, the schema editor ensures that the XML Schema you create is valid. An alert is shown when an element is not in the required order or the XML schema

provided is not as per the standards.

6. Click **Save**.

The schema fragment is created and is attached to the XML schema.

XSD and instance views of the ProductionOrders schema fragment

- XSD view

```
<xsd:element name="ProductionOrders" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="ID" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
            <xsd:element maxOccurs="unbounded" name="OrderDetails">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="UnitPrice" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                        <xsd:element name="Quantity" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                        <xsd:element name="Discount" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Customer" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                <xsd:element name="Order_Date" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="Required_Date" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="Shipped_Date" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipVia" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="Freight" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipName" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipAddress" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipCity" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipRegion" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipPostalCode" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="ShipCountry" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

■ Instance view

```

<xsd:element name="ProductionOrders" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="ID" type="xs:string"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
            <xsd:element maxOccurs="unbounded" name="OrderDetails">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="UnitPrice" type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                        <xsd:element name="Quantity" type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                        <xsd:element name="Discount" type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="Customer" type="xs:string"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                <xsd:element name="Order_Date" type="xs:string"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                    <xsd:element name="Required_Date" type="xs:string"
                        xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                        <xsd:element name="Shipped_Date" type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                            <xsd:element name="ShipVia" type="xs:string"
                                xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                <xsd:element name="Freight" type="xs:string"
                                    xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                    <xsd:element name="ShipName" type="xs:string"
                                        xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                        <xsd:element name="ShipAddress" type="xs:string"
                                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                            <xsd:element name="ShipCity" type="xs:string"
                                                xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                                <xsd:element name="ShipRegion" type="xs:string"
                                                    xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                                    <xsd:element name="ShipPostalCode" type="xs:string"
                                                        xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                                                        <xsd:element name="ShipCountry" type="xs:string"
                                                            xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:element>

```

XML schema created from an external URL

The XML schemas created from an external URL are saved in the workspace of the project. If the parent schema refers to the imported schemas by the import or include option, the

schemas are imported into the corresponding folder of the workspace as specified in the schema location URL.

The imported schemas are created in the imported schemas folder under the project. The details about the location path of the schema is explained in the following example.

Consider that you import the schema from the external URL

`http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd0` and save it in the Schemas folder in the project with name CurrencyService.

This schema imports XSD from

`http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd2`, which in turn imports

`http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd1`.

The following artifacts are created in the project:

- <Project>/Schemas/CurrencyService.xsd
- <Project>/importedschemas/www.restfulwebservices.net/wcf/CurrencyService_svc/xsd2.xsd
- <Project>/importedschemas/www.restfulwebservices.net/wcf/CurrencyService_svc/xsd1.xsd

If you expand <Project>/Schemas/CurrencyService.xsd, you see its Elements and Types and also a reference to the imported schemas as `Reference_to_xsd=xsd2`.

Note: Moving the schemas to a different location will not duplicate the schema or affect the schema functionally.

For example, if you import another schema from the external URL

`http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd3` that imports `http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd1`, XSD1 is not created in the workspace. Instead XSD1 is updated with the external schema.

Creating an object template

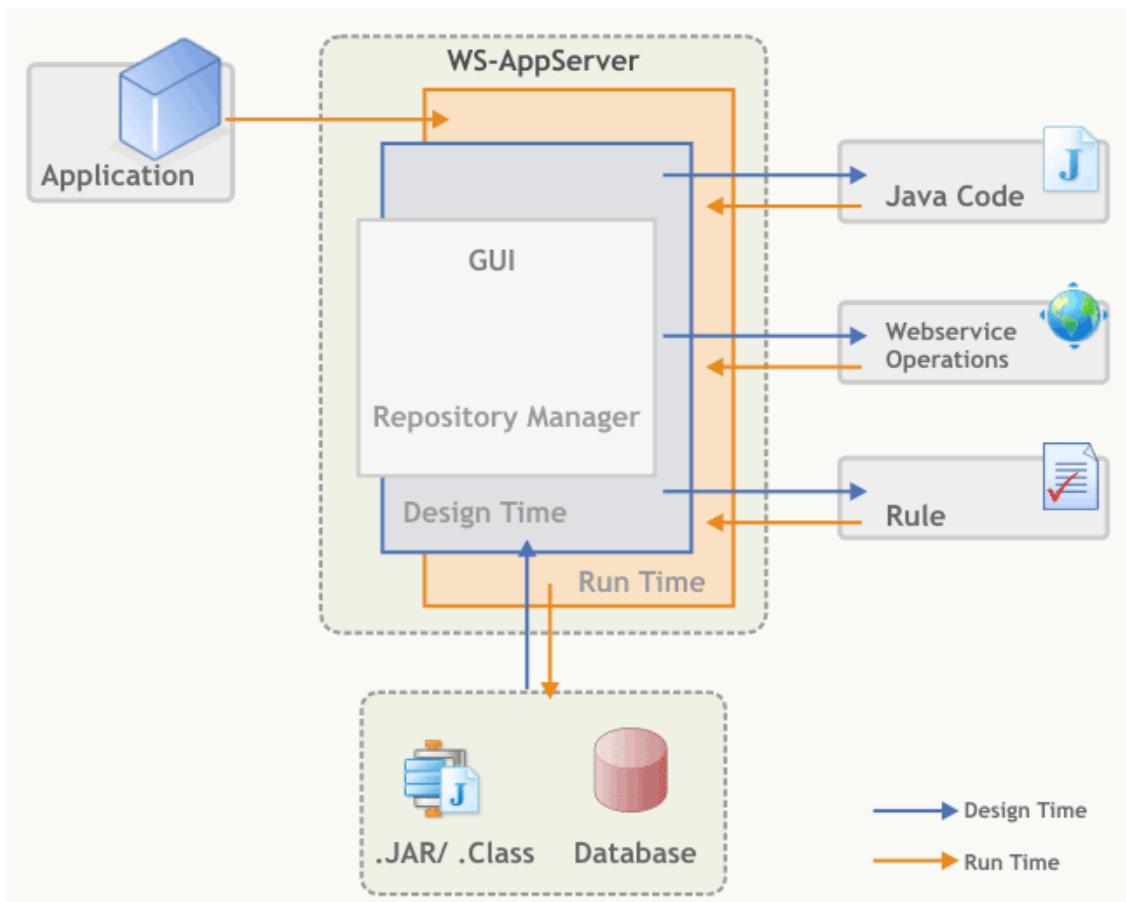
Note: Object Template is deprecated in BOP 4.1. Use WS-AppServer Custom class instead. See [Using WS-AppServer Custom Class as an Alternative to Object Template](#) for more information on the procedure and guidelines to use WS-AppServer Custom Class as an alternative to Object Template.

Important: BOP 4.1 still supports the existing templates upgraded from earlier AppWorks Platform releases to ensure backward compliance. However, you cannot create new Object Templates any more. Therefore, it is recommended to adopt the suggested alternative to Object Templates: WS-AppServer Custom class.

Modeling relational data

Modeling relational data in AppWorks Platform begins with extracting database metadata and storing it in the form of data models. In this context, WS-AppServer, which functions as an independent processing layer between a relational data source and a UI client, reads the database details, [extracts metadata](#), and creates data models. Using the packaging feature (known as [WS-AppServer Package](#)), WS-AppServer classifies and organizes the application content into well-defined components.

WS-AppServer plays a vital role in auto-generating the core business logic that is required by the application, extending the business logic, and exposing it as a Web service. Depending upon the data source and the tables that you have used to model your application, WS-AppServer determines the relations among classes and creates those relationship links. Thus, each model (a data object) is available with all its relational mapping that helps in the generation of Java logic to be used in applications. Using the data in these object models, WS-AppServer generates the business logic in Java, and maps the created Java class names to the database names, using the object-relational mapping technique. In absence of a database, WS-AppServer facilitates generation of classes on the basis of the custom queries. These are called [custom classes](#), which enable the developer to write the logic so that it suits the application's objectives. WS-AppServer also generates the Web service interface, Web service operations, and the XML schema for each model. You can also define and execute rules on these models. All these are later used to expose the business logic as Web services and load them into LDAP. Thereafter, these Web services are used by the application during run-time.



WS-AppServer also facilitates validation by employing database constraints (field checks, checks on relations between tables, and so on), and other procedures for data processing and/or retrieval. It also has the capability to [interact with other web services](#) and reuse them in combination with the internally developed Web services.

Using WS-AppServer package to model relational data

A WS-AppServer package contains data models in the form of Java classes, methods, and attributes, which are used in building a WS-AppServer application. It also contains information on the relationship that exists between different models. Package contains the namespace information, which is unique. The same information is carried over to the models, Java classes, methods, and attributes that reside within the package. As a result, the contents of one package are easily distinguished from the contents of another package.

You can create a WS-AppServer package in several ways:

- Create an empty WS-AppServer package
- [Create a WS-AppServer package directly on database](#), using [Database Metadata](#)
- Create a WS-AppServer package while generating Web service operations on database, using Database Metadata.

After creating a WS-AppServer package, you can perform the following activities:

- Load the package with database details, using Database Metadata
- [Generate Java code for the models](#)
- Generate Web service interface on the models
- [Add Custom Methods to the model](#)
- [Create custom classes using Object Layout](#)
- [Create custom classes using Query builder](#)

Creating database metadata

A Database Metadata represents the structure of a database. It uses the WS-AppServer Service to connect to the database and get the details. A Database Metadata provides direct access to the database and helps in quick generation of Web service. It also forms the basis for modeling WS-AppServer data objects.

Before you begin:

- You must have the role of Developer to create the Database Metadata.
- You need to create a WS-AppServer Service using WS-AppServer Application Connector that is configured to a relational database, in the same organization where you are creating the Database Metadata. You cannot use the WS-AppServer Service that exists in another organization.

While creating the WS-AppServer Service Group, ensure that the WS-AppServer Runtime Method Set Web service interface is selected on the Provide Service Group Details page of the wizard. This is mandatory. You may also attach other Web service interfaces depending upon your requirement.

To create database metadata:

1. [Select a starting point](#) and click  (Database Metadata Modeler).
2. Select the **WS-AppServer Service** from the list.
3. Click **Search in database** to retrieve all DBObjects (Tables, Views, and Procedures).
4. To retrieve DBObjects of a specific type in the pattern, click **Search in database**.
To get DBObjects with Order in their names, the pattern should be "Order".
5. Clear the **Load Related Tables** option to retrieve the selected tables only.
By default, the related tables of the selected tables are also retrieved.
6. Select the required DBObjects and move them to the Workspace using **>>** and **>**.
7. Click **Save**.
The Save Document dialog box closes and the database tables (represented by) appear on the right pane of the window.
8. Close the Database Metadata window.

The Database Metadata is created with the provided name and is stored at the selected location.

After you complete this task:

1. Go to the required  (database table) and click  (Extend Menu) > Actions and select [Generate Custom Web Service Operation](#) or Reload Database Metadata.
2. You can do any of the following on the toolbar of Database Metadata:
 - Click  to load tables from database.
 - Click  to create a WS-AppServer Package.
 - Click  to generate Web service operations.
3. On the left pane of the Database Metadata, you have an option to select the **WS-AppServer Service** and view its contents.

At any point, to view the tables or views available in a Database Metadata, do one of the following:

- In the Workspace Documents (Explorer), expand the Database Metadata.
- Double-click the **Database Metadata** and on the toolbar of Database Metadata window. The tables and views appear in a pane on the right side of the window.

At any point of time, to switch between Database Metadata abstracted by different WS-AppServer Services, do the following:

1. In the left pane of the **Database Metadata** window, select the **WS-AppServer Service** from the list.
All the corresponding contents are loaded and refreshed in the Children - Database Metadata pane.
2. Click **Save**.
The Database Metadata displays the contents from the selected WS-AppServer Service until it is changed again.
3. In the **Workspace Documents** (Explorer), expand the **Database Metadata** to refresh its contents (Tables, Views, Procedures).

Creating a WS-AppServer package from database

Creating a WS-AppServer package from database is a one-time activity to create data models, Java code, and Web service Interface on the database tables. This procedure saves time and effort when compared to creating an empty WS-AppServer Package and later updating it with database details.

Before you begin:

- You must [create a Database Metadata](#) before creating a WS-AppServer package from a database.

To create a WS-AppServer package from database:

1. Select one of the following starting points:
 - In **Workspace Documents** (My Recent Documents), point to  (Database Metadata), click  (Extend Menu) and select Actions > Generate WS-AppServer Package.
 - In **Workspace Documents** (Explorer), open **<solution>** > **<project>**, right-click  (Database Metadata) and select Generate WS-AppServer Package. The WS-AppServer Package Generation Wizard opens, displaying the name of the <database metadata> on its title bar.
2. Provide the necessary details on the wizard to create the WS-AppServer package.
 - a. You need to enter details on several pages. Click **Next** on each page to move to the subsequent one. Continue until you complete entering information on the wizard.
 - b. The WS-AppServer package containing the Schemas, is generated with the provided details, and placed in the folder with the name provided in the wizard. If you generated the Java code, it is also created and placed in a separate folder.
 - c. The generated Schemas and the WS-AppServer package are tightly linked. For more information on this, see topic Using WS-AppServer Package to Model Relational Data.
 - d. The Java Package Name and Namespace appear on the WS-AppServer Package Interface when you open it. For each model, you will see a Mapped Name (usually the name that corresponds to the actual database table name) and a Class Namespace (the name of the Package Namespace). If you do not want to retain the default names, you may replace these with names of your preference. The generated Java code and Web service operation names will reflect the Mapped Name given for the model. You can edit the Mapped Name for Model Attribute in the Attribute Properties pane, which will be reflected in the corresponding method parameter names (both in Java code and Web service operation).

The tables from a Database Metadata are modeled as Standard Classes and stored in the WS-AppServer Package.

If you selected related database tables while creating the WS-AppServer Package, the Standard Classes in the package retains those relations. As a result, you will not be able to delete any Standard class that has a relation to another Standard Class. Also, if a Custom class inherits, aggregates, or is derived from a Standard class, you cannot delete that Standard class. You can delete a Standard class from the WS-AppServer Package only if the tables were not related at the time of WS-AppServer Package creation and if they are not used by any Custom class.

To delete a Standard class:

- Right-click the class in Models group box and select **Delete**.

This action, however, deletes only the Java classes and not the Web services attached to the class. If required, you must delete the Web services manually.

After you complete this task:

- Publish the generated Java code so that the code is compiled and the Java classes are packaged in a .JAR file.

By default, the .JAR file is stored at <AppWorks Platform_installdir>\<instance name>\bsf\runtime\deploy and the location is added to the classpath of the WS-AppServer Service.

The path details are mentioned against the wsappserver.deploy.folder property in the wsapps.properties file located at <AppWorks Platform_installdir>\<instance name>\components\wsappserver\config.

An Administrator can modify the path, if required.

For example, in Windows, wsappserver.deploy.folder = C:\PROGRA~1\Cordys\DEFAUL~1\bsf\runtime\deploy and in Linux wsappserver.deploy.folder = <AppWorks Platform_installdir>\bsf\runtime\deploy. While changing the path, keep the folder separators as they are in the default path.

Creating an empty WS-AppServer package

Before you begin:

- You must have the role of WS-AppServer Developer in AppWorks Platform.

To create a package for modeling data using WS-AppServer:

1. Select a starting point and click . The WS-AppServer Package wizard opens.
2. Provide the **Name, Description, Namespace**, and the **Java Package Name**.
3. To provide the Java Archive Folder Name, click  to browse and select a folder in the project.
 - Ensure that the folder you select is empty and will eventually contain only the generated JAR file. Otherwise, it may cause publish failures, because according to the publish logic of Java Archive Definition, the JAR will also include all the folders and documents (including other AppWorks Platform documents) that exist in this folder.
 - Ensure that the folder or path names that you specify do not contain any spaces.

The selected folder appears in the field.

4. Provide the Java Archive Name that will contain the build output of the Java sources, the .class file, and it will be created in the selected Java Archive Folder.
5. Click  to browse and select the Location to save the WS-AppServer Package.

Note: Skip this step if you are in the Workspace Documents (Explorer) view because this field appears on the wizard only if you are creating the WS-AppServer package in the Workspace Documents (My Recent Documents) view. The path where the WS-AppServer Package will be created appears in the Location field.

6. Click **Finish**.

The WS-Appserver Package wizard closes and the WS-AppServer Package window opens.

The WS-AppServer Package is created with the provided details and is stored at the selected location. The details - Java Package Name and Package Namespace appear on the WS-AppServer Package window. You may modify these later, if required.

Modeling data transformation

Data is often received in different formats and needs to be converted into other formats supported by the target applications. Complex mapping rules need to be defined while these formats are converted. Facilitating the data integration and reuse from diverse applications and multiple data formats is a challenge for the enterprise computing arena. Data transformation addresses this challenge.

Data mapping is the first step in data transformation. Data map is a model that captures the transformation logic between the source and target data models. It is the process of mapping elements of a source data model to elements of a target data model. Data transformation converts data from a source data format into a target data format using XQuery expressions or eXtensible Stylesheet Language Transformations (XSLTs).

The Data Transformation Modeler provided by AppWorks Platform, enables the visual transformation of data from one format to another through a drag-and-drop mechanism and engages the power of XQuery functions and operators. AppWorks Platform Data Transformation Modeler's thin client-based environment offers extensive design time capabilities. It allows you to map, simple as well as complex data between two source and target data models, and provides a library of standard mathematical, scientific, and logical functions to manipulate data during the transformation. [Creating a Data Transformation Model](#) is a one-time operation, and the same model can be reused for subsequent data transformations between business objects of the source and target schemas.

Data transformation models can also be built as Web services which are treated as resources and reused across multiple processes and integration solutions. They can be used in a business process as a Web service or can be used to transform data received as an incoming message into the business process. The target can be a node, an attribute, or a function block. You can also import an existing XSLT into the Transformation modeler for data transformation.

This section contains the following topics:

- [Creating a Data Transformation Model](#)
- [Testing a Data Transformation Model](#)
- [Transforming Data using XSLT Functions](#)
- [Generating a Web Service on a Data Transformation](#)

Creating a data transformation model

Data mapping is the first step in data transformation. Data map is a model that captures the transformation logic between the source and target data models. It is the process of mapping elements of a source data model to elements of a target data model. Data transformation converts data from a source data format into a target data format using XQuery expressions or eXtensible Stylesheet Language Transformations (XSLTs).

Before you begin:

- Create the required source and target (object templates or schema fragments) to be mapped.

Depending on how you access the modeler, the way the objects are accessed differs:

- If you access the modeler from the Workspace Documents (Explorer), you can drag the source or target from the workspace explorer tree.
- If you access the modeler from the Workspace Documents (My Recent Documents) view:
 - On the modeler toolbar, click  (quick access menu).
 - On the side bar, select  (Import).
 - Drag the required objects to the **Source** or **Target** as needed.

To create a data transformation model:

1. Click  (Access Data Transformational Modeler) to create a data transformation model. The Data Transformation modeler page appears.
2. Drag the required source schema fragment or object template from the project tree to the **Source Template** pane of the **Modeler** tab. The schema of the object appears on the modeler.
3. Drag the required target schema fragment or object template onto the **Target Template** pane of the **Modeler** tab. The Transformation Modeler page displays the schema of the selected source and destination templates.
4. Map (linking the elements) the elements between the source and target templates.
 - a. Select the element on the source tree.
 - b. Hold the **CTRL** key, and select the corresponding element on the target tree. A line appears connecting the mapped elements in the source and target templates.
 - c. While transforming the data, if required, you can have the target XML contain elements even when they are not mapped.
 - Right-click the required element of the **Target XML**, and select **Mandatory**.
 - Right-click and select **Not Mandatory** to remove this option.

5. To group the mappings using the **Group** feature, click **Create New Group** from the **Group** list.
The Add Group dialog box opens.
6. Provide the group name, and click **OK**.
A group is created, and all the mappings created after this point will be placed in this group.
A group allows you to identify a collection of maps with a unique identity. This feature is especially helpful when you have to manage a large number of maps. Grouping reduces the clutter in such cases. You can switch between groups or choose to display all the mappings by selecting the appropriate option in the Group list. You can move a mapping along with its associated functions from one group to another by right-clicking the link, and selecting Move to Group.
7. If needed, transform the data using the XSLT function library as described in [Transforming Data using XSLT Functions](#). Alternatively, you can directly edit the XSLT for the transformation model.
For more information on the XSLT functions, see [XSLT Functions](#).
8. If required, in the **Edit Model Source** tab, edit the XSLT code for the transformation model.
 - a. Click **Validate Model Source** to check the accuracy of the XSLT code.
The XSLT code of the transformation model is validated.
 - b. You can debug the XSLT code, if validation on the transformation model fails.
9. Test the transformation model through the Test Model tab as described in [Testing a Data Transformation Model](#).
10. If required, to view the XSLT source, click the **Model Source** tab.
The generated XSLT code for the transformation model can be viewed in the Model Source tab.
11. Apply the required changes and save.

Caution: After the transformation model is created, if there are any changes made to the elements of the schema fragments, the changes are not automatically reflected in the modeler. For instance, if the name of an element in the schema fragment is changed from ID to Emp ID, then the change is not automatically reflected in the transformation modeler. You must go to the **Edit Model Source** tab and apply the relevant changes to the elements manually.

12. **Save** the data transformation model.

The data transformation model is created and is attached to the corresponding project.

You can identify the mappings:

- To which Target is the Source Element mapped
- From what Source is a Target mapped

1. Click **Source** or **Target** elements on the tree.
The mappings are highlighted for easy identification.
2. You can also identify mappings by clicking on a function.
This action highlights the TO and From functions.

Transforming data using XSLT functions

To transform date using XSLT functions:

1. Access the Data Transformation Modeler.
The Data Transformation modeler page appears.
2. Right-click the **Map Canvas** (middle) pane of the **Modeler** tab, and select the required **<Function>** library.
For more information on the XSLT functions, see [XSLT Functions](#).
The selected **<Function>** icon is displayed on the Map Canvas.
You can view instructions on using the modeler, functions, or the properties pane, at the bottom of the modeler.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code can use the equivalent in-house javascript functions and their wrapper implementations. If users want to use the functions as specified in the XSLT 2.0 and XPath 2.0 specifications, they can create their own transformation scripts in the Model Source. However, these will not reflect on the Map canvas in the modeler.

3. Right-click the **<Function>** icon and select **Properties**. Alternatively, you can double-click the **<Function>** icon to open the Properties pane.
The Properties pane appears below, displaying a table in which the parameters and values of the selected function have to be specified.
4. Select the element in the source template, hold the **CTRL** key, and select the required parameter in the Properties pane.
5. You can also specify a custom value as the parameter.
 - a. Type the custom value in the box displayed in the parameter row.
 - b. If you are using the Web service lookup function, select the element in the **Source** template, hold the **CTRL** key, and click the required parameter in the Web service.
The element is displayed in the value column of the function block.
6. Select the **<Function>**, hold the **CTRL** key, and select the target element.
A line connecting the **<Function>** and the target element appears. This line indicates that the output value, based on the inputs to the **<Function>** from the source template, is mapped to the selected target element. The target can be a node, an attribute, or a function block.

7. Save the transformation.
8. To add another XSLT function, repeat Steps 2 to 6.
9. To delete a function, right-click the function, and click **Delete**.

The data is transformed using the XSLT function library

Extending custom functions

The default XSLT functions that are available with AppWorks Platform address most data transformation requirements. However, if you need to build new functions with specific logic, you can create custom mapping functions.

Modify the following files to create custom data mapping functions:

- `funcinfo.xml`, which contains the mapping logic
- `functionsindex.xml`, which contains information to display the function name in the mapCanvas pane

To extend custom functions:

1. On the Welcome page > My Applications, click  (XMLStore Explorer).
The XMLStore Explorer window opens, displaying the Collection folder containing all the files and directories in the XMLStore.
2. Expand the Collection folder.
The various folders containing objects of the ISV or product are displayed in a tree structure.
3. Expand the **datatransformation** folder, open the `funcinfo.xml` file, and place the custom mapping logic of the function at the location shown in the following example:

```
<LibFunction>...<Function Attr="Scientific" ID="20">
    <FunctionName>Arc Cosine</FunctionName>
    <FunctionType>JavaScript/JAVA</FunctionType>
    <Expression>fnScientificArcCosine(iNum)
        <!-- In case of Java Function type give the class name including the
package name -->
    </Expression>
    <Arguments NoOfParam="1">
        <Param DataType="Number" Required="1"/>
    </Arguments>
    <SourceCode>
        <!--your logic implementation for the function goes here See the example
below --> function fnValueMapping() { var TargetValue=''; var SourceValue =
fnValueMapping.arguments\[0\]; var NumberOfValuePairs =
fnValueMapping.arguments.length - 1; var i = 1; while (i &lt;
NumberOfValuePairs) { if (fnValueMapping.arguments\[i\] == SourceValue) {
TargetValue = fnValueMapping.arguments\[i+1\]; break; } i = i + 2; } if (TargetValue
=='') { if (fnValueMapping.arguments[NumberOfValuePairs-1] == 'DEFAULT') {
TargetValue = fnValueMapping.arguments\[NumberOfValuePairs\]; } } return
(TargetValue) } </SourceCode>
```

```
<Info>
    <!--Any url of the html page for help put it here .See the example
below.-->
/cordys/documentation/onlinehelp/mergedProjects/cordysorchestratorfordevelopers/usin
gvaluemapping.htm#About Value Mapping </Info>
</Function> . . .</LibFunction>
```

4. Save the `funcinfo.xml` file containing the new custom function logic.
5. Open the `funcinfoindex.xml` file and place the group information that contains the mapping functions, at the location shown in the following example:

```
<categories> . . .
    <!-- Add the group name for your custom function, for example-->
    <eibus:menuitem submenu="stringfnsubmenu">String</eibus:menuitem>
    <!-- where, stringfnsubmenu is the Id for the context menu (listed in the next
step) and String is the group name --> . . .</categories>
```

6. Place the user interface related code at the location shown in the following example:

```
<categories>
. . .
</categories>
<functions>
    <eibus:contextmenu id='stringfnsubmenu' style='display:none'>
        <eibus:menuitem onclick='registerFunction(20)'>Arc Cosine</eibus:menuitem>
        <eibus:menuitem onclick='registerFunction(21)'>LowerCase</eibus:menuitem>
    </eibus:contextmenu>
</functions>
```

7. Save the `funcinfoindex.xml` file.

The custom mapping functions are ready for implementation.

Ensure that the value you provide for `registerFunction` in the `funcinfoindex.xml` file is the same as that of the Function ID in the `funcinfo.xml` file. The implementation will fail if there is a mismatch.

Testing a data transformation model

Before you begin:

- The data transformation model that is to be tested must have been created already.

To test a data transformation model:

1. Open the [Data Transformation Modeler](#) and select the required data transformation model.
The selected data transformation model appears on the Transformation Modeler page.

2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the **Source XML** pane, type appropriate values for the source elements.
Alternatively, you can click **Fill Source Data**, which assigns sample string values to the source elements. Make changes to these values, if required. The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.

The data is transformed and displayed in the Transformed XML pane indicating the completion of the process.

Creating a content map

A content map is a common repository in AppWorks Platform used to maintain pairs of source and target values. This is very useful in scenarios where a large set of values need to be transformed, and mapped to the target. AppWorks Platform provides its own repository for storing content maps, thus eliminating the need to use external tools.

Content maps are meant to store static data and do not dynamically change as in the case of Web services or other models used in Data Transformation. As many content maps as needed can be created, bundled, and deployed along with the ISV applications. While creating a data transformation model, you can look up the content repository using an XSLT function(Local Lookup), and map the values. For example, let us consider the instance where data needs to be transformed between two disparate systems. The Source system uses a BaaN application and the Target system uses an SAP application. The value for an entity like itemcode is represented in the Source system as 100 but the same in the Target system is represented as IC100. You can create a relation between these two systems in a content map and use it in the Local lookup function and transform the data.

To create content maps:

1. Access the Content Map  (modeler) to create a content map.
The Content Map modeler page appears, displaying a table.
2. To add a new value pair, click and do the following:
 - a. In **Entity**, type an ID that identifies the source and target values. For example, you may type itemcode as the Entity.
 - b. In **Source System**, type the name of the system which consists the source value.
For example, provide the Source System as BaaN.
While mapping data, the XSLT function (Local lookup) enables you to specify the source system from which the value pairs are to be picked up.
 - c. In **Source Value**, type the value for the entity in the source system.
For example, provide the Source Value as 100.
 - d. In **Destination System**, type the name of the destination system for the target value pair.
For example, provide the Destination System as SAP.

- e. In **Destination Value**, type the value for the entity in the destination system. For example, provide the Destination Value as IC100.
3. To add more value pairs, repeat Step 2.
4. Click **Save**.
The Save Document dialog box appears.
5. Type a **Name** and **Description**.
6. Click  (Lookup) to browse and select the location to save the Content Map.
7. Click **Save**.
The Save Document dialog box closes.

A content map is created for the selected value pairs. When the itemcode in the Source Template is provided as 100 in the local lookup function, the itemcode in the Target Template is transformed to IC100.

To use this content map in local lookup function, you must publish it:

- Right-click the content map and select [Publish to Organization](#).

Chapter 8

Modeling business rules

Rules define the way a business object reacts to a situation, and eventually charts the course of a set of actions. For example, business rules determine the logic behind the premium calculations by the insurance companies, risk management by banks, portfolio management by finance companies, pricing levels by retailers, and so on.

AppWorks Platform provides a rich set of features for modeling and executing rules. The runtime execution of rules handles the evaluation and execution of conditions and actions based on input data XML. The expressions for conditions and actions are defined based on XPath 1.0 compliance. The Rule Engine evaluates rules in two phases; it evaluates constraint rules before the objects are committed to the database and the Business rules are used to validate objects after a transaction is committed to the database. It provides a set of built-in data types to facilitate modeling of complex rules and enables you to invoke Java program constructs from rule expressions.

Rule Modeler	AppWorks Platform provides a simple-to-use and highly intuitive rule modeler, to facilitate the business user create the rules driven by the business requirements.
Decision Table	Decision Tables allow the user to define a set of related rules with conditions and actions that are evaluated based on the given data set as input. A decision table can be built based on a business object and each decision table can contain one or more rules acting on that business object. When you have a large number of rules to implement and are working with structurally similar rules, decision tables are very efficient in handling them. There are two ways of using a decision table. One is to generate a Web Service and reuse it. The other way is to directly use the Decision table in a business process as an activity. When a Decision table is used directly in a Business process model, it allows the users to configure monitoring options on the decision table activity to get the graphical representation of decision table execution during runtime.
Templates	When there is a need to use a complex condition in a decision table, you can use a Condition Template . For example, if you have used a particular condition in a rule, and want to use the same condition in another rule, you need not build that condition from scratch. You can

	open a condition template and define the variables that are required. Similarly, to reuse an action in other rules, you can use an Action Template .
Rule Groups	You can organize and group functionally similar rules for better management and ease of retrieval in a rule group. Each rule group has a priority associated with it thus enabling you to prioritize the execution of rules in that rule group. All rules and decision tables should be linked to a rule group.

Rules

Rules dictate the manner in which a business object reacts to a situation, and eventually charts the course of a business process. You can model the behavior of data or business objects using rules. Each process in a business can use rules to govern intra- and inter-enterprise collaboration. During the implementation of a process, rules that apply to business objects determine the way business functions are engineered towards the completion of the process.

Note: Business Rule Template is dropped in Cordys BOP 4.1. But it is supported for the existing templates upgraded from earlier AppWorks Platform releases to ensure backward compliance. However, you cannot create new Business Rule Templates any more.

Rule groups

A rule group allows you to organize functionally similar rules for better management. Rules are grouped for ease of retrieval. Each rule group has a priority associated with it thus allowing you to prioritize the implementation of rules in that rule group. All rules are linked to a rule group and rules cannot be shared across multiple rule groups.

Rule types

AppWorks Platform supports the following types of rules that are defined while [creating business rules](#).

Type of Rule	Description
Constraint	<p>This type of rule is used to validate objects before they are persisted in CoBOC. It works within the context of a transaction. The rule may relate to business constraints and partner agreements that come into force at the beginning or during the data modeling process.</p> <ul style="list-style-type: none"> ▪ This type of rule is recommended, when defining internal and assignment actions. ▪ You can use the request object only in Constraint rules. Therefore, if you have created a constraint rule, and want to change the rule type,

Type of Rule	Description
	ensure that references to the request object are removed from the rule definition.
Business	This type of rule is used to determine the behavior of objects after a transaction is committed to the database. The business rules are executed after the constraint rules. This type of rule is recommended, when defining external actions.
Passive	This type of rule allows you to fire a rule from within a rule, thus allowing you to nest rules. Therefore, if a rule is set as a passive rule, then that rule runs only if it is triggered from another rule. This rule has no scope outside the context of a rule that has triggered it. This rule is available only when using a fire rule action.

Creating a business rule

During the implementation of a process, rules that apply to business objects determine the way business functions are directed towards the completion of the process. AppWorks Platform provides a highly efficient, simple-to-use rule modeler that enables the business user to create the rules required by business.

Before you begin:

- A schema fragment, or WS-AppServer model must exist prior to the creation of a rule.

Though rules can also be created on object templates that were migrated from the previous versions of AppWorks Platform, use WS-AppServer classes. See [Using WS-AppServer Custom Class as an Alternative to Object Template](#) for more information on the procedure and guidelines to use WS-AppServer Custom Class as an alternative to Object Template.

Depending on how you access a document, creating a rule and accessing the business object can be done in various ways.

To create a business rule:

1. Define the Rule Behavior by setting the conditions and actions in the rule modeler.
2. Click  on the toolbar.
Alternatively, you can double-click in the Rule Definition pane to open the Rule Properties Sheet.
A Property Sheet is displayed under the Rule Definition pane.
3. You can set the properties for the rule, such as, the version of the rule, the name of the rule group to which it is attached, set the rule as a mutually exclusive rule or an overriding rule. For more information on the rule properties interface, see [properties of a rule](#).
4. Click **Save**.
The Save Document dialog box opens.

5. Provide **Name** and **Description**.

6. Click  to browse and select the location to save the rule.

The folder path where the rule is created appears in the Save in Folder field.

A business rule is created.

Note

- Before publishing the rule, ensure that you have a [rule group](#) created for logical grouping of rules.
- You can test the execution of the rule, using the Rule Test Tool.
- Ensure that the Rule Management service is started before you publish or test the rule.

Properties of a rule

Rules are associated to a rule group and are triggered upon certain actions. They are fired based on their priorities and can be set as mutually exclusive or overrides and so on. These properties can be set through various tabs on this interface.

The following table describes the various fields on the **Basic Properties** to be assigned while [creating a rule](#).

Version	The current version of the rule. This value cannot be edited.
Rule Group	The rule group to which the rule belongs. While defining the rule on a rule group, this field is automatically filled. While defining rule on an object template or a schema fragment, drag the required rule group from the solution explorer onto the Rule Group text box.
Rule Type	Type of rule. You can select the following rule types from the list: <ul style="list-style-type: none">■ Constraint■ Business■ Passive For more information on the types of rules, refer to Types of Rules .
Effective From	The date from which the rule comes into effect. For more information on specifying the dates, see Setting effective and expiry dates for rules .
Expires On	The date from which the rule is no longer effective. For more information on specifying the dates, see Setting effective and expiry dates for rules .
Triggers	<ul style="list-style-type: none">■ If you want the rule to trigger when an object is inserted, select Insert.■ If you want the rule to trigger when an object is updated, select

	<p>Update.</p> <ul style="list-style-type: none"> ■ If you want the rule to trigger when an object is deleted, select Delete.
--	--

The following table describes the various fields on the **Additional Properties** tab that can be set while creating a rule.

Rule Template	<p>Name of the rule template this rule is based on. This is a Read-Only field and is only visible for the migrated rules that are already associated to a template rule.</p> <ul style="list-style-type: none"> ■ For migrated rules, if a rule is already associated to a template rule, you cannot de-link or change the association to another template rule. ■ For new rules, you cannot create association to a template rule.
Link	<p>Name of the rule that should be executed after this rule is executed when you have to trigger a series of rules. Only rules of the same type are displayed in the list. For instance, if you intend to link a business rule, the list displays only the related business rules that can be triggered.</p>
isActive	Select this check box to enable the rule.
Priority	States the priority based on which the rule is executed.

Example:

- When a schema fragment has 2 rules defined on it, the rule with highest priority is executed first. Also considered is the priority of the rule group that has the rules.
- When rules of 2 different rule groups are to be triggered, the priority of the rule group will be considered first. The rule in the rule group with the highest priority is executed first.
- Within a rule group, the priority of an individual rule is considered. Priority levels for a rule range from 1 (lowest) to 10 (highest). The default value is 5.

The following table describes the various fields on the **Namespaces** tab.

Prefix	Contains the prefix of the elements in the input schema.
Namespaces	Contains the namespaces of the elements in the input schema.

Mutex

You can set a particular rule (source rule) to be mutually exclusive (mutex) to one or more rules. This way, if the source rule happens to trigger first, and the condition is satisfied, the mutex rule will not trigger. Likewise, if the rule that is set to be mutually exclusive happens to trigger first, the source rule will not trigger.

- Click the **Mutex** tab and select the check boxes of one or more rules to be made mutually exclusive. For more information on Mutex rules, refer to Mutex.

Overrides

You can set a rule to override another rule. This way, even if a rule is acting on an object, you can choose to override that rule with another rule.

- Click the **Overrides** tab and select the check boxes of one or more rules that you want to override.
- Ensure that you have created the required passive rules under the template that you are working on.

Setting effective and expiry dates for rules

Consider the suggestions when setting effective and expiry dates for rules.

Note: The date specified for Expires On must be later than the date specified for **Effective From**.

To	Do
Set a rule that never expires	Leave Effective From and Expires On empty. The rule is effective from the current date.
Set the rule that is effective from the current date and time and expires on a specified date	Enter the current date for Effective From and a date other than the current date for Expires On. The effective date is set to the current date and time and the expiry date is set to 00:00:00 hrs on the specified date.
Set the rule that is effective on a specified date and expires on another specified date	Enter the specific dates for Effective From and Expires On text. The rule is effective from 00:00:00 hrs on the specified date and expires on 00:00:00 hours on the specified date. Note: If the rule is set to expire on the current date, it automatically expires at 23:59:59 hrs of the current date.

Testing rules using rule test tool

Rule Test Tool is used to simulate the execution of rules on an object. It provides a simple interface where you provide the details of the object and view the changes made to the object, when the rules are executed. The conditions along with the test results are displayed in the test execution results pane.

To test rules:

1. On the Welcome page > My Applications App Palette, click  (Rule Test Tool). Alternatively, you can right-click the required schema fragment in your workspace and select Test Rules. The Rule Test Tool window appears, displaying the Rule Details and Rule Test Results panes.
2. Click  next to the Select Entity text box. A Select Entity dialog box opens, displaying CoBOC Templates or WS-AppServer Classes on which the rules can be tested.
3. Do one of the following:
 - To test rules built on Object Templates, select CoBOC Templates. The objects tree containing all the object templates is displayed in the pane below.
 - To test rules built on WS-AppServer Classes, select WS-AppServer Classes. The objects tree containing all the WS-AppServer classes is displayed in the following pane.
4. Traverse and select the required template or class from the object tree and click **OK**. The object's XML is displayed in the Business Object text box below.
5. Select the type of rule, to be executed on the object, from the Rule Type list.
6. Select the type of operation, to be performed on the object, from the Operation list.
7. Select the **Execute Actions** check box, if you want the rule to perform external actions, like triggering a business process, invoking a Web service, or sending notifications.
8. Provide the required parameters to the XML in the Business Object text box and click the Test Rule button.
The results of the test are displayed in the Rule Execution Results pane on the Tree View tab, by default. The conditions that are satisfied are shown with a  attached to them and the conditions that fail are shown with a  attached to them.
9. Click **ruleset** to view the definition of all the rules on the object.
10. To view the definition of the rule, condition, or the action, click the corresponding tree item.
The rules details are displayed in the text box below.
11. Click **Changed Object** tab, to view the changes made to the object.
The object's details are displayed on the Changed Object tab.

The rule is tested.

Creating a business rule template

If you have existing business rule templates, they are supported when upgrading from a previous release of AppWorks Platform.

However, AppWorks Platform no longer supports creating new business rule templates.

Building a decision table

Decision tables present a simple way to model complex business logic (rules) by offering a simple user interface. This provides a concise, easy-to-read, and logically organized way of representing and querying data. Decision tables, which constitute business cases, associate conditions with actions to be performed.

The following procedures walk you through the procedures for building a decision table.

Before you begin:

- You can build a decision table on a schema fragment, or an object template, or a WS-AppServer model based on your requirement.
- You must have a rule group for logical grouping of decision tables.

To build a decision table:

Depending on how you access a document, creating a decision table and accessing the business object can be done in various ways. See [Accessing the Decision Table Editor](#) for more information

1. Click  (decision matrix) to set the properties of the decision table.
The Conditions, Actions, and Attributes matrix opens.
For more information, see [Properties of a Decision Table](#). Alternatively, you can set these properties before publishing the decision table.
2. Drag the required attributes from the **Select Attributes** section to the **Attributes** column of the **Conditions** section.
3. Click the attributes and set the properties, such as name of the attribute and how it should be displayed in **Name** and **Display As** of the **Properties** pane.
4. Click  (Select Action) on the toolbar.
The Select Actions pane opens, displaying the Repository Actions.
5. Drag the required actions from the **Select Actions** pane to the **Actions** section.
The Actions section is populated with the selected actions. Only external actions, abort transaction, and assignment actions are supported for a decision table.
6. Click the selected **<action>** in the **Actions** section and set the parameters for the actions in the **Properties** pane that is displayed on the right side.
See [Rule Actions](#), for more information on the various actions that can be defined.
7. To define a rule, click the rule header and set the properties, such as name of the rule and how it should be displayed in **Name** and **Display As** of the **Properties** pane that is displayed on the far right of the window.
8. Click the condition cell in the **Rule** column to set the condition and fill the Properties pane with the appropriate details for each rule.
For more information on the fields of the properties pane, see [Setting Properties of Conditions in a Decision Table](#).

9. Click the action cell in the rule column and set the properties of the actions in the **Properties** pane.
10. Ensure that you have selected **Enable Action** to trigger the specified action. See [Rule Actions](#) for more information on the various actions that can be defined.
11. If required, click the **Namespaces** tab.
The prefix and namespace of the elements are displayed in the Prefix and Namespace columns.
12. Click **Save**.
13. To add another rule to the decision table, click  (Rule) on the toolbar and repeat Steps 4 through 11.
14. To remove empty attributes and rules, click  (Delete Row) on the toolbar.
15. To rearrange the conditions or rules, click the rule header and use the  (Move) icons on the toolbar. Alternatively, you can right-click the required rule header and select **Insert Before** and **Insert After** options as required.
16. You can also duplicate a rule. Right-click the required rule header and select **Duplicate** from the context-menu.
A copy of the rule is displayed in the adjacent column.
17. Click the rule header and change the name of the rule.
18. Click  (Property) to show/hide the Properties pane.

The conditions and actions are set for the template, and the decision table is built.

- You can view this decision table in the Workspace Documents (Explorer), under the rule group to which it is linked.
- You can drag this decision table onto a business process and use it as an activity.
- You can test the decision table using the Rule Test Tool.
- Ensure that the Rule Management service is started before you publish or test the decision table.

Properties of a decision table

The following table describes the basic properties to be assigned while building a decision table.

Rule Group	Name of the rule group. This is automatically populated if you are building a decision table on a rule group. Click  to associate the decision table to a rule group or change another rule group.
Rule Type	Type of rule. You can select either Constraint Rule or Business Rule from the list. <ul style="list-style-type: none"> ■ A Constraint Rule is used to validate objects before they are persisted in CoBOC. It works within the context of a transaction. The rule may

	<p>relate to business constraints and partner agreements that come into force at the beginning or during the data modeling process. This type of rule is recommended, when defining Assignment or Abort actions.</p> <ul style="list-style-type: none"> ■ A Business Rule is used to determine the behavior of objects after a transaction is committed to the database. The business rules are executed after the constraint rules. This type of rule is recommended, when defining external actions, such as Invoking a Web service, Triggering a Business Process, or Sending Notifications. <p>Caution: Rule types do not have any impact if this Decision Table is used in a business process model.</p>
Priority	<p>States the priority based on which the decision table is executed.</p> <p>Example:</p> <p>When a schema fragment has two decision tables defined on it, the decision table with highest priority is executed first. Also considered is the priority of the rule group that has these decision tables.</p> <p>When decision tables of two different rule groups are to be triggered, the priority of the rule group will be considered first.</p> <p>The decision table in the rule group with the highest priority is executed first. Within a rule group, the priority of an individual decision table is considered. Priority levels for a decision table range from 1 (lowest) to 10 (highest). The default value is 5.</p>
Is Active	Selecting the Is Active check box allows the decision table to be executed at run-time based on the object's operation.
Triggers	<ul style="list-style-type: none"> ■ If you want the decision table to trigger when an object is inserted, select On Insert. ■ If you want the decision table to trigger when an object is updated, select On Update. ■ If you want the decision table to trigger when an object is deleted, select On Delete. <p>Caution: Triggers do not have any impact if this decision table is used in a business process model.</p>
Effective From	The date from which the decision table comes into effect. For more information on specifying the dates, see Setting effective and expiry dates for rules .
Expires On	The date from which the rule is no longer effective. For more information on specifying the dates, see Setting effective and expiry dates for rules .
Show Expression	Displays the expression of the corresponding conditions or actions selected. Click the required condition or action in the decision table. If the Show Expression field is selected, the expression is displayed in the text box below.

When the decision table is used in a business process, the decision table is executed based on the input and output actions of the preceding and succeeding activities. Hence, triggers as well as the type of rules set on the corresponding schema fragment or WS-AppServer classes are ignored.

Accessing the business object in decision table editor

Based on the starting point, there are two ways of accessing business objects in a decision table editor:

- Accessing business objects using the Workspace Documents (Explorer) view.
- Accessing business object using the Workspace Documents (My Recent Documents) view.

-	Accessing the Business Object in the Decision Table Editor from Workspace Documents (Explorer)	Accessing the Business Object in the Decision Table Editor from Workspace Documents (My Recent Documents)
Building a Decision Table on an Object Template	Select the required <object template> from the Workspace Documents (Explorer), right-click the template and select Add > Decision Table. The Decision Table Editor appears, displaying the schema of the object template in the Select Attributes section.	In the Workspace Documents (My Recent Documents), place the pointer on the <object template>, click  that appears, and select Actions > Add Decision Table. The Decision Table Editor appears, displaying the schema of the object template in the Select Attributes section.
Building a Decision Table on a Schema Fragment	<ol style="list-style-type: none"> 1. Expand the required XML Schema from the Workspace Documents (Explorer). 2. Select the required <schema fragment>, right-click the schema fragment, and select Add > Decision Table. The Decision Table Editor appears, displaying the schema in the Select Attributes section. 	<ol style="list-style-type: none"> 1. In the Workspace Documents (My Recent Documents), select the required <XML Schema>. The <XML Schema> window appears, displaying the schema definition properties. 2. Click  on the <XML Schema> tool bar to view the fragments of the schema. A pane appears on the right side of the editor, displaying all the available schema fragments. 3. Select the required <schema fragment>, place the mouse pointer on the <schema fragment>, click  that

	Accessing the Business Object in the Decision Table Editor from Workspace Documents (Explorer)	Accessing the Business Object in the Decision Table Editor from Workspace Documents (My Recent Documents)
		appears, and select Actions > Add Decision Table. The Decision Table Editor appears, displaying the schema in the Select Attributes section.
Building a Decision Table on a Rule Group	<p>1. Select the required <rule group> from the Workspace Documents (Explorer), right-click the <rule group> and select Add > Decision Table. The Decision Table Editor appears, displaying the properties page. The Rule Group field automatically populated.</p> <p>2. Specify Properties of a decision table on the Decision Table Properties tab.</p> <p>3. Drag the required object template or schema fragment from the workspace explorer to the Select Attributes section.</p> <p>Alternatively, you can click  next to Input Schema and select the required <object template> or <schema fragment> from the Select Schema Fragment dialog box. The Select Attributes section populates with the schema.</p>	<p>1. In the Workspace Documents (My Recent Documents), place the mouse pointer on the required <rule group>, click  that appears, and select Actions > Add Rule. The Decision Table Editor appears, displaying the properties page. The Rule Group field automatically populates.</p> <p>2. Specify Properties of a decision table on the Decision Table Properties tab.</p> <p>3. Click  on the decision table editor tool bar. A Quick Access Menu is displayed.</p> <p>4. Click  Insert on the sidebar. All the related artifacts are displayed on the Quick Access Menu.</p> <p>5. Drag the required object template or schema fragment from the Quick Access Menu to the Select Attributes section.</p> <p>Alternatively, you can click  next to Input Schema and select the required <object template> or <schema fragment> from the Select Schema Fragment dialog box. The Select Attributes section populates with the required schema.</p>

	Accessing the Business Object in the Decision Table Editor from Workspace Documents (Explorer)	Accessing the Business Object in the Decision Table Editor from Workspace Documents (My Recent Documents)
Building a Decision Table directly	<p>1. From the Workspace Documents (Explorer), right-click the <project> or <folder> where you want to create the decision table and select New > Decision Table. The Decision Table Editor is displayed.</p> <p>2. Drag the required object template or schema fragment from the workspace explorer to the Select Attributes section. Alternatively, you can click  next to Input Schema and select the required <object template> or <schema fragment> from the Select Schema Fragment dialog box. The Select Attributes section is populated with the schema.</p>	<p>1. In the Workspace Documents (My Recent Documents), click  New AppWorks Platform Model. The New AppWorks Platform Model dialog box opens, displaying all the models.</p> <p>2. Click  Decision Table. The Decision Table Editor is displayed.</p> <p>3. Click  on the decision table editor tool bar. A Quick Access Menu is displayed.</p> <p>4. Click  Insert on the sidebar. All the related artifacts are displayed on the Quick Access Menu.</p> <p>5. Drag the required object template or schema fragment from the Quick Access Menu to the Select Attributes section. Alternatively, you can click  next to Input Schema and select the required <object template> or <schema fragment> from the Select Schema Fragment dialog box. The Select Attributes section is populated with the required schema.</p>

Working with a condition template

A condition template represents the draft of a condition, which can be used as it is or can be modified and used. For example, if you have used a particular condition in a rule, and want to use the same condition in another rule, you need not build that condition from scratch. Instead, you can pick a condition template, fine tune it to your specifications, and use it to build the rule.

You can design conditions as straightforward expressions, or can design them to accept parameters to form the complete condition.

To work with a condition template:

1. From the Workspace Documents (Explorer), open the condition template editor.
The Condition Template editor opens.
2. Select the required <schema fragment> from the **Workspace Documents** (Explorer) and drag it to the Input Schema pane in the editor.
 - a. Alternatively, on the condition template editor toolbar, click  (Quick Access Menu).
 - b. On the Quick Access menu sidebar, click  (Insert) and drag the required <schema fragment> to the Input Schema pane in the editor.
The Input Schema pane is populated with the required schema fragment.
3. **Build the Condition Expression**, using the attributes from the required Business Objects and the functions from the Function Library, in the Expression Grid.
The expression should be a valid XPath expression.
4. To view the namespaces of the elements, click  on the toolbar.
A Namespace tab is attached to the Parameter Details appPallette, displaying the prefix and namespace of the elements in the Prefix and Namespace columns.
5. Click **Save**.

You can drag these condition templates to the Condition section of the decision table. When you use a condition template for setting the conditions in a decision table, you can select values from the set of parameters that are set in the condition template.

Building a condition expression

This task is performed while adding a condition to a decision table and [using the condition expression](#).

1. On the **Condition Template** tab, select the appropriate starting parenthesis for the expression from the **(** column.
The starting parenthesis for an expression depends upon the number of sub-expressions that will be evaluated within an expression.
2. In **Column One** of the **Expression Grid**, click the cell, and do **one** of the following:
 - To use a business object as the operand, drag the required attribute of the business object from the **Business Object** section into the **Column One** cell.
OR
 - To have the user input the parameter to the operand while designing the rule, do the following:
 - Type the variable name preceded by a colon in the **Column One** cell.
For example: :OrderID

- a. Press **tab** or click anywhere outside the cell.
The parameter details are displayed in the Parameter Details table
 - b. Type the appropriate label for the parameter in the **Description** text box.
This label is displayed to the user when designing the rule.
 - c. Select the [type of parameter](#) the user should use from the Parameter Type list box.
3. Click the **Operator** cell, select the required function from the **Function Library**, and drag the required operator into the **Operator** cell.
 4. Click the **Column Two** cell and repeat steps 2 and 3 specify the second operand.
 5. Select the closing parenthesis for the expression from the **)** column to close the first expression.
 6. Select **And** or **Or** from the **And/Or** cell if you want to add another expression. Ensure that the number of opening and closing parentheses match.
 7. Click  on the expression grid to add more condition expressions.
You can see the expression as you build, in the Current Expression box above the Parameter Details section.

The condition expression is built.

Types of parameters

The following table describes the types of parameters that can be used while building a condition template or an action template.

Type	Description
Any	Any value.
Boolean	Either true or false.
Business Object Attribute	Any attribute of a business object.
Date	Date in the format YYYY-MM-DDThh:mm:ss.s. For example: 2005-11-18 T 05:51:30.0
Email	Email in the format -name@domain.ext. For example: Jdoe@cordys.com
Expression	Expressions like order/quantity>100.
Float	Any float value.
Integer	Any integer value.
Long	Any long integer value.
Non Negative	Any value greater than or equal to zero.

Type	Description
Non Positive	Any value less than or equal to zero.
Quoted String	Any string within double quotes. For example: "xyz".
String	Any string value.
Metadata - Rule Name	Parameter that denotes a rule name. When you want to use the name of a rule, as a variable, in a condition template or an action template, you can set this parameter.

Working with an action template

An action template represents the draft of an action that can be reused. For example, if you have used a set discount action in a rule, and want to use the same action in another rule, you need not build that action from scratch. Instead, you can pick an action template, fine-tune it to your specifications, and use it to build the rule.

You can either design actions that can be used directly by the users or accept parameters from users to form the complete action. The parameters preceded by a colon (:) are taken as variables and the users using this template in a decision table can specify the values to these parameters.

To work with an action template:

- From the Workspace Documents (Explorer), open the required action template editor. The Action Template editor appears.
- From the **Action Type** list, select the type of action the rule must perform. Based on the actions selected, the fields in the Define action here pane change.

Abort	Type the message that is to be displayed when the transaction is aborted, in XML format.
Assign	Select the required <schema fragment> from the Workspace Documents (Explorer) and drag it to the Input Schema in the editor. Alternatively, click  (Quick Access Menu) on the action template editor tool bar. On the Quick Access Menu sidebar, click  (Insert) and drag the required <schema fragment> to the Input Schema in the editor. The attributes of the schema fragment are displayed in the Input Schema pane.
Invoke Web service	Click  next to Method Name and select the required <Web service> from the Web service dialog box. Alternatively, to select the business process, you can do any of the following:

	<ul style="list-style-type: none"> ▪ Select the required <Web service> from the Workspace Documents (Explorer) and drag it to the Message box in the Action - Invoke Web service pane. ▪ On the action template editor toolbar, click  (Quick Access Menu). ▪ On the Quick Access Menu sidebar, click  (Insert). ▪ Drag the required <Web service> to the Action - Invoke Web service pane in the editor. <p>The name of the Web service is displayed in the Method Name field and the request of the selected Web service is displayed in the Request box.</p>
Send Notification	<ol style="list-style-type: none"> 1. Provide the necessary details in the Define action here pane. Refer to Steps 3 to 6 of Using Send Notification for information on the fields of Send Notification action. 2. Click Save on the tool bar after completing the action.
Trigger Business Process	<p>Click  next to Process Name and select the required <business process> from the Business Process dialog box.</p> <p>Alternatively, to select the business process, you can do the any of the following:</p> <ul style="list-style-type: none"> ▪ Select the required <business process> from the Workspace Documents (Explorer) and drag it to the Message box in the Action - Trigger Business Process pane. ▪ On the action template editor toolbar, click  (Quick Access Menu). ▪ On the Quick Access Menu sidebar, click  (Insert) and drag the required <business process> to the Action - Trigger Business Process pane in the editor. <p>The name of the process is displayed in the Process Name field and the input message of the selected process is displayed in the Message box.</p>

For more information on the actions, see the specific [Rule Actions](#).

3. To enable the user to select values for the parameter while defining the rule action, do the following:
 - a. Select the required parameter in the message and type the parameter name preceded by a colon (:) in the expression.
For example, use the prefix : for the Order parameter as shown in the sample code. While using this action template in a decision table, users can specify the values to these parameters.

```
<GetOrderId xmlns="http://schemas.cordys.com/orders">:Order</GetOrderId>
```

- b. Click anywhere outside the cell.
The parameter details are displayed in the Parameter Details table.
 - c. Type the appropriate label for the parameter in the Description text box.
This label is displayed to the user when designing the rule.
 - d. Select the type of parameters to be used, from the Parameter Type list box.
The parameters are displayed in the Parameters pane.
4. On the toolbar, click  to view the prefix and namespace of the elements.
A Namespaces property sheet appears, displaying the prefix and namespace of the elements.
 5. Click **Save**.
The Save Document dialog box opens.
 6. Provide **Name**, **Description**, and **Location** for the action template.
 7. Click **OK**.

You can drag these action templates to the Action section of the decision table. When you use an action template for defining the action in a decision table, you can select values from the set of parameters that are set in the action template.

Creating a rule group

A rule group allows you to organize functionally similar rules or decision tables for better management. Rules are grouped for ease of retrieval. All rules are linked to a rule group and rules cannot be shared across multiple rule groups.

To create a rule group:

1. [Select a starting point](#) and click  (Rule Group) to open the Rule Group editor. The Rule Group editor appears.
2. Type a name in the **Name** field and description in the **Description** fields.
3. Select the required priority from the **Priority** list box and click  on the toolbar. Each rule group has a priority associated with it thus allowing you to prioritize the implementation of rules in that rule group. Priority levels for a rule group range from 1 (lowest) to 10 (highest). The default value is 5.
4. To associate the rule group to a rule or a decision table, do the following:
 - a. In the Rules section, click the **Click Here to Add** link.
The Select Rules window opens.
 - b. Select the required rule and click **OK**.
The selected rule is now associated to this rule group.
 - c. In the Decision Table section, click the **Click Here to Add** link.
The Select Window opens.

- d. Select the required decision table and click **OK**.

The selected decision table is now associated to this rule group.

5. Type a **Name** and **Description**.

6. Click **Save** on the tool bar.

The Save Document dialog box opens.

7. Click  next to the Location field, select the location to save the rule group, and click **OK**.

The Rule Group is created and is displayed in the Workspace Explorer. The rules and decision tables selected in the editor are associated to this group.

Chapter 9

Modeling schedules

In a real-time business environment, the ability to trigger processes or applications based on specific system responses or at a specific time is a critical requirement. AppWorks Platform facilitates modeling of schedules that can trigger time-based events or processes. AppWorks Platform provides an intuitive UI-based schedule modeler, enabling you to model different kinds of schedules and integrate them into your applications.

Schedules are broadly of two kinds:

- One-time schedule, which is executed only once
- Repeating schedule, which is triggered at specified periodic intervals

Types of schedules

AppWorks Platform facilitates very fine-grained scheduling of events, ranging from annual to hourly recurrences. The various kinds of schedules supported by AppWorks Platform are listed in the following table.

Category	Type of Schedule	Description
One-time Schedules	Run Now	Once created, these schedules are instantly executed.
	Duration	Can be set to execute after a specified duration.
Repeating Schedules	Hourly	Can be set to recur at a specified time every hour.
	Daily	Can be set to recur at a specified time every day.
	Weekly	Can be set to recur at a specified time every week.
	Monthly	Can be set to recur at a specified day every month.
	First Day of Month	Can be set to recur at a specified time on the first day of every month.
	Last Day of Month	Can be set to recur at a specified time on the last day of every month.
	First Week Day of Month	Can be set to recur at a specified time on the first weekday of every month.

Category	Type of Schedule	Description
	Last Week Day of Month	Can be set to recur at a specified time on the last weekday of every month.
	Fortnightly	Can be set to recur at a specified time every 15th day.

This section covers the procedures in creating and managing the following:

- [Creating a Schedule](#)
- [Deploying a Schedule](#)

Creating a schedule

By scheduling activities, you can ensure that a set of actions is executed at a set of preset times. For example, if you want to trigger a Web service that summarizes orders at the end of each business day, you can set a schedule to trigger at 17:30 pm every day.

To create a schedule:

1. Select a starting point and click  (Schedule modeler).
2. Provide **Name** and **Description** for the schedule.
3. In **Count**, provide the number of times that the schedule must be triggered. Depending upon the type of schedule selected in the Type list, the schedule is triggered the number of times specified in the Count box. For example, if you specify Count as three and set the Type as Every hour at minute, the schedule triggers three times, at every one hour on the minute specified from the time it is deployed. To run the schedule infinitely, provide the count value as -1.
4. Select **Auto Deploy** to have a schedule triggered when it is published or deployed.
 - If the schedule model with this option enabled is deployed or published, an instance of the schedule model is created when it is deployed or published into the organization.
 - If the schedule model with this option enabled is deployed into the Shared space, a schedule instance is not created. This model is displayed on the Inactive Schedules tab of Schedule Manager in all the organizations. The creation of the schedule instance, that is [deployment of the schedule](#) for this model is possible across multiple organizations.
 - If this option is not selected, the schedule models are visible on the Inactive Schedules tab of the Schedule Manager task upon publishing them. See [Deploying a Schedule](#) for more information.
5. In the schedule modeler, in the **Schedule Type** list, select the schedule type. Depending upon the schedule type selected, the schedule table below displays the

required columns accordingly. For more information on types of schedules, see [Types of Schedules](#).

6. Click  and specify the date and time at which the schedule is to be executed.

Use the following guidelines while specifying the date and time:

- The Hour and minutes must be specified in the 24 Hr format. For example, to indicate the time as 10:30 PM, specify it as 22 Hours and 30 minutes.
- The time specified must correspond to the time in the server you access.
- For hourly schedules, specify the minute of the hour in the Every hour at minute box.
- For daily schedules, select the time of the day in the Hour and Minute columns.
- For weekly schedules, select the day of the week from the Every Week Day At list, and provide the time of the day in the Hour and Minute columns.
- For monthly schedules, select the day of the month from the Every Month on the Day list, and provide the time of the day in the Hour and Minute columns.
- For first day of month schedules, select the time of the day in the Hour and Minute columns.
- For last day of month schedules, select the time of the day in the Hour and Minute columns.
- For first weekday of month schedules, select the time of the day in the Hour and Minute columns.
- For last weekday of month schedules, select the time of the day in the Hour and Minute columns.
- For fortnightly schedules, select the time of the day in the Hour and Minute columns.
- For duration schedules, specify the duration after which the schedule must be triggered in the Time text box.
- For an instant schedule (Run Now), provide the name of the schedule and the target information. The schedule is triggered and executed instantaneously upon deployment. For Run Now type of schedules, the Count is by default specified as 1.

At times, there may be delays in executing the schedule. The execution time of the schedule may not match the defined schedule time due to the following reasons:

- The schedule instances may be queued for execution, and based on the load at that moment, there may be a delay of a few seconds for the defined schedule to trigger.
- When the schedule triggers an instance of a Business Process Model (BPM), the request further may be queued at the Business Process service container. The time for the instance creation depends on the load on the service container.
- There may be network traffic or slow responses from the database. This behavior applies both for the schedules created directly or created implicitly when elements like Delay and Timeout are defined in a BPM.
- If a schedule is triggered after the 50th second of a minute, and it tries to instantiate an instance of a BPM, due to the queues and delays, the instance creation at the

Business Process service container can finally happen at the next minute. For example, even when a schedule to create a process instance is defined at 9/20/2013 9:23:52 AM, the actual process instance creation may happen only at 9/20/2013 9:24:02 AM.

7. Select the target from the Target Document Type list, and click  to select the required Web service or Business Process Model.
The name and definition of the selected Web service or Business Process Model is displayed in the Target and Request XML boxes respectively. You must provide the appropriate parameter values to the Web service or Business Process Model.
8. Click **Save**.
A Save Document dialog box opens if you are creating the schedule from the Welcome page > My Applications.
9. Provide the **Name** and **Description**.
10. Click  for Location.
11. Select a location to save this schedule, and click **OK**.

The required schedule is created and added to the project.

After you complete this task:

- To deploy the schedule, right-click the schedule document and select **Publish to Organization**.
- Ensure that the relevant schedule service container is running while publishing schedules to the organization.

The schedule is sent to the Inactive Schedules list. See [Deploying a Schedule](#) for more information.

Deploying a schedule

After a schedule is created, it must be deployed so that it is triggered at the specified time.

Before you begin:

- You should have created and published the schedule to the organization to deploy them.
- Ensure that the relevant schedule service processor is running while deploying schedules.

To deploy a schedule:

1. On My Applications, click  (Schedule Manager).
The Manage Schedule window opens, displaying all the published schedules on the Inactive Schedules tab, by default.
If same schedule model exists in both shared space and organization space, then only the model available in the organization space is shown in Inactive Schedules tab.

Tip: Clicking a schedule displays the name, type of schedule, number of times the schedule is to be instantiated , the ID of the template, and the target of the schedule in the Name, Type, Count, Template ID, and Target fields of the Schedule Details section respectively.

2. On the **Inactive Schedules** tab, right-click the required schedule and do the following:
 - a. To deploy the schedule template instantaneously, select **Deploy**.
 - b. To deploy the schedule at a specified date and time, select **Deploy with Time**.
A Set Deployment Time of Schedule dialog box opens.

Deploy At	Specify the start date for the schedule.
End At	Specify the end date for the schedule.
Hours, Minutes, Seconds	Specify time.
Run Once on	Select if you want the schedule to be executed as soon as it is deployed. Clear if you want the schedule to be executed only at the specified intervals.

- c. Click **OK**.

The inactive schedule is deployed and moved to the Active Schedules tab.

See Schedule Details for more information on the Schedule Details section.

The **Deploy with Time** option is disabled for **Run Now** schedule type.

The schedule is deployed successfully.

To execute a schedule instantaneously upon deployment:

- Select the required schedule and click **Execute Now** in the **Schedule Details** section.

To undeploy a schedule:

- Right-click the required schedule on the **Active Schedules** tab and select **Undeploy**.
The undeployed schedule is moved to the Inactive Schedules tab.

Chapter 10

Modeling human interactions

A business process can be termed as the process of arranging and describing the flow of activities required in reaching the desired output. Notifications and tasks are passed in the flow, from one participant or activity to another, to perform certain actions following a set of procedural rules. AppWorks Platform provides a solution that includes support for the human interaction in business processes, making it far easier for the users to participate in, measure, and facilitate processes that involve multiple players in the form of teams and roles.

The applications have support for human participation in a process through a simple interface called My Inbox. Every AppWorks Platform user is assigned an Inbox, which is used to send and receive these tasks and notifications. Users can view and execute tasks in the Inbox, which is designed for usability and productivity. Users can also collaborate on work processes through work lists that contain teams entitled to work on certain tasks.

These work lists can also be used to balance the workload by assigning or relieving teams dynamically. Users belonging to a work list can view all the tasks that are assigned to their work lists and claim the tasks they can work on. Every work list contains a manager who will be in-charge of maintaining the work list. A Work list manager has additional responsibilities, such as, assigning the tasks in the work list to users, forward tasks to other work lists when faced with some constraints, customize the view of tasks shown in the Inbox, and so on. As part of human interaction in a business process, the data passed between various activities can be modeled through [delivery models](#).

These delivery models can also be used to define various search attributes as well as design the tasks sent to the AppWorks Platform Inbox through an Inbox model or [Creating a delivery model](#) or an e-mail address through an e-mail model.

While the Inbox and E-mail models are for visual interpretation of messages only, you can create wrappers around the applications that use these models by adding a workflow library. For more information on creating wrappers and adding workflow library to the User Interface (UI) applications, see [Adding a Workflow Library to User Interface Applications](#). You can also access and work with applications developed using non-AppWorks Platform or external user interfaces in the AppWorks Platform Inbox.

Delivery mechanism

A task or notification can be delivered to the AppWorks Platform Inbox of a user, to an external email address, or to an application that uses the Inbox. You can create and define a [dispatch algorithm](#) that determines the receiver of the task at runtime. AppWorks Platform

provides a range of customizable dispatch algorithm options to suit specific business requirements.

Creating a delivery model

Delivery models are used to model the input message for a human interaction activity. The XForms or User Interfaces used as human Interaction activities are based on data models, Web services, or simple freeform controls, that contain the data required for the business process. Not all the activities in a business process may require the complete data in the XForm. AppWorks Platform enables you to model or segregate this data through the delivery models. While creating the delivery models, you can select only those models or Web services that are needed for specific activities.

You can create several delivery models on an XForm, and use them on various human interaction activities associated to the same XForm, as per your requirements.

For **transactional models**, you can choose to provide only the unique identifier as the input instead of the entire message. The XForm or User Interface fetches the complete message part from the source.

Based on the input message, you can also:

- Create **Inbox Models**, where you can model the attributes that can be displayed as columns in the Inbox. These attributes can be used as conditions when searching for tasks in the AppWorks Platform Inbox. You can model a maximum of 25 attributes of type String, 5 attributes each of type Integer, Float, Date, and Boolean columns in the Inbox.
- Create **Email Models**, where you can model an E-mail message using the attributes and send it to the preferred e-mail addresses of the users.

If the response of selected records have to be passed on from the Task in My Inbox to a process instance, you must ensure that the model used to fetch the records is of the type - Transactional

Before you begin:

- [Create an XForm or an External User Interface](#).

To create a delivery model:

1. Right-click the required XForm or External User Interface from the project tree and select **Create Delivery Model** from the context menu.
Alternatively, open the External User Interface, click  (Application Palette) on the toolbar, and click in the **Children - Untitled External User Interface** pane.
An Untitled Delivery Model - Delivery Model window opens.
2. Click  (Add) to select the Operations.
The Select Operations window opens displaying the Operations that are used in the XForm along with their namespaces. The window comprises Free-Form Controls that are dragged onto the XForm and do not have binding to any models, Web service Operations

and Dummy models used in the XForm.

If the Delivery Model is being created on an External User Interface, the Select Operations window displays only the Input Schema.

3. Select **Operations or controls** and click **Add**.

The selected operations are displayed in the Messages tab.

Based on the operations selected in this tab, you can set the task identifiers that will be used to search the tasks in the Inbox and model the messages that will be sent to the users.

4. In the **Pass Data By** column, select any of the following.

Value	To pass the entire object as the input to the task in the business process. By default, this option is selected.
Key	<p>To pass a key, which is a unique identifier of the business object, as the input to the task in the business process. The input criteria is defined for the selected Operations.</p> <ul style="list-style-type: none"> ■ Based on the selection in this column, the data is populated in the Inbox Model and Email Model tabs. ■ If Pass Data By is selected as Key, only the unique identifier of the business object is available for selection in the remaining tabs. If Pass Data By is selected as Value, the complete business object is available for selection. ■ For Dummy models and Free-form controls, only the Value option is available.

5. To model the task content sent to the AppWorks Platform Inbox, select **Enable Inbox Model** at the bottom of the screen.

For the complete procedure, see [creating an inbox model](#).

The tasks that are sent to My Inbox are modeled in the format defined.

6. To model the task content sent to the preferred e-mail ID of the user, select **Enable Email Model** at the bottom of the screen.

For the complete procedure, see [creating an e-mail model](#).

The task content is modeled as defined and is sent to the e-mail ID of the users.

If response of the selected records have to be passed on from the Task in the My Inbox to the process instance, ensure that the model used to fetch the records is of the type Transactional. The non-transactional models must not be used in the User Interface when it is needed to pass on the selected record details to the process instance.

The Delivery Model is created.

After you complete:

- Right-click the delivery model in the project tree and select **Publish to Organization**. The delivery model is now available and can be selected from the property pane of the selected activity in a process.

When a Delivery Model is published, the associated Inbox Model and E-mail Model are also published. While publishing or deploying the Delivery Model, the notification service re-indexes the task data for all the tasks associated with the Inbox model. This action ensures that all the task data is indexed properly as per the new Inbox Model. This action is done in the background. My Inbox will not show the data in the business attribute columns until this re-indexing is completed.

Inbox model datatypes

The Inbox Model supported data types include Integer, Float, Date, Boolean and String. All the data types that are defined through XForm freeform controls or XMLSchema are mapped to the one of the five data types.

The following table shows schema element types available in XMLSchema Editor (CWS Document) and its supported datatype in Inbox Model.

Inbox Model Datatype	XML Schema element type
Integer	<ul style="list-style-type: none">■ byte■ short■ int■ integer■ long■ negativeInteger■ positiveInteger■ nonNegativeInteger■ nonPositiveInteger■ unsignedByte■ unsignedLong■ unsignedShort■ unsignedInt
Float	<ul style="list-style-type: none">■ decimal■ double■ float
Date	<ul style="list-style-type: none">■ date■ time■ dateTime
Boolean	<ul style="list-style-type: none">■ boolean

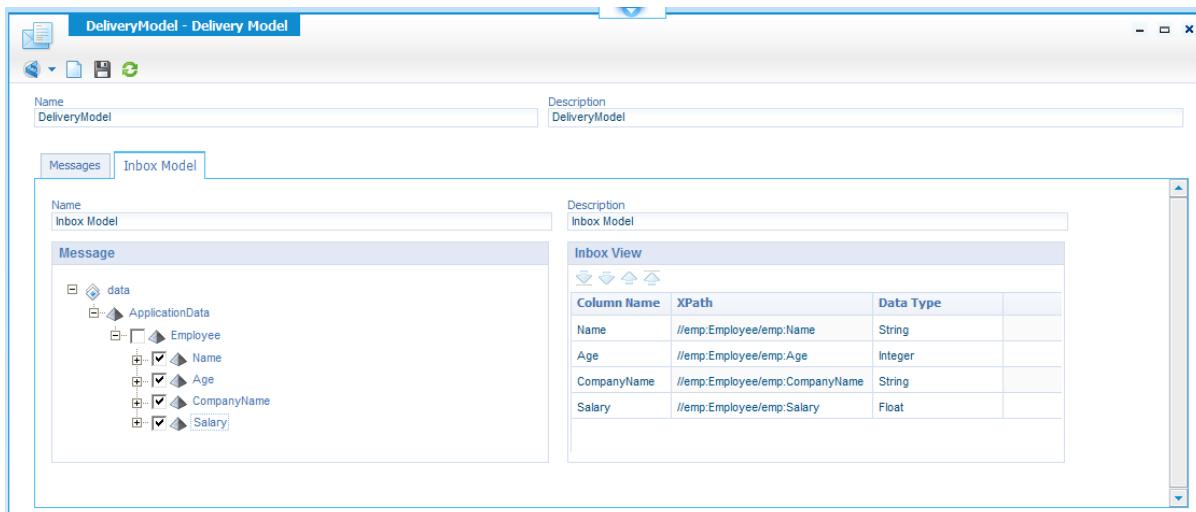
Inbox Model Datatype	XML Schema element type
String	<ul style="list-style-type: none"> ■ anyURI ■ base64Binary ■ duration ■ ENTITIES ■ ENTITY ■ gDay ■ gMonth ■ gMonthDay ■ gYear ■ gYearMonth ■ hexBinary ■ ID ■ IDREF ■ IDREFS ■ language ■ Name ■ NCName ■ NMOKEN ■ NMOKENS ■ normalizedString ■ NOTATION ■ QName ■ string ■ token

The following table shows XForm controls datatype and its supported datatype in Inbox Model.

Inbox Model Datatype	XForm Controls datatype
Date	<ul style="list-style-type: none"> ■ Date
Float	<ul style="list-style-type: none"> ■ Float ■ Double ■ Decimal

Inbox Model Datatype	XForm Controls datatype
	<ul style="list-style-type: none"> ■ Amount
Integer	<ul style="list-style-type: none"> ■ Integer
String	<ul style="list-style-type: none"> ■ String ■ HexBinray ■ base64Binray ■ AnyURI

The following screenshot shows the Delivery Model containing Inbox Model. The tree shows the Employee schema and its elements types are defined as Name - string, Age - int, CompanyName - string, Salary - double. When we select the schema elements to make Inbox Model, the schema element types string, int, and double are mapped as in the previous tables.



Working with dispatch algorithm

While designing a business process, you model the tasks to be sent to targets such as work lists, teams, roles, or users. However, at the runtime due to various factors, such as workload balancing, changes in priorities, expertise and so on, you may want to re-route these tasks to other targets. AppWorks Platform enables you to dynamically re-route the task to other targets, as per the requirement, with the help of intelligent work dispatching algorithms.

You must do the following to create the Dispatch Algorithm and use it:

- Implement the Custom Dispatch Logic in a Java class
- Create the Custom Dispatch Algorithm document and provide the details of the Java class

- Configure the Business Process Model to use this dispatch algorithm
- Publish the dispatch algorithm contents to make them available at the runtime
- Register the dispatch algorithm with the Notification Service Container

To create a dispatch algorithm:

1. Specify the required logic in a Java class that implements the `com.cordys.notification.customdispatch.CustomTaskDispatcher` interface provided by AppWorks Platform, in the `notification.jar` found at `<AppWorks Platform_installdir>/components/notification:`

```
public interface CustomTaskDispatcher {
    /**
     * Implement the custom dispatch logic in this method.
     * @param taskInformation An instance of <code>TaskInformation</code>
     * @return returns a list of <code>IAssignment</code> instances.
     * The notification would be dispatched to all the assignments in this
     list.*/
    public Collection<IAssignment> getAssignments(TaskInformation taskInformation);
    /* implement your logic to decide on task dispatch */
}
```

`TaskInformation` holds the task details. See `CustomTaskDispatcher` and `TaskInformation` in the Java APIs documentation for more information on implementing the Custom Dispatch Algorithm and the usage of `TaskInformation` APIs.

`TaskInformation` also contains `EntityInstanceId` when the task is delivered through a lifecycle or any of its sub-processes. You can use `EntityInstanceId` to obtain the required entity information in your business logic using a Web service on the entity.

2. Create a Java archive of the Custom Dispatch Algorithm.
See [Creating a Java Archive Definition](#) for the procedure to create a JAR file.
3. To create the Dispatch Algorithm document that contains the reference to the Java class, do the following:
 - a. Select a starting point and click  (Dispatch Algorithm modeler).
The Dispatch Algorithm properties page appears.
 - b. Enter the name and description of the Dispatch Algorithm, to identify the Custom Dispatch Algorithm, in the Name and Description text boxes respectively.
 - c. Type the complete path for the class that implements the Custom Dispatch Algorithm in the Fully Qualified Class Name field.
 - d. Click **Save** on the toolbar.
The Custom Dispatch Algorithm is saved and displayed in the Workspace Documents (Explorer) under the specified project.
4. To use this dispatch algorithm in a business process:
 - a. From the [Human Task activity property sheet](#), select the created Dispatch Algorithm from the list.

5. To make the dispatch algorithm contents available at runtime, right-click the project and select **Publish to Organization**.
6. Add `CustomTaskDispatcher.jar` to < AppWorks Platform installation directory >.
7. To Register the Custom Dispatch Algorithm, add the JAR file to the classpath of the Notification Service Container.
See the [JRE section](#) of the Service Container Configuration Interface for more information.
The Dispatch Algorithm is registered with the Notification Service.

Note

- The Dispatch Algorithm is created and configured to be used at runtime. When the business process is executed, the tasks are routed as per the logic provided in the dispatch algorithm. However, if the conditions provided in the dispatch algorithm are not met, then the default work assignment specified for the human task is applied.
- The Dispatch Algorithm is used to resolve the target only while creating a task and the task is sent to the resolved target.

Example

The following example code demonstrates the implementation of a Java Class using `CustomTaskDispatcher` and how to specify the custom dispatch logic in the method `getAssignments`.

Prerequisite: `notification.jar` must be in the CLASSPATH to compile the given sample java code.

```
package com.cordys.notification.customdispatch;
Â
import java.util.ArrayList;
import java.util.Collection;
import com.cordys.notification.customdispatch.CustomTaskDispatcher;
import com.cordys.notification.customdispatch.TaskInformation;
import com.cordys.notification.internal.task.Assignment;
import com.cordys.notification.task.AssignmentType;
import com.cordys.notification.task.IAssignment;
Â
public class MyCustomTaskDispatcher implements CustomTaskDispatcher
{
    public Collection<IAssignment> getAssignments(TaskInformation taskInformation)
    {
        // Write your logic here to decide on the target.
        String userDN = "cn=testuser,cn=organizational
users,o=system,cn=cordys,cn=w212,o=vanenburg.com";
        String RoleDN = "cn=testrole,cn=organizational
roles,o=system,cn=cordys,cn=w212,o=vanenburg.com";
        String TeamID = "001CC438-8F47-11E2-EB11-FC2D603E9626";
        String WorklistID = "001CC438-8F47-11E2-EB12-013A4A739626";
```

```

Â
    // Create Assignment object based on the target. Targets could be either a
User, Role, Team, or a Work list.
    // If the target is a role or a user, provide the corresponding DN as the
parameter.
    // If the target is a Team or a Work list then provide the ID of the team or
work list as the parameter.
Â
    IAssignment assignmentUser = new Assignment(userDN, AssignmentType.user);
    IAssignment assignmentRole = new Assignment(RoleDN, AssignmentType.role);
    IAssignment assignmentTeam = new Assignment(TeamID, AssignmentType.team);
    IAssignment assignmentWorklist = new Assignment(WorklistID,
AssignmentType.worklist);
Â
    ArrayList<IAssignment> assignments = new ArrayList<IAssignment>();
    // Either of these the assignment objects or any combination of these
assignment objects can be added to the returned assignments collection.
    assignments.add(assignmentUser);
    assignments.add(assignmentRole);
    assignments.add(assignmentWorklist);
Â
    /** UseCase # If any of the current target types is 'team', then only add
the 'team Assignment' to the Assignments **/
    Collection<IAssignment> CurrentTargets = taskInformation.getAssignments();
    Iterator<IAssignment> oIterator = CurrentTargets.iterator();
    while(oIterator.hasNext())
    {
        IAssignment assignment1 = oIterator.next();
        if(assignment1.getType() == AssignmentType.team)
        {
            assignments.add(assignmentTeam);
        }
        break;
    }
    return assignments;
}
}

```

Creating a Java archive definition

Before you begin:

- You must have uploaded Java files into the project.

Java Archive Definition helps you to create a compiled Java Archive (.JAR) from the Java files that are available in the project, without using any Java development tool. As Java sources or classes are not in a format compatible with AppWorks Platform, they cannot be packaged with the project in their native format. This method helps you to bundle the Java logic in the project in a format compatible with AppWorks Platform that can be deployed within an application and used at runtime.

1. Select a starting point and click  to open the Java Archive Definition editor.
2. Enter the required details on the interface.
3. Click .
The Save Document dialog box opens.
4. Provide the Name and Description, and click  to browse and select the location to save the Java Archive Definition. The folder path where the Java Archive Definition will be created appears in the Save in Folder field.
5. Click **Save**.
The Save Document dialog box closes.
6. Close the Java Archive Definition window.

The Java Archive Definition is created with the provided name and is stored at the selected location. When you package the Java Archive Definition, it stores a **.JAR** file in the **.cap** file.

After you complete this task:

- **Publish** the Java Archive Definition to an organization.

When deploying an application that contains a Java Archive Definition (or when publishing such an artifact), the generated Java Archive will be deployed at the `<AppWorks Platform_installdir>`. The exact path at which the Java Archive will be deployed is based on the qualified name of the Java Archive Definition, considering the `<AppWorks Platform_installdir>` as its root.

Working with Inbox

Every AppWorks Platform user is assigned an Inbox, which is used to send and receive tasks and notifications. The Inbox is configured for a given user profile and functions like a basic mailbox. As a AppWorks Platform user, you can access tasks that are sent to your work lists, to the roles you are associated with, to the teams you are part of, or to your personal task folder.

To open My Inbox:

1. Open the AppWorks Administration page > **OpenTextEntityIdentityComponents** workspace > **Identity User of OpenText Entity Identity Components** role.
2. Enable List Security for all the lists having a suffix *Internal* for the role. See the *AppWorks Platform Low-Code Design Guide* for more details on enabling the list level security.

Tasks

A task is a message sent from an action to a human participant in the process that a certain activity has occurred or an instruction telling the person to perform the next step in the process. Users can respond to the task, add a new action to the flow after the task, or do both. For example, when the stock of a particular item in a warehouse reaches below a certain level, a re-order task is sent to the warehouse manager. On receiving the task, the

warehouse manager opens it, fills in a purchase order form, and forwards it to the purchase manager.

Notifications

Notifications are used to convey event-specific information to designated recipients or roles. For example, a notification can be used to send a message to the warehouse manager that the purchase order is approved.

Inbox activities

Through AppWorks Platform Inbox, users can perform the following actions on the tasks in the Inbox:

- Claim the tasks assigned to their work lists
- Start executing their tasks
- Temporarily stop working on a task
- Delegate a task to other users
- Completely stop working on a task
- Complete a task
- Skip a task
- Modify the start date of a task
- Forward a task to other users
- Pause and Resume a task

For more information on the operations that can be performed on the tasks in the Inbox, see [Working with tasks](#).

In addition to the listed operations, every user can also perform the following:

- [Search for specific tasks in the work list](#)
- [Search for notifications in the work list](#)
- [Attach memos to a task](#)
- [Attach files to a task](#)
- [Tag a task](#)
- [Set Reminders on tasks](#)
- [Set Out of Office Information](#)
- [Set preferences on the Inbox](#)
- [Personalize the Inbox View](#)

Every work list manager, team lead, or a user with a specific role, can do the following:

- Work on the tasks for which they are the users
- Assign a task to a user
- Revoke a task from a user
- Suspend a task

- [Forward a task to another work list or team](#)
- [View the status of various tasks](#) belonging to the teams in the work list

For more information on a Work list Manager or Team Lead operations, see [Working with tasks](#).

Notes

- Users can choose to show or hide certain tabs to be displayed in My Inbox. When the unwanted tabs are not selected, their associated functionality will not be loaded, thus reducing the application loading time.
 - Users can choose to view or hide only the tabs that are viewable to them. They can see and toggle the visibility of the tabs by clicking  (Extend) on the titlebar. All the tabs displayed on the interface are shown in the list and are marked as selected.
 - My Inbox application persists the user settings. When the Inbox is closed, it remembers the settings and when reopened, displays the Inbox based on the last modified settings.
 - Click  (Extend) on the titlebar. Users can toggle the visibility of each tab by selecting and clearing the respective tab from the list.
 - BPM Task/Other Task - This view contains the following tabs: Attachments, Tags, Memos, Status Info, and Activity Form
 - Case Activity - This view contains the following tabs: Related Activities, Followups, Attachments, Events, Tags, Memos, Status Info and Overview Tab containing Case and Activity Overview.
 - Case Instance - This view contains the following tabs: Related Activities, Followups, Attachments, Events and Overview.
 - Users can set each tab to be displayed in one of the five regions - top, right, bottom, left and middle.
 - My Inbox persists and uses the view of the respective task or activity as set by the user. Inbox persists the visibility and region of the visible tabs. Persisting happens on close of the respective task or activity view.
- Users can [validate the actions performed on tasks or case activities](#) in the Inbox, before or after they are done.
- Users can use Inbox as a composite control in other XForms. See [Using Inbox as a Composite Control](#), for information on the procedure to use Inbox as a composite control.
- Users can configure the language, locale, and time zone settings on User Preferences to enable Inbox to be displayed with the corresponding language with date and date-time as set in locale and time zone.
- If a task is sent to a user with multiple roles, the All Tasks view in the Inbox will display the task only once instead of displaying the same task multiple times; this is done to

reduce redundancy thus enhancing the Inbox performance. Lifecycle tasks are also displayed.

- If tasks are sent to a target, that is, Role or Team or Worklist, and if the tasks are not claimed by any user, the count of all the unclaimed tasks will be displayed after the target label within parenthesis (). The unclaimed tasks count does not depend on any filter criteria related to viewing of the Inbox tasks. If multiple users act on the Inbox at same time and one of them claims the tasks, the count will be reflected based on the Inbox View Mode.

Attaching files to a task

Attaching a file to a task is an easy way to send important documents or images that may contain additional information about the task or instructions on performing certain actions on it. When the files are attached, the subsequent activities in the business process also contain these attachments and the users who have the necessary permissions can act upon those attachments. You can attach files to tasks that are displayed in My Inbox, through the Inbox toolbar, context menu, task toolbar, or from the attachments pane.

You can customize the tabs displayed in the Inbox, tasks, or activities, to enhance or minimize the number of items displayed. This reduces the clutter as well as increase the productivity.

To customize the tabs:

1. If you cannot see the Attachments tab, click  on the title bar.
2. All the tabs, that are displayed on the interface, are shown in the list as selected. The tabs that are not displayed on the interface are shown as unselected.
3. Select the **Attachments** option from the list.
4. To close any of these tabs, clear the check box of the tab you want to close.

To attach a file to a task:

1. On the Welcome page, double-click  My Inbox.
My Inbox window opens, displaying the personal tasks, by default.
2. To add a file to a task, select the required task in the Inbox and do **any** of the following:
 - Click  on the Inbox task toolbar.
 - Click the **Attachments** tab.
All the files that are attached to the process are displayed in the Attachments pane.
 - Click  on the toolbar.
An Add Attachments dialog box opens.
 - Based on the access permissions set on the attachment while modeling the business process, the user can view or update the files that are displayed. For instance, if the attachment has only Read permission, the user can only view the existing files but cannot attach any new file to the activity. If the attachment has

both Read and Write permissions, the user can attach a new file as well as modify the existing file and reload it. If the attachment has Read, Write, and Delete permissions, the user can delete the file as well.

3. Select **Attachment Type** from the list.
This Attachment Type is defined while modeling the case.
4. Click  next to the Document field and select the type of document to attach to this activity.
5. Provide a meaningful **Description** to the attachment.
6. Click **OK** to close the dialog box.
The attachments are displayed in the Attachments section.

The attachment is added to the task.

To view the attachment:

- Double-click the task in the Inbox.
The attachments related to the task are displayed in the Attachments pane.
- Alternatively, you can view a file attached to the task in the preview section at the bottom of the Inbox, when you click a task in the Inbox. Preview mode should be set by the work list manager while customizing the work list view.

To delete an attachment:

When not required, attachments of a task can be deleted.

- Select the required attachment and click  displayed on the toolbar of the Attachments pane.

Creating an Inbox model

Every task from a business process is sent to the Inbox of the relevant user, role, team, or work list, which will eventually be picked up by the concerned users. The attributes that must be displayed as columns in the Inbox can be defined, by the Worklist manager, through the Inbox modeler.

For example, assume there is a task of registering an Order, a worklist manager can choose to display the fields, such as, OrderID, Order Quantity, and Discount as columns in the Inbox, so that users can sort their messages based on these columns.

This task is part of creating an inbox model on the messages in the Delivery Model. Using the Inbox model tab, you can personalize and model the tasks that are sent to a user's AppWorks Platform Inbox.

To create an inbox model:

1. Select **Enable Inbox Model** at the bottom of the screen of the [Delivery Model](#).
An Inbox Model tab is added to the existing tabs.

2. Click the **Inbox Model** tab.

An Inbox modeler appears, displaying the XML structure defined for the Task in a tree structure on the model tab by default.

3. Type **Name** and a meaningful **Description** for the Inbox model.

4. Expand the **Application Data** and select the attributes, that you want to display as columns in the My Inbox.

The selected attributes and their XPaths are displayed in the Inbox View section on the right pane, in the respective columns. The data type associated to the selected attribute is shown in the Data Type column.

- The selected attributes are displayed in the Inbox only when the task is delivered to any of the following:

- work list
- team
- role
- user

- Furthermore, the attributes to be displayed in the My Inbox can be customized through the Inbox customization interface.

- You can modify the names of the attributes in the Column Name field, which are used as information headers in the My Inbox.
- For the Inbox columns for which the data type value is not available, the data type value is set to Read from Source. While publishing such columns, the data type defined in the Source (Schema, Free Form Controls, and so on) will be associated with them. The data type will not be available only when the content is migrated from earlier versions of AppWorks Platform.
- While changing the data type of the Inbox column, ensure that the selected data type is same as the source or it belongs to the supported data types shown below. If the selected datatype is not one of the supported conversions, you may not be able to view the data in Inbox grid of the corresponding Inbox Column.

Supported data type conversion

	Boolean	Integer	Float	Date	String
Boolean	✓	✓	✓	✗	✓
Integer	✗	✓	✓	✗	✓
Float	✗	✗	✓	✗	✓
Date	✗	✗	✗	✓	✓
String	✗	✗	✗	✗	✓

- If you do not select any attribute and try to publish the model, the following error occurs: "Inbox Model does not have any attributes defined to enable search in My Inbox. Define the required attributes and retry the operation." If you want to continue using Inbox Model, you must select at least one attribute and republish the model.

On opening the Inbox Model tab, if some of the Inbox columns are marked with !, it indicates that these fields are no longer available in the Source (Schema, Free Form Controls). Click to delete such columns and save the Delivery Model. Otherwise, validation of your project content will fail.

- After synchronizing your projects or upgrading them in CWS, you must publish the projects in sequence. Do not try to publish your projects in parallel when doing it for the first time. However, subsequent publishes can be done in parallel
- In any AppWorks Platform instance, the number of unique inbox model columns (task identifiers) allowed across all projects and organizations are as follows:
 - A maximum of 25 String data types.
 - A maximum of 5 each of type Integer, Float, Boolean and Date.

Note: If the number of inbox model columns exceed the above limits, then publishing or deploying the inbox model columns fails.

Customizing the Inbox view

Before you begin:

- You need to be a Work List Manager, a Team Lead, or a user with any other Role, to be able to view this tab.

As part of human interaction in a business process or a case, tasks are sent to the task folders in the Inbox, such as Work lists, Teams, Roles, or to the users themselves. The corresponding managers or leads can customize the way these tasks are displayed in the Inbox.

For example, work list managers can view all the work lists they manage. Team leads can view all the teams they lead. Users of particular roles can view the roles they belong. Users who have all the three functional roles can view all the task folders. Users with a combination of the above mentioned functional roles can view the corresponding task folders and can do the following.

Functional Role	Action
Work List Manager	Can customize the way tasks are displayed in a work list folder. All the members in the work list receive the tasks in their Inbox based on these settings.
Team Lead	Can customize the way tasks are displayed in a team folder. All the team members in the receive the tasks in their Inbox based on

Functional Role	Action
	these settings.
<Role>	Can customize the way tasks are displayed in a role folder. All the members of that particular role receive the tasks in their Inbox based on these settings.

They can set:

- The type of tasks that must be visible in the Inbox, such as, displaying completed tasks, non-workable tasks.
- The visibility of preview pane for tasks.
- The order or position in which the tasks are displayed.
- The attributes that can be displayed as columns in the Inbox.
- The sort order for tasks.

To customize the Inbox view:

1. On the Welcome page > My Applications App Palette, click  (My Inbox). The My Inbox window opens, displaying the personal tasks, by default.
2. Click the **Customization** tab on the left side of the Inbox. Based on your role, the folders are displayed on the right side.
3. Double-click the folder you want to customize. A pane containing the set of properties that can be customized is displayed on the right side.
4. Customize the view by changing the required settings.
 - The **System Attributes**, such as Activity, Process, Due On, and so on, cannot be modified.
 - The **Task Attributes** belonging to a task, such as OrderId, Quantity, and so on, can be modified according to the requirement. The column in the Inbox will be displayed per the modified name.
 - Ensure that the task identifiers defined while creating the delivery model, match the modified column names. Otherwise, the search and sort functionality of the inbox will not work.
5. Click  on the toolbar.

The tasks in the Inbox are displayed according to the customization.

Personalizing the Inbox view

Tasks are either assigned to the users or are claimed by users themselves from their worklists, roles, and teams.

The tasks are received in their Personal Tasks folder and the All Tasks folder in the Inbox. Usually, the tasks sent to the Inbox have several attributes which are displayed as Columns in the Inbox. The columns that are displayed are determined by the attributes set while defining the Inbox models.

The tasks received by the users can be from several Inbox models. Hence, the number of attributes for tasks in the Inbox might vary.

AppWorks Platform provides a facility for the users to personalize the view of the tasks received in their Personal Tasks and All Tasks folders. They can choose the columns to be displayed in their Inbox. For instance, users can choose to display business attributes like Quantity, Shipping date, as columns in their Inbox so that they can sort based on these attributes. By default, the tasks in the Inbox are sorted based on the Received Date of tasks. Users can also override the settings set by the work list manager, thus personalizing their Inbox according to their ease of use.

To personalize the Inbox view:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Right-click any column header in the Inbox and move the pointer over the **Column Chooser** field.
All the default attributes that are selected while modeling the Inbox model are displayed. The attributes that are shown as columns are selected and the attributes that are not shown are cleared. You can select or clear the attributes by clicking them.
3. To select more attributes, do the following:
 - a. Right-click any column header in the Inbox and select **Column Chooser > More Attributes**.
A Choose Columns page appears, displaying system attributes as well as business attributes, in the Existing Attributes column. All the delivery models are displayed in the From list. By default, the system attributes are displayed in the From list.
 - b. From the **From** list, choose the required delivery model.
All the business attributes pertaining to the delivery model are displayed in the box below.
 - c. Select the required attributes and click The selected attributes are displayed in the Selected Attributes column.

While you cannot modify the **System Attributes** such as Activity, Process, Due On, and so on, the business attributes can be renamed as per the requirement. These are displayed as columns in the user's Inbox.

The Inbox is personalized.

To remove the columns:

- Select the required attribute in the Selected Attributes column and click The business attributes that are displayed in the Inbox are derived from several Inbox models and need not always contain values in them. While personalizing, you may not have selected all the attributes that are provided in the Inbox model.

To show all the default attributes:

- To show the attributes selected in the Inbox model, you can click . This restores all the attributes selected in the Inbox model.

Setting Inbox preferences

You can set the following preferences with regard to the display of tasks in your Inbox, such as the number of tasks to be displayed in a page, hide or display work lists that do not contain any tasks and so on.

To set Inbox preferences:

1. On the Welcome page, double-click  (My Inbox). The My Inbox window opens, displaying the personal tasks, by default.
2. Click in the shortcut bar. A Preferences dialog box opens.
3. In the **Worklists** group box, do the following:
 - To manage the **Empty Work Lists** in the Inbox, select any of the following options.

Apply System Preferences	Apply the settings that are set at the system level, while configuring the Notification Service Container.
Hide	Does not display the work lists that do not have any tasks in them.
Show	Displays the work lists that do not contain any tasks.

The option selected here will override any default settings.

- To manage **Notifications** in the Inbox, select any of the following options.

Show	Displays the Notifications tab in the Inbox.
Hide	Does not display the Notifications tab in the Inbox.
Apply System Preferences	Applies the settings that are set at the system level, while configuring the Notification Service Container.

The option selected here will override any default settings.

- To manage **Memo as Tooltip** feature in the Inbox, select any of the following options.

Apply System Preferences	Applies the settings defined at the system level while configuring the Notification Service Container.
Hide	Disables the feature of showing the Memo as a tooltip when the

	mouse is hovered on an Activity column in the Inbox.
Show	Enables the feature of showing the memo associated to a task as tooltip. Select the required task in the Inbox, and hover the mouse on the Activity column to view the memo. Tooltip contains the following: 'Name of the user who added the memo', 'Time when the memo is added', and the 'Memo content'. By default, the tooltip displays the first memo associated to the task. You can associate multiple memos to a task and the memo with the lesser timestamp value amongst the memos will be displayed as the tooltip. In Refresh mode, the tooltip information that is associated to the selected task is cached. The cache is updated only when the Inbox is refreshed.

The option selected here will override any default settings.

4. In the **Task Paginations** group box, click the **No. of Tasks** per Page list and select the number of records to display in a single page of the Inbox.
5. In the **Notification** group box, do the following to enable the preference:
 - Select the **Send task or notification to this e-mail** check box to enable the tasks or notifications to be sent to the e-mail id mentioned in the E-mail field.
 - The read-only E-mail field displays the E-mail Id, which is specified in the Welcome Page Preferences. This is the E-mail Id to which you want the tasks or notifications to be received.
 - Ensure that the E-mail connector is configured in the Notification Service Container to send tasks to the users e-mail IDs.
 - The tasks or notifications from a process or cases are, by default, sent to the AppWorks Platform Inbox of the user. However, By enabling the option Send task or notification to this e-mail the tasks or notifications can also be delivered to the user's preferred e-mail address.
6. Click **OK**.

The preferences for the Inbox are saved and all the tasks and notifications are displayed as specified.

Searching notifications

This topic describes the procedure to filter notifications based on specified criteria in a user's Inbox. Search filters are useful when you routinely want to perform certain actions on messages, as per the criteria specified.

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.

2. On the left side of the Inbox, click the **Search** tab.
3. Select **Notifications** from the **Within Work List**.
4. You can search the notifications on any or a combination of the following:
 - Click  next to Source and select the required source on which you want to search.
 - Click  next to Subject and select the required source on which you want to search.
 - Click  next to the Delivered field and select the date on which the required notification was delivered to the Inbox.
5. Click **Search Now**.
The notifications matching the specified criteria are displayed on the right pane.
6. To clear all the values of search criteria, click **Clear**.

The notifications in the Inbox are filtered based on the specified criteria.

To save the search for future references:

1. On the top of the Search Criteria pane, click .
The Save Search dialog box opens.
2. Type a name for the search.
The search is saved in the list.

To view your previous saved searches:

1. In the **Select** list, select the search name.
The specified criteria is displayed in the corresponding fields below.
2. At the bottom, click **Search Now**.
The notifications that match the search criteria are displayed on the right pane.
Alternatively, you can select the previous searches from the View list on the top right corner of the Inbox.

To delete the saved searches from the list:

- Click .

Searching tasks in the Inbox

This topic describes the procedure to search tasks based on specified criteria in a user's Inbox, using:

- **Basic Search** - To search the Inbox for specific tasks based on the system specific attributes, like the name of the process that activated the task, the name of the activity, assignee, and so on.
- **Advanced Search** - To search on the task data such as the attributes or task identifiers specified while creating the delivery model and setting the task identifiers.

To search tasks in the Inbox:

1. On the Welcome page, double-click  (My Inbox).
My Inbox window opens, displaying the personal tasks, by default.
2. On the left side of the Inbox, click the **Search** tab.
A search interface is displayed.
3. From the **Within** list, select the folder you want to search.
Users can search for the tasks that are received in the following folders of their Inbox.

Personal Tasks	By selecting this folder, the tasks that are either claimed by the user or assigned to the user can be searched upon.
<work lists>	By selecting this folder, the tasks that are sent to the work lists the user belongs can be searched upon.
<roles>	By selecting this folder, the tasks that are sent to the roles the user belongs can be searched upon.
<teams>	By selecting this folder, the tasks that are sent to the teams the user is part of can be searched upon.
Cases	<p>By selecting this folder, the activities belonging to a <case model> assigned to the user can be searched.</p> <ol style="list-style-type: none">1. Select Cases from the Within list. A From Case list box appears below.2. Select the required <case> from the list. Based on the case selected, the search interface is populated. <p>For more information on the procedure to search cases, see Searching Cases in the Inbox.</p>

4. Do one of the following:
 - To search the Inbox based on the system specific attributes such as status of the case, the due date and the start date, fill the search details in the search criteria section and click **Search Now**. For more information on the fields of the basic search interface, refer to [Basic Search Interface of the Inbox](#).
 - To search the Inbox based on task identifiers, fill the search details in the search criteria section, click **Add** to add the search criteria to the criteria list and click **Search Now**. For more information on the fields of the advanced search interface, refer to [Advanced Search Interface of the Inbox](#).

To save the search for future reference:

1. On top of the Search Criteria pane, click .
The Save Search As dialog box opens.
2. Type a name for the search.

The search criteria are saved in the list.

A work list manager can save a search criteria which can be accessed by all the members of the worklist, by selecting Public in the Save Search As dialog box. If the work list manager selects Private in the Save Search As dialog box, the search criteria can only be accessed by the work list manager or the user who created it.

To view your saved searches, do the following:

1. Select the search name from the **Select** list.
The specified criteria is displayed in the corresponding fields below.
2. Click **Search Now** at the bottom to display the tasks matching the specified criteria on the right pane.
3. Alternatively, you can also select the previous searches from the **View** list on the top left corner of the Inbox.
Based on the selection, the filter is expanded to narrow your search.

To view the tasks that were completed during a certain period:

1. Select **Completed Tasks** from the list.
2. In the subsequent list boxes that appear, select **Between** and the range of **Dates**. All the tasks completed during that period are displayed in the Inbox. The Inbox does not display tasks unless they are due to be started.
3. To view the tasks in the Inbox that have a start date in the future, select **Calendared Tasks** from the **View** list.

Search interface of Inbox

This topic describes the various fields of the search interface of an Inbox.

Basic search interface

The following table contains the various fields of basic search interface, their description, and the actions that can be performed on those fields.

Field	Description	Action
Activity	Searches the Inbox based on the Activity in the Process that invoked this task.	Click  next to Activity and select the subject based on which you want to search the Inbox. Alternatively, you can type the subject in the text box. If you are searching the Inbox using multiple subjects, they should be separated by a semi-colon.
Assignee	Searches the Inbox based on the name of the Assignee who was assigned the task.	Click  next to Assignee and select the name of the user you want to search the Inbox on.
Due On	Searches the Inbox based on the date on which the tasks	Click  next to Due On and select the required date.

Field	Description	Action
	are due to be completed.	
Process	Searches the Inbox based on the Process that invoked this task.	Click  next to Process and select the subject based on which you want to search the Inbox. Alternatively, you can type the name of the process in the text box. If you are searching the Inbox using multiple subjects, they should be separated by a semi-colon.
Sender	Searches the Inbox based on the Sender's name.	Click  next to Sender and select the name of the sender you want to search the Inbox on.
Started On	Searches the Inbox based on the date on which the tasks are started.	Click  next to Started On and select the required date.
Priority	Searches the Inbox based on the priority levels.	Select the required priority levels from the list.
State	Searches the Inbox based on the state of the tasks at that point of time.	Select the required state from the list.

Advanced search interface

The following table contains the various fields of advanced search interface, their description, and the actions that can be performed on those fields:

Field	Description	Action
Process	Searches the Inbox based on the Process that invoked this task.	Click  next to Process and select the subject based on which you want to search the Inbox. Alternatively, you can type the name of the process in the text box. If you are searching the Inbox using multiple subjects, they should be separated by a semi-colon.
Activity	Searches the Inbox based on the Activity in the Process that invoked this task.	Click  next to Activity and select the subject based on which you want to search the Inbox. Alternatively, you can type the subject in the text box. If you are searching the Inbox using multiple subjects, they should be separated by a semi-colon.

Field	Description	Action
Tasks	Searches the Inbox based on the tasks selected.	Select the required task from the list box. The list is automatically populated based on the work list selected.
Attribute	Searches the Inbox based on the attributes that are set while creating a delivery model.	Select the required attribute from the list box. The list automatically populates based on the selected task.
Operator	Operators for creating a condition	Select the required operator to build the condition. The conditions in the list are based on the attribute selected.
Value	Value for the identifier	Type the required value for the identifier.
Sort Order	Contains the order in which the search results are sorted.	Select the required attribute from the list box, click Sort Order , and define the sort order. When the search is saved, the sort order is automatically saved and the results are displayed in the defined order henceforth.

Searching cases in the Inbox

You can search the Inbox for cases based on system specific attributes or on business attributes using:

- **Basic Search** - To search the Inbox for specific case instances based on the system specific attributes, such as the status of case instances or the start and due dates of the case instances.
- **Advanced Search** - To search the Inbox for the case instances based on the attributes or case identifiers specified while modeling the case and setting the case properties.

You cannot search the Inbox for lifecycle instances.

To search cases in the Inbox:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying personal tasks, by default.
2. On the left side of the Inbox, click the **Search** tab.
3. Select **Cases** from the **Within** list.
The For Case list populated with the existing case models, is displayed. This list of case models is initialized when the user opens My Inbox. If other case models which are published new are needed for search, close and open My Inbox.
4. Select the case on which you want to define the search criteria from the **For Case** list.
The descriptions that are provided for the case models are what is shown here instead of

the names. This behavior helps users to have a better view of what case models they are selecting and their purpose.

5. Do one of the following:

- To search the Inbox based on the system specific attributes, fill the search details in the search criteria section and click **Search Now**.

For more information on the fields of the basic search interface, see [Basic Search Interface of Cases in the Inbox](#).

- To search the Inbox based on case identifiers that are [set while modeling the case](#), fill the search details in the search criteria section, click **Add to the List** to add the search criteria to the criteria list and click **Search Now**.

For more information on the fields of the advanced search interface, see [Advanced Search Interface of Cases in the Inbox](#).

To save the search for future reference:

1. On the top of the Search Criteria pane, click .

The Save Search As dialog box opens.

2. Type a name for the search.

A work list manager or a team lead can save a search criteria which can be accessed by all the members of the work list or team respectively, by selecting **Public** in the **Save Search As** dialog box. When **Private** is selected in the **Save Search As** dialog box, the search criteria can be accessed by the work list manager or the User whoever created it.

The case instances in the Inbox are filtered based on the specified criteria. The case instances that match the search criteria are displayed on the right pane of the Inbox.

Click the required instance and all the activities related to that case instances are displayed in the bottom pane.

The search criteria is saved in the list.

To view your saved searches:

1. Select the search name from the **Select** list.

The specified criteria is displayed in the corresponding fields below.

2. To display the cases matching the specified criteria in the right pane, at the bottom, click **Search Now**.

3. Alternatively, you can also select the previous searches from the **View** list on the top right corner of the Inbox.

Search interface for cases in the Inbox

This topic describes the various fields of the case search interface in the Inbox.

Basic search interface

The following table contains the various fields of basic search interface, their description, and the actions that can be performed on those fields.

Field	Description	Action
Due Date	Searches the Inbox based on the date on which the case instances are due to be completed.	You can search for the cases that started on a particular date, before a particular date, after a particular date or between two dates. 1. Select the required Operator from the Due Date list. 2. Click  next to date field and select the required date.
Start Date	Searches the Inbox based on the dates on which the case instances are started.	You can search for the case instances that started on a particular date, before a particular date, after a particular date or between two dates. 1. Select the required Operator from Start Date list. 2. Click  next to date field and select the required date.
Status	Searches the Inbox based on the status of the case.	Select the check boxes against the various status of the cases, such as <ul style="list-style-type: none">■ New■ Completed■ In Progress■ Suspended■ Aborted■ Terminated

Advanced search interface

The following table contains the various fields of advanced search interface, their description, and the actions that can be performed on those fields.

Field	Description	Action
Attribute	Searches the Inbox based on the attributes that are set for the case instances while modeling the case.	Select the required attribute from the list. The list is automatically populated based on the task selected.
Operator	Operators for creating a condition	Select the required operator to build the condition. The conditions in the list are based on the attribute selected.
Value	Value for the identifier	Type the required value for the identifier.

Forwarding notifications

To forward notifications:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the content on the Work List tab, by default.
2. Click **Notifications**.
The notifications sent to this work list are displayed on the right side.
3. Do any of the following to access the Forward Notifications dialog box:
 - Right-click the required notification from the work list and select **Forward**.
 - Select the required notification from the work list and click  on the toolbar.
 - Right-click the required notification from the work list, select Open, and click  on the notification tool bar.

A Forward Notifications dialog box opens.

4. Beside User Name, click .
- The Select User dialog box opens.
5. Select the user to which you want to forward the notification and click **OK**.
6. Click **Send**.

The notification is forwarded to the specified recipients.

Forwarding tasks

Tasks in the Inbox can be forwarded from one user to another or across teams, work lists, and roles. While users can forward tasks to other users within their group, work list managers or team leads can forward tasks across work lists or teams respectively.

Note

- While forwarding a task, the ownership of the task is transferred to the receiver.
- Tasks can be forwarded to a user if they are in the Assigned State only.

To forward tasks:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Right-click the required task and select **Forward**.
A Forward Tasks dialog box opens.
Based on the authorizations to the users, the Forward Tasks dialog box contains various options. See [Forward Tasks Interface](#) for more details.
3. Select the required options and click **Send**.

The task is forwarded to the selected work list, team, role, or user and appears in the corresponding work list, team, role folders, or in the personal tasks list of the user respectively.

Forward task interface

Based on the authorizations to the users, the Forward Tasks dialog box contains various options.

Work List Manager	<ol style="list-style-type: none"> 1. Based on the requirement, do any of the following: <ul style="list-style-type: none"> ■ Click User next to User Name. The Select User dialog box opens. <ul style="list-style-type: none"> ■ Select the required work list and click OK. The <name> of the user is displayed in the User Name field. ■ Click Work List next to Work List Name. The Select Users dialog box opens. <ul style="list-style-type: none"> ■ Select the required work list and click OK. The selected work list is displayed in Work List Name. ■ In Reason for Transfer, provide the reason for forwarding the task. When forwarding a task to another work list, you must provide a reason. The reason is displayed as a memo to the user to whom the task is forwarded. 2. Click  and select the Due date by which this task must be completed. <p>A Work List Manager can also forward tasks across other work lists or to users in other teams.</p>
Team Lead	<ol style="list-style-type: none"> 1. Based on the requirement, do any of the following: <ul style="list-style-type: none"> ■ Click the User option, click next to the User Name field. Select the required work list from the Select User dialog box that appears and click OK. The <name> of the user is displayed in the User Name field. ■ Click the Team option, click next to the Team Name field. Select the required team from the Select Team dialog box that appears and click OK. The selected team is displayed in the Team Name field. Provide the reason for forwarding the task in the Reason for Transfer text box. When forwarding a task to another team, it is mandatory to provide a reason. This will be displayed as a memo to the user to whom the task is forwarded. 2. Click  and select the Due date by which this task must be completed.

	A Team Lead can also forward tasks across other teams in the organization.
Role	<ol style="list-style-type: none">1. Based on the requirement, do any of the following:<ul style="list-style-type: none">■ Click User next to the User Name field. The Select User dialog box opens.■ Select the required user and click OK. The name of the user is displayed in User Name.■ Click Role next to Role Name. The Select Role dialog box opens.<ul style="list-style-type: none">• Select the required role and click OK. The selected role is displayed in the Role Name field. Provide the reason for forwarding the task in the Reason for Transfer text box. When forwarding a task to another role, it is mandatory to provide a reason. This will be displayed as a memo to the user to whom the task is forwarded.
User	<ol style="list-style-type: none">2. Click  and select the Due date by which this task must be completed.1. Click  next to the User Name field. The Select User dialog box opens.2. Select the required user and click OK. The <name> of the user is displayed in the User Name field.

Working with tasks in the work list

Users can work on the tasks available in the personal list. However, they can also claim the tasks that are assigned to the roles, teams, or work lists that they are associated with. If applicable, users can view the tasks, open them, and complete them from the Inbox itself.

User operations

AppWorks Platform Inbox provides a task toolbar to the users through which various operations can be performed on the tasks such as claiming the tasks, starting the claimed or assigned tasks, pausing the tasks when required and so on. Every user can perform some basic operations, such as, claim task, start task, pause task, complete task, and so on. A Work list Manager or a Team Lead can perform additional operations, such as, skipping a task, suspending a task, or forwarding a task to another work list or team and so on. The following sections describe the various operations that can be performed on various tasks or cases in the Inbox, by users as well as managers.

Apart from using the icons on the toolbar, you can also use keyboard shortcuts to perform actions on the tasks. For more information on the keyboard shortcuts list, see [Keyboard Shortcuts](#).

Icons	Operation	Description	Task status
	Claim	Claim the task from the work list. Any user in the work list can claim the task and work on it. Once the task is claimed by the user, it is displayed under the Personal Tasks tab of the user and the corresponding work list.	The task status changes from (New) to the (Assigned) state.
	Start	Start executing the task. Once the task claimed is started, the user can perform the following: <ul style="list-style-type: none"> ■ Click to Pause the task. ■ Click to Stop the task. ■ Click to Complete the task. 	The task status changes from (Start) to (In-Progress) state.
	Pause	Temporarily stop executing the task. Click on the task toolbar to resume a paused task.	The task status changes to Paused state.
	Stop	Completely stop executing the task. Click to start executing a stopped task.	The task status changes to Assigned state so the user can start the task again.
	Revoke	Only a claimed task can be revoked. Once it is revoked, it is moved back to the corresponding folder. For example, a revoked task will be moved back to the work list folder, if it was claimed from the work list or a task claimed from a team is moved back to the team folder when it is revoked. When the users who are delegated or forwarded tasks do not acknowledge to work on those tasks, they will be revoked and assigned back to the user who has forwarded those tasks.	The task status changes to New state.
	Complete	Finish executing the task. Click to complete the task.	The task status changes to Complete state.

Icons	Operation	Description	Task status
	Skip	Skip the task. When the task is skipped, it becomes a non-workable item.	<p>Note: Completed tasks are not displayed in the Inbox, by default. They are visible only when the work list manager selects the Show 'Completed' Tasks in the Work List check box while setting work list preferences on the Inbox.</p> <p>The task changes to Obsolete state.</p> <p>The skipped task is not visible in the work list, by default. It is visible in the work list, only if the work list manager selects the Show Non-Workable tasks in the Work List check box, while setting work list preferences on the Inbox. For the procedure on skipping a task, see Skipping a Task.</p>
	Delegate	Delegate the task to another user.	This is performed on the personal tasks. The ownership of the task lies with the User who delegated the task. For the procedure on delegating a task, see Delegating a Task .
	Forward	Forward the task to another user.	This is performed on the personal tasks. The ownership of the task lies with the User to whom the task is forwarded. For the procedure on forwarding a task, see Forwarding a Task .
	Modify System Attributes	Modify the start date of a task. For Case activities, users can modify Start Date and also the Due Date of the task if they have the required access permissions defined while modeling the case.	For the procedure on modifying the system attributes of a task, see Modifying the System Attributes of a Task .
	View Process instance	Open the read-only graphical view of the process instance.	To open the read-only graphical view of a process instance, users must have:

Icons	Operation	Description	Task status
			<ul style="list-style-type: none"> ■ Read permission on the process instance ■ One of the following roles - Developer, Administrator, or Process Task User internal role

Manager operations

The following table describes the icons on the Task Toolbar and various operations that can be performed by a work list manager or a team lead, apart from the operations.

Icons	Operation	Description	Task status
	Assign	Assign the task to a user. A user with a work list manager role or a team lead role can assign a task to other users in their corresponding work lists or teams respectively. Once a task is assigned, the task is displayed under the Personal Tasks tab of user and the corresponding work list or team folders.	The task status changes from New to the Assigned state.
	Complete	Finish executing the task on behalf of any user, if required. Click to complete the task.	<p>The task status changes to Complete state.</p> <p>The completed task is not visible in the Inbox, by default. It is visible only when the work list manager selects the Show 'Completed' Tasks in the Work List check box while setting work list preferences on the Inbox.</p>
	Suspend	Suspend the task from executing. Only a work list manager or team lead can suspend the execution of a task in the work list or team folder. A suspended task is visible only to a worklist manager or team lead.	<p>The task status changes to Suspend.</p> <p>The suspended task will not be visible in the work list or team folder, by default. It will be visible, only when the work list manager selects the Show Non-Workable tasks in the Work List check box while setting work list preferences on the</p>

Icons	Operation	Description	Task status
	Skip	Skip the task. Once the task is skipped, it will become a non-workable task .	Inbox. To resume the task again, the manager or lead has to click on the Task Toolbar. The status of the task in the will change to Pause state. Users can then click on the Task Toolbar to start executing it again. The task changes to Obsolete state. The skipped task will not be visible in the work list, by default. It will be visible in the work list or team folder, only if the work list manager or team lead selects the Show Non-Workable tasks in the Work List check box, while setting work list preferences on the Inbox. For the procedure on skipping a task, see Skipping a Task .
	Forward	Forward the task to another work list or team.	This is performed on the tasks in the work list or team folder. The ownership of the task lies with the User to whom the task is forwarded. For the procedure on forwarding a task, refer to Forwarding Tasks .

View history of a task

To view the history of the task:

Do any of the following:

- Right-click the required task and select **View History**.
- Select the required task and click on the Inbox tool bar.
- Open the required task and click on the Task tool bar.

An Item History page appears. This page provides a Read-Only view of various states of the selected task.

It contains information on:

- Current state of the task
- Old and new states of that task
- Actions that were performed on the task
- Names of the users who performed any action on the task
- Process that executed this task

Viewing task status in the Inbox

Before you begin:

- Only a Work List Manager, Team Lead, or a user with a specific Role will have the option of viewing the target overview, using the **Overview** tab in My Inbox. A target in this context refers to a Work List, Team, or a Role.

The Overview tab displays statistics based on the status of various tasks that reflect the workloads of a target; it also provides a view of the workload on each user associated to the target. This enables the respective managers to assess the progress of the tasks in their targets.

As a work list manager, team lead, or a user with a specific role, you can view the status of various tasks as well as have a consolidated view of the tasks taken up by different members of your target in the Overview tab of My Inbox. After opening the Overview page, you can select the target in the left pane. The overview of the corresponding target will be displayed along with a user-level breakdown of tasks in the right pane.

Target (Work list/Team/User) overview

The overview provides a graphical representation of various states of the tasks against their count. You can have a consolidated view on the progress of tasks in that target. You can also set the states for which the graphical representation is to be displayed. Preferences on which states to be displayed by default are saved when the page is closed, and is automatically loaded when you open the page the next time. By default, when the Inbox page is opened for the first time or if no preference is set earlier, the tasks in 'Completed' state are not displayed.

The User level breakdown of tasks section provides the following:

- **For a Work List** - provides the breakdown of all the teams in the work list. On selecting each team, it further displays the user-level breakdown.
- **For a team or a role** - provides the breakdown of the tasks of all the users belonging to a team or a specific role.

The Overview option enables you to reallocate the tasks and balance the workload within the target. Based on the data, you can also decide if you need to add or delete more teams or users to the target to balance the workload.

Setting reminders on a task

Reminders are like alarms. They are triggered or activated before the task for which the reminder has been set is due. You can also set a single reminder for multiple tasks.

Note: You cannot set reminders on Completed or Skipped tasks.

To set reminders on a task:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Right-click the required task and select  (Reminder) from the context menu.
The Add Reminder dialog box opens.
3. Type a subject for **Reminder Subject**.
4. Set the **Days, Hours** and **Seconds** in the respective fields, as required, and click **OK**.
 - When the **Due Date** for the task is set, the reminder will be activated, on the set time before the task is due.
 - When the **Due Date** is not set, the reminder will be activated after the set time, from the current time.
 - While setting the time for reminder, you must set the reminder to execute after 2 minutes or more in order to achieve the desired result.

The reminder is set on the selected task. At the set time, a Reminders dialog box appears, displaying the reminder along with all other reminders set on various tasks, by the current user.

The reminder dialog box contains the **Subject** and the **Due In** fields, that contain the subject of the reminder and the time when the tasks are due respectively.

You can view reminders only when for the claimed or assigned tasks. You can do the following operations in the Reminders dialog box:

- Dismiss the reminder - Select the reminder and click **Dismiss** to dismiss the reminder.
- Open the task - Select the reminder and click **Open** to open the task on which the reminder is set and perform the required operation.
- Snooze the reminder - Select the required reminder and select the time you want to snooze this reminder from the **Snooze For** list.

Delegating an out of office user's tasks

Users who propose to be out of office for a specified period can set the dates and send the information to their corresponding manager. The manager on receiving this information, can further delegate the tasks that are required to be worked upon by the user during this period, to another user (Delegatee).

To delegate an out of office user's tasks:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Click  on the left top corner of the My Inbox.
An Out of Office Information dialog box opens.
3. Click  and select the start and end dates in the respective Starts On and Ends On fields, for the period of unavailability.
4. Type a message in the **Message** field, if required.
5. Click **OK**.
The specified information is sent to the Inbox of the Manager, as a Task Delegation task.
 - If the user belongs to multiple teams, the out of office request is sent to the lead of the principal team that the user is a part of.
 - If the user is a part of a team, but does not have a principal team, then the request is sent to the lead of the corresponding team.
 - If the user is not a part of any team, the Task Delegation is sent to the user's Inbox itself. The user can then delegate those tasks to the relevant user.
6. The manager or user, on receiving the task delegation request in the Inbox, can do the following:
 - a. Open the Task Delegation and click  next to the Delegate field.
 - b. Select the user to whom the tasks will be delegated. Alternatively, the manager or user can type the name of the user (Delegatee) in the input field and select the user from the list of options available in the auto-suggest-options listed.

All the tasks that are to be worked upon in that specific period, by the sender, will be delegated to the selected user.

- The tasks are sent to the Inbox of the user (Delegatee) and are displayed in the <absentee name folder> listed under the Work List(s). The ownership of the tasks will not be transferred to this user. It remains with the original owner of the tasks.
- If the users who opted for out of office access the Inbox during their absence period, they will be prompted to reset the auto delegation period. The users, depending upon their need, can reset the auto delegation period by clicking **Reset** and repeating steps 3 through 5 of this task.

Modifying the system attributes of a task

Users can modify the start date of a task, so that they can plan to take that task at a later point of time. For Case related Activities, if users have the necessary permissions, they can also change the due date of an activity.

- The Start Date of an In-Progress task or activity cannot be modified.
- Start Date should be earlier than the Due Date of the selected tasks.

- Due Date should be later than the Start Date of the selected tasks.
- Due date of a task can be modified only if:
 - users have change due date permission set on their roles, while configuring the authorization for a case activity.
 - that particular activity allows Due Time Change in Inbox, while setting the Duration property of the activity in the business process mode.

To modify the Due Date of a task, users must have the change due date permission set on their roles while configuring the authorizations for a case activity.

To modify the system attributes of a task:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Click the required task from the work list and click  (Modify System Attributes) on the Task Toolbar.
The Modify System Attributes dialog box opens.
3. Click  next to Start Date and select the required date from the calendar that appears.
The selected date is displayed in the New Start Date text box.
4. Click  next to Due Date and select the required date from the calendar that appears.
The selected date is displayed in the Due Date text box.
5. Click **OK**.

The System Attributes of the task are modified.

- When the start date of a task is modified, it is moved to the Calendared Tasks view. To view these tasks, select **Calendared Tasks** from the **View** list seen on the top left corner of the **Personal Tasks** view. All the tasks that must be started at a future date are displayed in the grid below.
- Alternatively, to view the future tasks, select **All Tasks** view on the top left corner of the Inbox. All the tasks that you can work upon, except for the Completed or Skipped tasks, are displayed in the grid.

Tagging a task

Tags allow users to organize information by adding self-created keywords rather than storing information in a predefined hierarchical structure, through the Inbox toolbar, context menu, or the task toolbar.

You can customize the tabs displayed in the Inbox, tasks, or activities, to enhance or minimize the number of items displayed. This reduces the clutter as well as increase the productivity. If you cannot see the Tags tab, click  on the title bar. All the tabs, that are displayed on the interface, are shown in the list as selected. The tabs that are not displayed on the interface are shown as unselected. Select the **Tags** option in the list. To close the **Tag** tab, clear the check mark of the tab you want to close.

To tag a task:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Select the required task from the work list in the Inbox, right-click the task and select **Add Tag** from the context menu. Alternatively, select the task to be tagged and click  on the Inbox task toolbar.
A text box along with Tag button is displayed.
3. Type the required keyword, to identify the tag, in the text box and click **Tag**.

The selected task is tagged.

To view the tags:

1. Click the **Tag** tab on the left side of the Inbox.
A list of all the tags related to the work list are displayed.
2. Click a tag and all the tasks tagged by that name are displayed on the right pane.

Alternatively, you can add or view the tags in the preview section at the bottom of the Inbox, when you click a task in the Inbox. Preview mode should be set by the work list manager while [customizing the work list view](#).

Delegating a task

Tasks can be delegated to other users for several reasons. As a user you may have claimed too many tasks and you feel that you cannot meet the time lines, or you may be assigned an important task which cannot be avoided, or you feel that the task is simple enough that another user can complete it and so on. In such cases, you can delegate the task.

Keep in mind

- The ownership of the task remains with the user who delegated the task.
- A delegated task can be delegated to only one user at a time, so once a task is delegated the DELEGATE action will be disabled.
- Before delegating a task, the user to whom the task is to be delegated must have the necessary permissions to work on it.
- Performing a REVOKECLAIM action on a delegated task revokes the delegation first (also enables the DELEGATE action). The REVOKECLAIM of task ownership will happen only when the action is performed again (which means the REVOKECLAIM action should be performed twice to revoke the claim on a delegated task).
- Similarly the REVOKECLAIM action will be enabled for the delegatee implying that he can revoke upon the delegation (not that he has manager permissions). The delegatee cannot REVOKECLAIM on the delegated task except when is a Manager/Admin.

To delegate a task:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.

2. Right-click the required task from the worklist and click **Delegate**. Alternatively, you can select **Delegate** from the **More Operations** list on the **Task Toolbar**. The **Delegate Tasks** dialog box opens.
3. Click  next to User Name.
The Select User dialog box opens.
4. Select the user that appears and click **OK**.
The name of the selected user is displayed in the User Name text box.
5. Click **Send**.

In case the task is In-Progress state, the delegated user can continue the task from where it was left. The task is delegated to the selected user and will appear in the personal tasks list of the user's Inbox.

Skipping a task

You can skip tasks before they are completed if required. Once the task is skipped, it will be removed from the work list.

To display the skipped tasks in the work list, the work list manager must select the Show Non-workable tasks in the Work list check box, while setting work list preferences on the Inbox.

To skip a task:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Right-click the required task from the work list and select **Skip**. Alternatively, select **Skip** from the **More Operations** list on the **Task Toolbar**. A **Skip Reason** dialog box opens.
3. Provide a reason for skipping the task in the **Memo** text box and click **OK**.
The selected task is skipped and the status of the task changes to  (Obsolete).

It is mandatory to provide a reason for skipping a task

The My Inbox window opens, displaying the personal tasks, by default.

Important: In a business process, if a dependent task is modeled to wait for the completion of this task, then the business process will be in the Wait state until the task status is changed to Complete in the Process Instance Manager.

Keyboard shortcuts (Tasks)

The actions that can be performed on a task from the task toolbar in the Inbox, such as claiming a task, starting a task, adding a memo or a file and so on, can be done by:

- Clicking the corresponding icon on the tool bar.
- Using the keyboard shortcuts.

The following table lists the actions that can be performed on a task in the Inbox, the corresponding keyboard shortcuts used to access these tasks in Internet Explorer, and in other browsers, such as Firefox, Chrome, and Safari.

Double-click the required task in the Inbox to open it. Use the following keyboard shortcuts to manipulate the task.

Action	Internet Explorer	Other browsers
Add Attachment	ALT + N	ALT + SHIFT + N
Add Memo	ALT + M	ALT + SHIFT + M
Add Reminder	ALT + I	ALT + SHIFT + I
Assign Task	ALT + A	ALT + SHIFT + A
Claim Task	ALT + C	ALT + SHIFT + C
Complete Task	ALT + O	ALT + SHIFT + O
Delete Notification	ALT + D	ALT + SHIFT + D
Forward Notification	ALT + F	ALT + SHIFT + F
Links	ALT + L	ALT + SHIFT + L
Next Task	ALT + W	ALT + SHIFT + W
Pause Task	ALT + P	ALT + SHIFT + P
Previous Tasks	ALT + X	ALT + SHIFT + X
Resume Task	ALT + E	ALT + SHIFT + E
Revoke Task	ALT + R	ALT + SHIFT + R
Start Task	ALT + S	ALT + SHIFT + S
Stop Task	ALT + T	ALT + SHIFT + T
View History	ALT + H	ALT + SHIFT + H
View Process	ALT + V	ALT + SHIFT + V

Validations on task actions

When you perform an action on a human task of a BPM, Case Model, or any other source, validations can be performed before and after the specific action is done. For instance, you may want to restrict a user from claiming more than one task at a time. To do that, you can provide your logic in the OnBeforeClaim hook, which is called when a task or the Case activity is claimed and verifies if the user has already claimed it. If it is found to be claimed already, the user is restricted from claiming this task or the Case activity. To do this, you must write a Web library as described in the Sample Validations Library. Refer to the table below for a complete list of available validations and refer to [Creating a Web Library Definition](#) for creating a Web library in CWS.

After creating the Web library, you must call the action validations defined in this Web library as follows:

1. Update the entry in the XML Store Key /Cordys/notification/task/taskoperationlibraries.xml. If this entry is not present, you can create an XML Store definition from CWS.
2. Add the custom validations library for your model as provided in the sample code snippet.

```
<TaskOperationsLibrary>
  <!--You can provide as many model elements as required.-->
  <model fqn="CasesAndBPMs/Sales">
    <libraryurl>cpc.nsinbox.utils.SalesTasksEventHandler</libraryurl>
  </model>
</TaskOperationsLibrary>
```

You can add multiple model nodes in the same XML Store definition entry and specify validation libraries for multiple models.

3. Provide the model names and the relative URLs as many as required in the format provided in the sample code snippet above.

model fqn	Fully Qualified Name of the Case Model or Business Process Model. For example: sample/case/casemodel1
libraryurl	Relative URL of the library that contains the validations. For example: cpc.nsinbox.utils.SalesTasksEventHandler <ul style="list-style-type: none">■ You must provide the logic that validates various actions in the relevant code blocks.■ The SalesTasksEventHandler.htm file must be created in the location mentioned in the libraryurl; for instance, /cordys/cpc/nsinbox/utils folder as provided in the code snippet above. While performing the required operations on the tasks, the relevant prototype methods are invoked. <p>Remove the prototype methods, which are not required from the file. This reduces the file size, thereby reducing the loading time.</p>

The following tables describe the various validations that can be performed on the task actions.

ASSIGN action

OnBeforeAssign	Fired before a task is assigned
OnAfterAssign	Fired after a task is assigned

CLAIM action

OnBeforeClaim	<p>Fired before a task is claimed</p> <ul style="list-style-type: none"> ▪ When the Automatically Claim / Revoke Claim in Inbox check box is selected while configuring the Notification service container and you perform some action such as starting the task, the validation is done in the following order: <ul style="list-style-type: none"> • Claim • OnBeforeClaim • OnAfterClaim • OnBefore<action> - for example: OnBeforeStart • OnAfter<action> - for example: OnAfterStart ▪ If the Automatically Claim / Revoke Claim in Inbox check box is selected, and you open an unclaimed task, view and close it without performing any actions, no validation is done on the task or Case activity. ▪ If the Automatically Claim is not enabled and you claim the task and close it without performing any operation, the OnBeforeClaim and OnAfterClaim events are fired.
---------------	---

REVOKECLAIM action

OnBeforeRevokeClaim	Fired before a claimed task is revoked
OnAfterRevokeClaim	Fired after a claimed task is revoked

COMPLETE action

OnBeforeCommit	Fired before a task is completed
OnAfterCommit	<p>Fired after a task is completed</p> <p>If you want to override the action to open the next available task from the related activities in the Case instance, the <code>Workflow.taskView.openTask()</code> API can be used. Also, the <code>dataObject.dontOpenNextActivity</code> flag must be set to true.</p> <pre>//In the prototype OnAfterComplete, add the following logic to override the default behavior. ActivityHookLib.prototype.OnAfterCommit = function(dataObject, callBackFunction) { Workflow.taskView.openTask(sTaskId); dataObject.dontOpenNextActivity = true; callBackFunction(dataObject); }</pre> <p>If you do not want to open any activity but open Case instance view, set the <code>dataObject.openCaseInstanceView</code> flag to true.</p>

```
//In the prototype OnAfterComplete add the following code to
open Case instance view.
ActivityHookLib.prototype.OnAfterCommit = function(dataObject,
callBackFunction)
{
    dataObject.openCaseInstanceView = true;
    callBackFunction(dataObject);
}
```

DELEGATE action

OnBeforeDelegate	Fired before a task is delegated
OnAfterDelegate	Fired after a task is delegated

FORWARD action

OnBeforeForward	Fired before a task is forwarded
OnAfterForward	Fired after a task is forwarded

PAUSE action

OnBeforePause	Fired before a task is paused
OnAfterPause	Fired after a task is paused

RESUME action

OnBeforeResume	Fired before a task is resumed
OnAfterResume	Fired after a task is resumed

START action

OnBeforeStart	Fired before a task is started
OnAfterStart	Fired after a task is started

STOP action

OnBeforeStop	Fired before a task is stopped
OnAfterStop	Fired after a task is stopped

SUSPEND action

OnBeforeSuspend	Fired before a task is suspended
OnAfterSuspend	Fired after a task is suspended

TRANSFER action

OnBeforeTransfer	Fired before a task is transferred
OnAfterTransfer	Fired after a task is transferred

SKIP action

OnBeforeSkip	Fired before a task is skipped
OnAfterSkip	<p>Fired after a task is skipped</p> <p>If you want to override the action to open the next available task from related activities in the Case instance, the <code>Workflow.taskView.openTask()</code> API can be used. Also, the <code>dataObject.dontOpenNextActivity</code> flag must be set to true.</p> <pre>//In the prototype OnAfterSkip add the following logic to override the default behavior. ActivityHookLib.prototype.OnAfterSkip = function(dataObject, callBackFunction) { Workflow.taskView.openTask(sTaskId); dataObject.dontOpenNextActivity = true; callBackFunction(dataObject); }</pre> <p>If you want to open the Case instance view without opening any activity, set the <code>dataObject.openCaseInstanceView</code> flag to true.</p> <pre>//In the prototype OnAfterSkip, add the following code to open the Case instance view. ActivityHookLib.prototype.OnAfterSkip = function(dataObject, callBackFunction) { dataObject.openCaseInstanceView = true; callBackFunction(dataObject); }</pre>

ADD MEMO action

OnBeforeMemo	Fired before a memo is added to the task
OnAfterMemo	Fired after a memo is added to the task

ADD ATTACHMENT action

OnBeforeAttachment	Fired before a file is attached to the task
OnAfterAttachment	Fired after a file is attached to the task

UPDATE ATTACHMENT action

OnBeforeUpdateAttachment	Fired before an attachment to a Case activity is updated. Applicable for Case activities only.
OnAfterUpdateAttachment	Fired after an attachment to a Case activity is updated. Applicable for Case activities only.

DELETE ATTACHMENT action

OnBeforeDeleteAttachment	Fired before an attachment to a Case activity is deleted. Applicable for Case activities only.
OnAfterDeleteAttachment	Fired after an attachment to a Case activity is deleted. Applicable for Case activities only.

PLAN INTERMEDIATE FOLLOWUPS action

OnBeforePlanInterMediateFollowups	Fired before the intermediate follow-ups are planned from Case activity. Applicable for Case activities only.
OnAfterPlanInterMediateFollowups	Fired after the intermediate follow-ups are planned from Case activity. Applicable for Case activities only.

PLAN FREE FOLLOWUPS action

OnBeforePlanFreeFollowUps	Fired before free follow-ups are planned from the Case activity or Case instance view. Applicable for Case activities only.
OnAfterPlanFreeFollowUps	Fired after free follow-ups are planned from the Case activity or Case instance view. Applicable for Case activities only.

OPEN action

OnAfterOpen Properties: isCaseOverviewRefreshRequired	Fired after a Case activity is opened. By default, the Case Overview tab refreshes while switching between the activities. If you do not want this behavior, set the <code>dataObject.isCaseOverviewRefreshRequired</code> flag to <code>false</code> . <code>//In the prototype OnAfterOpen add the following code, so that the data or UI in the Case Overview tab will not be refreshed while switching between the activities of the same Case model.</code> <code>ActivityHookLib.prototype.OnAfterOpen = function (dataObject, callBackFunction)</code> <code>{</code> <code> dataObject.isCaseOverviewRefreshRequired = false;</code> <code> callBackFunction(dataObject);</code> <code>}</code>
---	---

CLOSE action

OnBeforeClose	Fired before a Case activity is closed
---------------	--

All the validations provided in the table above require `dataObject` and `callback` function as parameters.

dataObject- The `dataObject` parameter has `CaseInstanceId`, `TaskId`, `ReturnValue`, and `Workflow` object as task data. You can retrieve the task data as shown below:

- `CaseInstanceId` or `ProcessInstanceId`:

```
for(var ctr = 0; ctr < SelectedOptions.length; ctr++)
{
    alert(dataObject.SelectedOptions[ctr].SourceInstanceId);
}
```

- `TaskID`:

```
for(var ctr = 0; ctr < SelectedOptions.length; ctr++)
{
    alert(dataObject.SelectedOptions[ctr].TaskId);
}
```

- `ReturnValue`:

```
for(var ctr = 0; ctr < SelectedOptions.length; ctr++)
{
    alert(dataObject.SelectedOptions[ctr].ReturnValue);
}
```

- Workflow - `dataObject.Workflow`.

callback function - This function is used to give the control back to the framework. Ensure that you do not make any changes to this function, and your validation logic ends with it.

Creating a Web Library Definition

Before you begin:

- You must have uploaded web files into the project.

A Web Library Definition enables you to package and deploy files, such as `.htm`, `.html`, and `.js` that are used in the application being created, to the AppWorks Platform web server. A Web Library Definition bundles them in a suitable format so that they can be deployed within an application and used at runtime.

To create a web library definition:

1. Select a starting point and click  to open the Web Library Definition editor.
2. Enter the following information.

Name	Name of the Web Library Definition file. Provide a name.
Description	Description of the Web Library Definition. Provide a brief description.
Web-content Folder	The folder from which the web files will be included. These web files will be used in the Application Package. The documents in this folder will be validated and published according to the publish logic of Web Library Definition. Note: Ensure that the folder does NOT contain the following: <ul style="list-style-type: none">■ any artifacts other than the web related files such as .htm, .html, and .js■ contains any references to projects Click  to browse and select the source content project/folder.

3. Click .
4. Provide the Name and Description, and click  to browse and select the location to save the Web Library Definition file.
The folder path where the Web Library Definition will be created appears in the Save in Folder field.
5. Click **Save**.
6. Close the **Web Library Definition** window.

The Web Library Definition is created with the provided name and is stored at the selected location. When you package the Web Library Definition, it stores the packaged web files that are in the scope of the Web Library Definition, in the Application Package (.isvp) file.

After you complete this task:

- **Publish** the Web Library Definition to an organization.

When deploying an application that contains a Web Library Definition (or when publishing such a project), the documents in the corresponding Web content folder will be deployed to the AppWorks Platform web server. The exact location at which the documents will be deployed is based on their qualified name.

Attaching a memo to a task

You can create memos and attach them to the tasks, to track information you wish to store for future reference, through the Inbox toolbar, context menu, or the task toolbar. These memos can also be used to share information with team members. Memos can be forwarded, or delegated to other users, as part of the task to which they are attached.

You can customize the tabs displayed in the Inbox, tasks, or activities, to enhance or minimize the number of items displayed. This reduces the clutter as well as increase the productivity.

- If you cannot see Memos tab, click  on the titlebar.
- All the tabs that are displayed on the interface are shown in the list as selected. The tabs that are not displayed on the interface are shown as unselected.
- Select the **Memos** option in the list.
- To close the **Memo** tab, clear the check mark of the tab you want to close.

To attach a memo to a task:

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Select the required task from the work list in the Inbox and select  on the Inbox task toolbar. Alternatively, right-click the required task, select **Memo > Add Memo**. A **New Memo** dialog box is displayed.
3. Type the required information in the Memo text box and click **Add**.
The memo is added to the task.
4. To view a memo, right-click the required task in the Inbox and select **Memo > View Memo**.
The memos related to the task are displayed in the Memos pane.
Alternatively, you can click the required task and add or view a memo attached in the preview section at the bottom of the Inbox. Preview mode should be set by the work list manager while customizing the work list view. See [Customizing the Inbox view](#).

To add, delete, or modify memos:

- In the toolbar of the Memos pane, click , , or .
- Only the user who created the memo can update or delete it, while other users can only view it.

You cannot delete:

- A memo that was provided while skipping a task. Once the task is skipped, even the users who created the memo cannot update or delete it.
- A memo that was provided while transferring a task.
- A memo that was provided as instructions for doing a task.

- Memos of the tasks that were completed. For tasks that were completed, memos can only be viewed; they cannot be deleted or modified.

Working with cases in the Inbox

To work with cases in the Inbox:

1. On the Welcome page > My Applications App Palette, click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default. Both case model tasks and lifecycle model tasks are displayed.
2. Click the **Case(s)** tab in the left pane of My Inbox.
All the case models are displayed. Lifecycle models are not displayed.
3. Click the required <case model>.
All the case instances to which the user is associated are displayed on the right-side.
Click the required case instance to display the related activities.
4. Do any of the following to open the Case Instance Overview:
 - Click the required <case instance> to display the related activities in the Case Activities section below.
 - Double-click the required <case instance> to view the Case Instance Overview.
The Case Instance Overview page is displayed.
For more information on the Case Instance Overview interface, see [Case Instance Overview Interface](#).
5. Do any of the following to open the Case Activity Overview:
 - If a certain action can be performed on an activity without opening it, select the required <activity> and perform the necessary actions from the task tool bar.
IMPORTANT: Only the automatic followups are triggered if the activity is completed from this view. For more information on the various actions that can be performed on an activity, see [Working with Inbox](#).
 - Double-click the required activity. The Case Activity View page appears, displaying all the released case activities for that instance in the left pane. For more information on the Case Activity Overview interface, see [Case Activity Overview Interface](#).

You can perform the required operations on the cases.

Additional procedures

- You can attach various file formats to the case models and case activities that are displayed in the AppWorks Platform Inbox. See [Attaching files to a Case](#) for more information on the procedure to attach files to cases.
- You can know the contact information of an assignee by placing the mouse pointer on a name in the Assignee column. A tooltip containing the contact information of the assignee appears.
- To view the history of cases, select the required case and click  on the Task tool bar.
A context menu appears, displaying Task History and Case History options.

- Select Task History to view the history of the task that is opened. An Item History page appears, displaying the status of the selected task.
- Select Case History to view the history of all the activities of the case instance. A Case Instance History page appears, displaying the states of various case activities on the Case Activity tab and the overview of the case on the Overview tab. Refer to Case Activities Interface for more information on the fields in the Case Instance History page.
- To view the history of a particular task, do any of the following:
 - Click the required task in the Case Instance History page, and click  on the tool bar. A Task page appears, displaying the status of the selected task.
 - Right-click the required task and select **Show History**.
A Task page appears, displaying the status of the selected task.
 - Double-click the task to open it in a Read-Only mode.
This helps in knowing the details of the task. Alternatively, you can right-click the task and select Show Details. The task is opened in a Read-Only mode.

Case instance overview interface

Events	<p>This section contains all the events that can be triggered on the Case Instance.</p> <p>Select the required events to be triggered and click Trigger Events.</p> <p>Free Followups - Activities that have no direct relation to the sequence of activities. They can be set on any state in the case model. This view displays only Free Followup(s).</p> <p>Right-click the required free followup and select any of the following:</p> <ul style="list-style-type: none"> ▪ Trigger and Assign to Me - assigns the followup to the current user. <p>Ensure that you have the necessary permissions to work on the task that will be triggered.</p> <ul style="list-style-type: none"> ▪ Trigger with Default Attributes - triggers the follow-up and assigns it as per the attributes specified while modeling the case. <p>Trigger with Custom Attributes - triggers the follow-up and allows you to select the attributes such as assignee type, assignee value, due on, priority and so on. On selecting this option a Trigger Followups page appears. Select the required attributes and click OK. See Selected Followups section for more information on the fields shown in this page.</p>
Followup(s)	
Related Activities	<p>This section contains all the released activities related to this case.</p> <p>Perform the necessary action.</p> <p>For more information on the various actions that can be performed on an activity, see Working with Inbox.</p>

	<p>The name of the Assignee and the attributes defined to be displayed as columns in the Inbox while modeling the Inbox model details are displayed below the related activities of a Case Activity.</p> <p>Double-clicking the required activity displays the Case Overview and Activity Overview tabs on the right-side.</p> <ul style="list-style-type: none"> ■ Case Overview - a case overview form that contains the summary of the complete case, as defined during case modeling. ■ Activity Overview - contains the case activity form that is defined during case modeling. For more information on the activity overview, refer to Activity Overview.
--	--

Select followups interface

The Select Followups page is divided into the following sections:

- The **Applicable Followups** section contains the list of all the possible followups that can be triggered on completion of current activity.
- The **Selected Followups** section contains the followups that are selected from the applicable followups to be triggered on completion of the current activity. If required, the attributes of the activities, such as, **Assignee Type**, **Assignee value**, **Due On** dates and so on, can be modified in this section.

1. From the list in the **Applicable Followups** section, select the required followups and click **Add**.

Automatic Followups	<p>All the selected automatic followups will be triggered. If an activity is completed without opening it, only the automatic followups are triggered.</p> <p>All the automatic activities are selected, by default.</p> <ul style="list-style-type: none"> ■ Select or clear the selection of the required activities and click OK. <p>While defining the case model, if the followups are set as mandatory, they are selected and are non-editable.</p>
Manual Followups	<p>All the selected manual followups will be triggered.</p> <ul style="list-style-type: none"> ■ Select Manual. A Select Manual Activities section is displayed below. ■ Select the required manual activities and click OK.
Free Followups	<p>All the selected free followups will be triggered.</p> <ul style="list-style-type: none"> ■ Select Free Followups. A list appears, displaying various states of the case model. You can filter the free followups per state in a case model. ■ Select the required state from the list.

	The list box below is populated with the activities of the selected state. Click OK .
2.	The followups that are selected to be triggered for the current activity are displayed in the Selected Followups section below.
Assignee Type	<p>The target for assigning the activity. You can assign the followup to the following:</p> <ul style="list-style-type: none"> ■ Work list ■ Team ■ Role ■ User <p>Select the type of target to assign the followup from the list.</p>
Assignee Value	<p>The value for the assignee depending upon the assignee type selected.</p> <ul style="list-style-type: none"> ■ Click  , select the corresponding value from the Select dialog box that appears and click OK. Based on the assignee type selected, the Select dialog box contains the following: <ul style="list-style-type: none"> • Work list - Contains the name of the work list. • Team - Contains the name of the team. • Role - Contains the role description and the DN of the role. • User - Contains the name of the user.
Due On	<p>The date the followup is due.</p> <p>Click  and select the required date.</p>
Priority	<p>The priority for the followup.</p> <p>Select the required priority from the list.</p>
Instructions	<p>Instructions or information to the assignee.</p> <ul style="list-style-type: none"> ■ Click  , type the instructions in the Add Memo dialog box that appears and click OK.

3. Click **OK**.

The selected followups are triggered.

To remove a followup from the list:

- Select it from the **Selected Followups** section and click **Remove**.

Creating an e-mail model

E-mail models are used to personalize e-mail messages and sent to the user's e-mail addresses. The E-mail model has a rich text editor that can be used to configure the e-mail messages. It provides the ability to:

- format the text through flexible font styles, sizes, bold facing, color schemes
- embed tables
- indent and align text
- drag-and-drop message data from the messages

Based on the need, e-mail models can be created in the following ways.

Create a static E-mail model

The standard E-mail Model editor enables a user to model e-mail messages with static e-mail content. If there is no input source (XML Schema or User Interface) defined for an E-mail Model, it is considered as a static e-mail model.

To create an E-mail Model on XML Schema document:

- Select a starting point and click  to open and work with the E-mail Model editor.

Create an E-mail model based on an XML schema document

When an XML Schema document is used to create the E-mail Model, it will serve as input source of data. The schema content can be used in any of the E-mail Model components to generate dynamic e-mail content.

This E-mail Model enables the users to generate a Web Service on it and the XML Schema and is the input parameter for the Web Service. The input schema will also contain parameters to configure To, CC, BCC, and Sender address details. These parameters must be filled in before triggering the Web Service.

To create an E-mail Model based on an XML Schema document:

1. Select a starting point and click  to open the E-mail Model editor.
2. Select the XML Schema Fragment from the Message tab in the editor as the input source.
3. See [working with E-mail Model editor](#), for more information on using the E-mail Model editor.
4. You can [generate a Web service on this E-mail Model](#).

Create an E-mail Model on a user interface document

You can personalize and model the e-mail message on User Interface document (Tasks). The input message of the task will be the input source of data while modeling the e-mail message. When the Business process models are executed, the tasks are delivered to the target users. If the users have configured the preference to receive tasks to their e-mail address, the e-mail model created on User Interface will be delivered.

E-mail Model on a User Interface can be created using [Delivery Model](#). These e-mail models can now be [selected from the Workflow Model tab](#), while setting the properties of an activity in a Business Process. On processing the activity, the task is sent to the user's email as per the created email model.

To create an E-mail model based on a user interface:

- Click the **E-mail Model** tab in the [Delivery Model](#) and [do the following](#) in the E-mail Model editor.

Working with the Email model editor

The following procedure describes the creation of an Email Model.

To work with the email model editor:

1. Type the **Name** and **Description** of the Email model in the respective fields.
2. Type the subject of the email in the **Subject** field, which will be displayed to the users when they receive mails to their email ID.
3. Select the message format, **HTML** or **Text**.
 - Select **HTML** format, if you want to customize the layout and have a graphical view of the message.
 - Select **Text** format, if you want a plain view of the message. Based on the message format selected, the corresponding editor appears.
4. If the message format is **HTML**, a layout editor is displayed in the Email Model pane below. Do the following:
 - a. Select the layout type from the **Select Layout** list box on the **Email Model** pane.
 - By default, AppWorks Platform provides four types of layouts. Layouts describe the way the email is modeled, for example the placement of the logo on the page, the placement of the link and so on.
 - You can select one of the default layouts or select the layout type that you defined earlier. In order to access these layout types from the **Select Layout** list box, the XSLT of these layout types must be stored in the following location in the XML Store: **/Cordys/notification/emailmodels/xslt/**.
 - b. Click  next to Set Style field, select the required styles from the Style Sheet Editor, and click OK.

- c. Type the path of the logo in the **Logo Path** text box.
You can provide both the absolute path as well as relative path of the logo. The relative path should have the **URL prefix** provided while configuring the notification processor.
 - d. Compose the email using the attributes in the **Model Components** and the Message panes as follows:
 - Click the required attribute in **Model Components** attribute, for example, Header. A Header editor is displayed on the right. Make the required modifications in the editor.
 - Expand the application data and message metadata elements in the Message group box and drag the required attributes that you want to display as in the mail, from the XML Structure to the layout editor.
 - e. If required, click **Edit Model Source** to view the XML of the model components, make the necessary changes to the XML, and click Preview button to view the changes.
You cannot modify or delete the names of the elements that are displayed in the XML. For example: You cannot change or delete the names of the elements, like **Header**, **Greetings**, and so on.
5. If the message format selected is **Text**, type the required message in the text box. To use the XML elements from the model tree structure of the message data, drag them to the text box.
 6. Click  on the toolbar.

The Email Model is created.

The Email Models can now be selected from the Workflow Model tab, while setting the properties of an activity in a Business Process. On processing the activity, the task is sent to the user's email as per the created email model.

The created Email Models are saved along with the Delivery model in the Workspace Documents (Explorer). To make further changes to the email model, open the corresponding delivery model, access the email model tab, and modify it as per the requirement.

Adding a workflow library to user interface applications

Before you begin:

- You must have the process developer role in the AppWorks Platform Business Process Engine application package to add a workflow library to a User Interface (UI) application.

Application services can be of different types, such as XForms, or other Web Page URL-based UI applications such as HTML, JSP, and so on.

An application service can be workflow-enabled by adding a workflow library to it. For an XForm, the wrappers for the corresponding application handle the workflow. For a UI application, the workflow library has to be added by an application developer.

To add a workflow library to user interface applications:

1. Open the <UI page> to which you are adding the workflow library in a text editor.
For example, consider an HTML application such as `clickappservice.htm`.
2. Provide the HTML element `<div>` under the body tag.
3. Provide `<div id='workflowEnabling' style='display:none'></div>` within this tag.
The `<div>` tag will not be displayed as the workflow library does not have a user interface.
4. Add `application.addLibrary('/cordys/cas/vcm/library/workflow.htm', workflowEnabling);` to the `<div>` tag to add the workflow library during startup or initialization of the page.
The workflow library is added to the `<div>` tag.
5. Register the function `(workflowEnabling.setPostInitializeFunction (postInitialize);`.
This function is called when initialization succeeds. The function is registered.
6. Add `workflowEnabling.initializeWorkflow();` to call the `initializeWorkflow()` function to initialize workflow synchronization (for example: retrieving task details).
The `postInitialize` function must take the data from the APIs found in the `/cordys/cas/vcm/library` and render the data in the UI applications.
For more information on the library functions, see the *AppWorks Platform API Guide*.
7. Add `application.removeLibrary('/cordys/cas/vcm/library/workflow.htm', workflowEnabling);` to remove the workflow library from the element to which it was added when the page is closed or unloaded.
The workflow library is removed.

The application service is workflow enabled.

Adding a workflow library to HTML applications

Before you begin:

- You must have the process developer role in AppWorks Platform Business Process Engine application package to add a workflow library to an HTML application.

Application services can be of different types, such as XForms and HTML applications. An application service, such as `choiceappservice.htm` can be workflow-enabled by adding a workflow library to it. For an XForm, the wrappers for the corresponding application handle the workflow. For an HTML application, the workflow library has to be added by an application developer.

To add a workflow library to HTML applications:

1. Click <AppWorks Platform_installdir> > Web > cas > vcm > casrepository > applicationservices > content > workflow.
The Application services that are HTML pages are displayed.
2. Open the HTML page to which you are adding the workflow library, in a text editor.
3. Type the HTML element <div> under the body tag.
4. Type <div id='workflowEnabling' style='display:none'></div> within this tag. The <div> tag will not be displayed as the workflow library does not have a user interface.
5. Add application.addLibrary('/cordys/cas/vcm/library/workflow.htm', workflowEnabling); to the <div> tag to add the workflow library during startup or initialization of the page. The workflow library is added to the <div> tag.
6. Register the function workflowEnabling.setPostInitializeFunction(postInitialize);. This function is called when initialization succeeds. The function is registered.
7. Add workflowEnabling.initializeWorkflow(); to call the initializeWorkflow() function to initialize workflow synchronization (getting task details, etc.).
8. Add application.removeLibrary('/cordys/cas/vcm/library/workflow.htm', workflowEnabling); to remove the workflow library from the element to which it was added, when the page is closed or unloaded. The workflow library is removed.

The application service is workflow enabled.

Chapter 11

Modeling roles

AppWorks Platform has many predefined roles such as System Administrator, Organizational Administrator, and Developer. These roles are associated with a prescribed set of rights or activities, which a role is authorized to perform. An application contains many activities to perform, but a role need not be allowed to perform all the activities or access all the features. This calls for simpler roles with lesser number of activities or fine-grained tasks or applications. Based on the authorization or access level, the application will enable or disable activities for that user. For example, a developer must be allowed only to view the Application details and not to install an Application.

AppWorks Platform provides a mechanism to define a set of activities for a given user depending on the role(s) associated with that user. This facilitates the user to work only on those predefined activities or features and disable other rights.

You can be aware of:

- Who is accessing the application
- What is the user's role(s)
- Which task(s) the user can perform

Typically, multiple users with multiple roles and authorization levels use AppWorks Platform with different goals. AppWorks Platform allows you to define custom roles depending on the user requirement and assign task(s) or activities to that role in the deployed environment. Using task, you can manage and maintain the tasks depending on the roles or user.

A **task** refers to an activity that can be performed independently.

A **configured task** refers to a group of task parts, which can be enabled or disabled for user access. You can configure task parts while assigning tasks to users or roles. See [Configured Task definition](#) for information on the associated properties.

Tasks and Configured Tasks are stored in a Task Repository that is a part of AppWorks Platform Repository. Task framework provides Web services and Java API to retrieve task Information. A task is automatically generated when you create a user interface.

Roles determine the access privileges and activities a user can perform in an organization. They facilitate users to have a set of permissions by means of which only they can perform specific tasks. This in turn is based on the Access Control List (ACL) settings on that role.

You can access the role modeler in multiple ways. For more information, see [Creating a Model](#).

Modeling roles involves the following tasks:

- [Creating a Role](#)
- [Editing or Deleting a Role](#)
- [Assigning Sub-Roles to Roles](#)
- [Assigning Tasks to Roles](#)

Creating roles

Creating a role helps to define the privileges that you can associate with a user. This enables the user to use certain features of AppWorks Platform and perform specific tasks.

Before you begin:

- [Create a project.](#)

To create roles:

1. Select a starting point and click  (Role editor).
The <UntitledRole> Role window opens.
By default, the Functional role type is selected.
See Understanding Roles in the *AppWorks Platform Overview* to select the appropriate role type.
2. Provide the appropriate name and description for the role in **Name** and **Description**.
3. Click **Save**.
4. Alternatively, do the following:
 - a. Click **Save** before providing the name and description in the <UntitledRole> Role window.
The Save Document dialog box opens.
 - b. Provide the **Name** and **Description**.
 - c. Click **Save**.
5. After you complete:
 - [Publish the role](#) to deploy it in the current organization.
 - After you create a role, you can assign sub-roles and tasks to it.

Editing or deleting a role

You can modify the role details such as name, description, sub-role, or task assignment in the role editor window.

Before you begin:

- [Create a role.](#)

To edit a role:

1. In the My Recent Documents or Explorer view of the Workspace Documents window, double-click the role.
For information about changing the view, see Working with Workspace Documents.
The <Rolename> Role window opens.
2. In the <Rolename> Role window, you can do one of the following:
 - Change the role details:
 - Click  (Quick Access Menu) > Properties.
 - In the General tab of the <Name of the Role> - Role dialog box, change the necessary details and click **OK**.
 - Change the role of task assignment.
3. Click **Save**.
4. [Publish the role](#) to deploy it in the current organization.

The details of the role are edited.

To delete a role, do one of the following:

- In the My Recent Documents view:
 - Position the pointer over the role and click beside the name of the Role.
 - Select **Delete** on the context menu.
- In the Explorer view:
 - Go to the role, right-click it and select **Delete**.
The role is deleted.

Assigning menus and toolbars to users

Before you begin:

- You must have the role of a `systemAdmin` to assign menus and toolbars to a user.

Menus and Toolbars were supported in the earlier versions and this topic aims to indicate the backward compatibility.

1. In the User Manager window, select Users - Roles or Users - Tasks view and click . The Manage Menus and Toolbars dialog box opens, listing Users and Menus and Toolbars panes.
Menus are represented as  and Toolbars are represented as .
2. Select the user in the **Users** pane.
The User Name and User ID are displayed for every user in the Users pane.
3. Select the menu or toolbar, right-click and select the **Assign to Selected User (s)** option.

Tip: Drag the menu or toolbar from the **Menus and Toolbars** pane to the Users pane.

The menus or toolbars are assigned.

4. To revoke the menu or toolbar from the user, right-click them and the **Remove from Selected User(s)**.

After you complete this task:

- Menus and toolbars assigned to you are displayed in the **Views** tab of User Preferences.
- Double-click <Menus> to view the artifacts appended to the menu and double-click the artifacts in the Menus window that appears to launch the application.

Assigning user interfaces to roles

Before you begin:

1. Create an External User Interface or a User Interface application for assignment. See [Creating a task part for a control](#) or [Creating XForms using web services](#)
2. [Create roles](#).

When you assign user interfaces to a role, you can prescribe the set of activities to associate with the role. The role governs the activities the user can perform. You can assign a user interface in the role editor window.

To assign user interfaces to roles:

1. In the My Recent Documents or Explorer view of the Workspace Documents window, double-click the role.
For information about changing the view, see [Working with Workspace Documents](#).
The <Rolename> Role window opens.
2. You can assign user interfaces in any of the following ways.

Quick Access Menu	<ol style="list-style-type: none">1. In the <Rolename> Role window, click Quick Access Menu. The Quick Access Menu dialog box opens and displays the existing models in the right pane.2. Select a user interface you intend to assign and drag it from the Quick Access Menu dialog box to the User Interfaces group box of <Rolename> Role window.3. In the Quick Access Menu, select Publish to make the updated role available for the current organization during deployment.
Role Editor	<ol style="list-style-type: none">1. Click in the User Interfaces group box of the <Rolename> Role window. The Select User Interface dialog box opens.2. Select a user interface.

Workspace Documents	<ol style="list-style-type: none"> 3. If there are no user interfaces added to the role: <ol style="list-style-type: none"> a. In the User Interfaces group, click Click here to add User Interfaces. The Select User Interface dialog box opens. b. Select a user interface and click OK. 1. Select a user interface from the My Recent Documents or Explorer view. 2. Drag it to User Interfaces group box of <Rolename> Role window. 3. Click Save. 4. In the Explorer view of the Workspace Documents window , right-click the role and select Publish to Organization.
---------------------	--

A user interface is assigned to the role.

To revoke the user interface assigned to a role:

1. In the User Interfaces group box of the <Rolename> Role window, point to a user interface and click .
2. Select **Remove**.
3. Click **Save**.
4. [Publish the role](#) to deploy it in the current organization.
5. Alternatively, select one or more user interfaces while pressing the CTRL key and click .

Remember

- A User Interface may involve multiple sub-tasks or sub-activities to complete an activity.
 - In such cases, you are prompted with the list of sub-tasks in the Configure Task Parts dialog box.
 - Select the sub-tasks you intend to associate with the role and click **OK**.

If you want to edit the sub-task assignment for a role:

1. Position the pointer over the user interface and click .
2. Go to **Action > Configure** on the context menu and change the configuration in the Task Part Details dialog box.
This is not applicable for external user interfaces.
3. Alternatively, select one or more user interfaces while pressing the CTRL key and click **Configure Tasks** icon in the User Interfaces group box.
See [Configured task definition](#).
Sub-tasks are also referred as Task Parts.

Assigning users to roles

Before you begin:

- You must have the role of a systemAdmin or organizationalAdmin to assign users to roles.

Roles determine the access privileges and activities a user can perform in an organization. By assigning users to roles, you are configuring the privileges for the user.

To assign users to roles in multiple forms:

- From the Welcome page > My Applications, click  (User Manager) and select Roles - Users view in the User Manager window.

To assign a single user to a single role:

1. Select a role in the **Roles** pane.
2. Select a user in the **Users** pane.
3. Right click the user and click **Assign to Selected Role(s)**. The required user is assigned to the selected role.

To assign a single user to multiple roles:

1. Select multiple roles in the Roles pane by holding down the CTRL key.
2. Select a user in the **Users** pane, right click it and click **Assign to Selected Role(s)**. The required user is assigned to the selected roles.

To assign multiple users to a single role:

1. Select a role in the Roles pane
2. Select multiple users in the **Users** pane, right click them and click **Assign to Selected Role(s)**. The required users are assigned to the selected role.

To assign multiple users to multiple roles:

1. Select multiple roles in the Roles pane by holding down the CTRL key.
2. Select multiple users in the **Users** pane, right click them and click **Assign to Selected Role(s)**. The required users are assigned to the selected roles.

Tip:

- Alternatively, drag user(s) from **Users** pane to the selected role(s) in the **Roles** pane.
- Select **Include internal roles** check box to view internal roles such as everyone role. You can assign users to these roles also.

Assigning multiple roles to a user in a team

Before you begin:

- You must have the role of a systemAdmin or organizationalAdmin.

A user can perform more than one activity in a team. Hence, the user would require multiple roles to work on multiple assignments in a team. You can perform this activity in **Users - Teams** or **Teams - Users** view.

In the Users - Teams view

1. Select a user and a team in the **Users - Teams** view of the Manage Users window.
2. Right-click the team and select **Assign to Selected User(s)**.
The Add Teams dialog box opens.
The team name remains the same in the Teams field.
3. From the **Role** list select a role.
4. Click .
A new row is appended to the table.
5. Select a role from the Role list and click **Save**.

In the Teams - Users view

1. Select a user and a team in the **Teams - Users** view of the Manager Users window.
2. Right-click the user and select **Assign to Selected Team(s)**.
The **Add Users** dialog box opens.
The user name remains the same in the **Users** field.
3. From the **Role** list, select a role.
The Add Users dialog box opens.
4. Click .
A new row is appended to the table.
5. From the **Role** list, select a role and click **Save**.

Repeat the procedures to configure more roles. Multiple roles are assigned to a user.

Assigning sub-roles (Managing roles)

You can extend the functionality of a role by assigning other roles to it. The roles that are being assigned are referred as sub-roles. By assigning sub-roles, you are deriving and grouping a set of privileges of the existing roles and assigning them to another role. You can assign custom roles or runtime roles in the role editor window.

Before you begin

- You must create roles or reuse roles of deployed applications.

To assign sub-roles:

1. In the My Recent Documents or Explorer view of the Workspace Documents window, double-click the role.

For information about changing the view, see Working with Workspace Documents.

The <Rolename> Role window opens.

2. You can assign sub-roles in any of the following ways.

Quick Access Menu	<ol style="list-style-type: none">1. In the <Rolename> Role window, click  (Quick Access Menu). The Quick Access Menu dialog box opens and displays the existing models in the right pane.2. Select a role you intend to assign and drag it from the Quick Access Menu dialog box to the Sub Roles group box of <Rolename> Role window.3. In the Quick Access Menu, select the Publish option to make the updated role available for the current organization.
Workspace Documents	<ol style="list-style-type: none">1. Select a role from the My Recent Documents or Explorer view.2. Drag it to Sub Roles group box of <Rolename> Role window. Alternatively, click in the <Rolename> Role window, select a role in the Select Sub Role dialog box, and click OK.3. Click Save.4. In the Explorer view of the Workspace Documents window, right-click the role and select Publish to Organization.

Roles are assigned as sub-roles.

To revoke the role, do the following:

1. In the Sub Roles group box of the <Rolename> Role window, move the pointer over the role and click beside the Role name.
2. Select **Remove**.
3. Click **Save**.
4. Publish the role to deploy it in the current organization. See [Publishing a document to an organization](#)

Assigning roles from multiple teams to a user

Before you begin:

- You must have the role of a systemAdmin or organizationalAdmin.

User Manager enables you to assign roles from different teams to a user in a single interface.

1. Select a user and multiple teams in the **Users - Teams** view of the **Manage Users** window.
You can select multiple teams by holding down the CTRL key.
2. Right-click the teams and select **Assign to Selected User(s)**.
The Add Teams dialog box opens. The teams that are selected in the Users - Teams view are displayed.
3. Select a team.
The roles that are available for that team are displayed.
4. Select a role and click **Save**.
5. To add more, click  in the Add Teams dialog box.
A new row is appended to the table.
6. Repeat the applicable steps to map the users and roles as required.

Roles from multiple teams are assigned to a user.

Configured task definition

Configured Task refers to a group of tasks with a set of permissions granted, based on the role or user. Depending upon the role or user, a set of sub-tasks and related tasks are configured for that role or user. The following code snippet indicates the configured task definition.

```
<ConfiguredTasks dn="" xmlns="http://schemas.cordys.com/task/1.0/">
    <Task id="GUID of task" name="name of the task" type="type of task">
        <TaskParts>
            <TaskPart id="GUID of task part" isAvailable="true" name="name of task part"/>
            <TaskPart id="GUID of task part" isAvailable="false" name="name of task part"/>
        </TaskParts>
        <RelatedTasks>
            <Task id="GUID of task" isAvailable="true" name="name of the task"/>
            <Task id="GUID of task" isAvailable="true" name="name of the task"/>
        </RelatedTasks>
    </Task>
    <Task id="GUID of task" name="name of the task" type="type of task">
        <TaskParts>
            <TaskPart id="GUID of task part" isAvailable="false" name="name of task part"/>
            <TaskPart id="GUID of task part" isAvailable="true" name="name of task part"/>
        </TaskParts>
        <RelatedTasks>
            <Task id="GUID of task" isAvailable="false" name="name of the task"/>
            <Task id="GUID of task" isAvailable="false" name="name of the task"/>
        </RelatedTasks>
    </Task>
</ConfiguredTasks>
```

The table below gives the parameters of the configured task definition. The example considered here is Application tasks. This will help you understand the context of each parameter. The Application task is used to perform the following activities:

- Shows Application details
- Shows installation details
- Shows content of Application
- Installing Applications
- Uninstalling Applications
- Upgrading Applications

Parameters of configured task definition

Field	Description	Example
Configured Task	The role or user to which tasks have to be configured dn - The DN of the role or user	cn=systemAdmin,cn=organizational roles,o=system,\$ldaproot
Task	<ul style="list-style-type: none"> ■ ID - The unique identifier of the task. ■ Name - Name of the task. ■ Type - The type of task. 	<ul style="list-style-type: none"> ■ <GUID> ■ Application Management ■ UI task
TaskPart	<ul style="list-style-type: none"> ■ ID - The unique identifier of the task. ■ Name - Name of the task. ■ isAvailable - Specifies if the task is available for the user. 	<ul style="list-style-type: none"> ■ Show Application Package details ■ true - TaskPart is available to user ■ false - TaskPart is not available to user
Related Task	<ul style="list-style-type: none"> ■ ID - The unique identifier of the task. ■ Name - Name of the task. ■ isAvailable - Specifies if the task is available for the user. 	<ul style="list-style-type: none"> ■ Load Application Packages ■ true - TaskPart is available to user ■ false - TaskPart is not available to User

See [Example for Task](#).

Example for Task

The Application Registry Task is used to perform the following activities:

- Shows Application details
- Shows installation details
- Shows content of Application
- Loading Applications

- Unloading Applications
- Upgrading Applications

The activities are classified into task parts and related tasks as follows. The Application Registry task is configured to different roles allowing specific activities based on the role. The below matrix also indicates various roles and their privileges on performing an activity.

Activities	Task type	Can a System Administrator perform this activity?	Can an Organizational Administrator perform this activity?	Can a developer perform this activity?
Loading Applications	Related Task	Yes	No	No
Shows Application details	Task Part	Yes	Yes	No
Shows content of Application	Task Part	Yes	Yes	No
Shows installation details	Task Part	Yes	Yes	Yes
Unloading Applications	Related Task	Yes	Yes	No
Upgrading Applications	Related Task	Yes	Yes	No

Overview of task status

You can view the status of various tasks as well as a consolidated view of the tasks taken up by different members of the team if you are either a Work List Manager, a Team Lead, or a user with any Role using the Overview tab of the Inbox.

You can view statistics that reflect the workloads on the teams and users and facilitates the manager or lead to assess the progress of the tasks in their teams. Tasks can be delegated, if required, based on the information.

- In the **My Inbox** application, click **Overview** tab.

Depending upon the role you have, the corresponding folders such as work lists, teams, or roles, are displayed on the pane next to the tabs.

Role	View
Work List Manager	You can view the status of the tasks in the work list in a graphical format, and the team level breakdown as well as the task status of the members of the corresponding teams. The Work List Overview and Team

Role	View
	<p>Overview sections are displayed on the right pane.</p> <ul style="list-style-type: none"> ▪ The Work List Overview section provides a graphical representation of various states of the tasks against their count. This gives the manager a consolidated view on the progress of tasks in that worklist. ▪ The Team Overview section provides a user-level breakdown of various task status pertaining to a team. It allows the manager to identify the workloads of users of a particular team. This helps in re-allocation of the tasks accordingly, if required.
Team Lead	You can view the status of tasks in the team in a graphical format and a breakdown of the task status of the members in that team.
<Role>	You can view the status of the tasks assigned to that particular role and a breakdown of the task status of the users associated to that role.

Chapter 12

Modeling master data

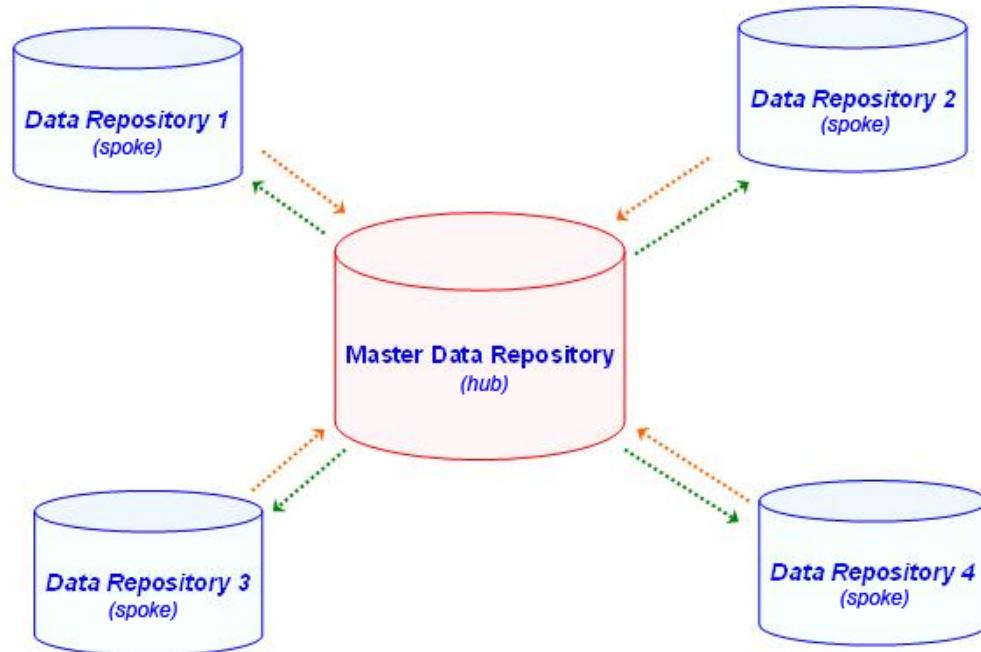
The data of an organization can be effectively used only when the information is properly organized in multiple locations across the organization. Master Data Management (MDM) synchronizes the data between multiple storage locations or data stores.

In AppWorks Platform, hub data store acts as the central source of accurate and consistent data (also called master data). Other data stores that must be synchronized should be connected with the hub data store.

Hub-and-spoke model

AppWorks Platform uses the hub-and-spoke model in which hub acts as the central master data store that contains the master data. The other data stores that must be synchronized connect to the hub data store and form the spokes.

A spoke can interact only with the hub and the association between the spoke and the hub is called subscription. The model is graphically represented in the following image.



When a change is detected in the spokes, the changed data is sent to the hub data store which in turn sends the change to other subscribed data stores .

Thus, MDM provides a mechanism to synchronize the data changes across multiple stores in a cost-effective and reliable manner. The synchronization process can be customized depending upon the needs of the organization.

Data integration in AppWorks Platform MDM can be broadly classified as follows:

- Designing an MDM model
- Monitoring and Managing the MDM model
- Administrative tasks to ensure smooth data integration

Note: Before designing the MDM model, ensure that you create the data stores and the appropriate tables.

For additional information, refer to the following topics:

- For information on creating repositories and tables, see [Preparing Master Data Repository](#).
- For information on designing an MDM model, see [Creating the Components of an MDM Model](#).
- For information on MDM Administrative tasks, see MDM Admin Settings in the AppWorks Platform Administrator's Guide.
- For information on monitoring MDM models, see Managing and Monitoring MDM in the AppWorks Platform Administrator's Guide.

Preparing master data repository

While preparing Master Data Repository (MDM), the first task is to identify the data stores in the applications that need to be kept consistent through regular data synchronization. For example, Organization A and Organization B are merged and you want to integrate supplier information of both the organizations. Each data source that you identify for Organization A and Organization B will form a spoke in the hub-and-spoke MDM model of AppWorks Platform.

After identifying the data sources, the next step is to identify the entities from each data sources for integration. Here, the term entity refers to a representation of relevant information about a business object, expressed in the form of an XML Schema. For a database, the entity is a database table - this can be expressed as an XML definition. For an external Web service, the schema of the data that the service provides becomes the entity definition.

In the example, the supplier data in both the organizations is described in terms of a database entity with appropriate attribute names and data types. The Supplier database entity is expressed in the form of an XML-based schema description.

MDM stores the related content in the MDM repository tables which are named as RCOR (Relational Common Object Repository) tables. If the tables are shared by all the spokes and hub, then the model is called as central repository. If the RCOR tables are created for both hub and spokes separately, then the model is a distributed one. For more information on configuring distributed and central repositories, see Configuring an MDM Repository in the AppWorks Platform Administrator's Guide.

Similar to the RCOR tables there are RCOR fields which contain MDM related information. These RCOR fields must be added to each data entity (table in the hub database) in the master data repository (hub).

To prepare the master data repository:

1. Determine the schema for the data entities that are part of the master data repository. If you are using an existing data repository as the master data repository, modify the data entities in the data repository to match this schema. Else, create a new data repository in DB2, Oracle, MS SQL, or PostgreSQL data repository server.
2. If you are using an existing data repository as the master data repository, add the RCOR fields to each data entity in the repository. See MDM Specific Fields in the *AppWorks Platform Administrator's Guide* for details. You do not need to add the RCOR fields to the hub data entities if you are using the State Models.
3. Run the database scripts in the master data repository to create some required procedures.

The database scripts are available in the `<AppWorks Platform_installdir>/<Instance name>/components/mdm/dbscripts/application` folder of the server where AppWorks Platform is installed.

Execute the following files depending on your database:

If the database is	Then run the script
DB2	MDM_DB2.sql
MS SQL	MDM_SQLSERVER.sql
Oracle	MDM_ORACLE.sql
PostgreSQL	MDM_POSTGRES.sql

For information on running the scripts, refer to the database vendor documentation. The application data stores are now ready for integration.

To create the MDM repository tables:

1. While installing MDM package (this is installed with AppWorks Platform), MDM Repository Tables are created with the specified database configuration. If it is required to create MDM Repository Tables separately then you can run the appropriate scripts from the following table.

The scripts are available in the folder <AppWorks_Platform_installdir>/<Instance name>/components/mdm/dbscripts/repository of the computer where AppWorks Platform is installed.

For information on running the scripts, refer to the database vendor documentation.

If the database is	Then run the scripts
DB2	<ul style="list-style-type: none">■ MDM_DB2_TABLES.sql■ MDM_DB2_TRIGGERES.sql■ 3. MDM_INIT
MS SQL	<ul style="list-style-type: none">■ MDM_SQLSERVER_TABLES.sql■ MDM_SQLSERVER_TRIGGERES.sql■ MDM_INIT
Oracle	<ul style="list-style-type: none">■ MDM_ORACLE_TABLES.sql■ MDM_ORACLE_TRIGGERES.sql■ MDM_INIT
PostgreSQL	<ul style="list-style-type: none">■ MDM_POSTGRES_TABLES.sql■ MDM_POSTGRES_TRIGGERES.sql■ MDM_INIT

2. Copy the CAP_SCHEMA_REVISION table along with the data from Cordys System database to the MDM Repository database. For information on copying the table, refer to the database vendor documentation.

Creating an MDM model

The MDM model acts as the framework for integrating organization data. The primary step in designing an MDM model is to identify the data stores for the spokes and hub in the model. Any change related to the data in the spoke is communicated to the hub. If required, the hub can propagate the change to the remaining spokes in the model.

An MDM model can be designed in the following ways:

- **Empty model:** The architecture of this model can be built manually by inserting the data stores.
- **Simple Synchronization Model:** This model can be built automatically by specifying the number of spoke data stores to be connected to the hub.

To create an MDM model:

1. [Create a Workspace](#) and click  (MDM Model).
The Choose a template before you begin dialog box opens.

2. To create an Empty Model:
 - a. Select **Empty Model** and click **OK**.
The <Untitled MDM Model> window opens.
 - b. Drag the hub and all the required spoke data stores from the Workspace explorer windows to the modeling area.
The Select entities for MDM <Hub or Spoke> Data Store window opens.

Tip: To accelerate the modeling process and quickly draw constructs, see [Easy Ways to Draw Constructs in Design Time](#).

- To quickly add a data store to draw an MDM model, click the Insert tab and drag the desired document to the MDM modeler canvas. Alternatively, click the Workspace tab in the modeler and drag the required document to the MDM modeler. To create a new data store you can directly drag Hub Data store or Spoke Data store from the Toolbox to the canvas and drag the data entities on to the Data stores.
 - To quickly associate a data entity or a subscription with another document, such as a Schema Fragment, MDM State Model, MDM Sequence Generator, Data Quality Plan, MDM Subscription Handler, Data Transformation, or a Rule, click the Insert tab and select the desired document. Alternatively, click the Workspace Explorer tab in the modeler and drag the required document onto a construct in the MDM Model.
 - c. Select the required data entities from the list and click **Add**.
The data store appears on the modeling area.
 - d. Select the hub or spoke entity on the modeling area and click **Add connector** from the toolbar.
 - e. To draw subscriptions between the hub and spoke entities, click the hub or spoke entity. Alternatively, click the source entity, press **CTRL** and click the destination entity.
3. To create a Simple Synchronization Model:
 - a. Select **Simple Synchronization Model**.
 - b. In **Number of Spoke Data Store**, enter the number of spoke data stores that need to be subscribed to the hub data store.
 - c. Click **OK**.
An MDM model is created with the specified number of spokes connected to the hub.

To define the properties of the model:

1. Double-click or right-click the Spoke or Hub data store and select **properties**.
The Property pane is displayed.
2. Enter the Data Store Properties such as the name, description of the data store.
3. Enter the Data Entities Properties for each data entity in the Property pane.
4. Double-click each subscription link in the model.
The subscription properties pane is displayed.

5. Enter the subscription properties for each link.

For more information on the fields in the subscription properties interface, see [Subscription Properties Interface](#).

6. Click **Save**.

The MDM model is created and ready for data integration.

After you complete this task:

- [Validate](#) the MDM model.

Whenever you update a record in a database table, MDM sends the updated record with all the columns instead of sending only the columns that were updated. If you are updating different columns of a particular record from two or more sources, the previous content is overwritten by the latest content thus leading to data loss and data inconsistency as the entire record is updated. Therefore, to ensure that only the updated/modified columns are propagated during sniffing, before publishing the model, select the **Allow Modified Fields Only** option in the MDM Admin settings page.

- [Publish](#) the MDM model to the organization before you start data integration.

Subscription properties interface

The following table describes the fields in the Subscription Properties interface.

Hub Entity	The name of the hub entity in the subscription
Spoke Entity	The name of the spoke entity in the subscription
Subscription Direction	<p>The direction in which the data flows</p> <ul style="list-style-type: none">■ Both Sides - Specifies that the flow of data is in both directions. When this option is selected, the Spoke to Hub and Hub to Spoke tabs appear.■ Spoke To Hub - Specifies that the flow of data is from spoke to hub. When this option is selected, the Spoke to Hub tab appears.■ Hub To Spoke - Specifies that the flow of data is from hub to spoke. When this option is selected, the Hub to Spoke tab appears.
Use Data Transformations	Selecting this option enables data transformation prior to data integration.

The following fields appear on the **Spoke to Hub** or **Hub to Spoke** tabs depending on the option that you choose for the **Subscription Direction** field.

Name	Specify a name for the subscription
Lookup Fields button	Lookup Fields is enabled on the Spoke to Hub tab only if you have

	<p>selected Enable Business Object Lifecycle while Creating the Hub entity.</p> <ol style="list-style-type: none"> 1. Click Lookup Fields. The Select Lookup fields dialog box opens. 2. Select the fields that uniquely identify the business object. 3. Click Save. <p>Lookup Fields is enabled on the Spoke to Hub tab only if the Hub entity has State Engine enabled. Do not include any auto-incremental field. Auto incremental fields are those fields whose value increases automatically for every new entry that is added. For example, if the employee ID of the last person recruited by a company is EMP598, and if the employee ID of the next new recruit will automatically be EMP599, then employee ID is an auto-incremental field.</p>
Select a Data Transformation	<ul style="list-style-type: none"> ■ Select a data transformation using  (Select a Data Transformation). ■ Select a data transformation when you have to map the data fields of an entity with the fields in the hub or vice versa. While using transformations, you can also apply some functions such as mathematical or string functions to modify the data before it is integrated. ■ Click  (Clear) to clear a data transformation. <p>Use the <code>IfExists</code> instruction provided by the Data Transformation Modeler while defining data transformations for MDM. This instruction creates a target element only if the source element is present. If this instruction is not used then empty elements will be created(in the target object)for those elements which are missing in source object. This could result in some issues while updating the database.</p>
Select a Rule	<ul style="list-style-type: none"> ■ Select a rule using  (Select a Rule). You need rules in situations where data must be integrated only if certain conditions are met. Rules are applied when you want to filter out certain records. For example, you may want to integrate data of all employees belonging to the accounts department. For information on rules, see Rules. ■ Click  (Clear) to clear a rule.
Select a Subscription Handler	<ul style="list-style-type: none"> ■ Select a Subscription Handler using  (Select a Subscription Handler). Subscription handlers are any custom logic that you want to apply on the data. ■ Create a Subscription Handlers or select a Subscription Handler

	<p>if you have already created.</p> <ul style="list-style-type: none">■ Click  (Clear) to clear a subscription handler.
Use Scheduler	<ol style="list-style-type: none">1. Select Use Scheduler to specify a schedule for integrating data at a specific time. The Configure button is enabled.2. Click Configure to specify the time at which data integration must be triggered. The Subscription Schedule dialog box opens.
Events Subscribed	<p>Specify the events for which you want to integrate the data. Select one of the following options:</p> <ul style="list-style-type: none">■ Update: To propagate the updates■ Update And Insert: To propagate the updates, and insertions■ Update, Insert And Delete: To propagate the updates, insertions, and deletions

Creating the components of an MDM model

The MDM model is a base model containing a set of spokes and a hub. The hub and spokes are data stores and in turn they contain data entities. A hub data entity can contain a state model. A state model contains states, transitions and events.

Details of the following procedures are cross-referenced with hyperlinks.

To create the components of an MDM model:

1. Create the required hub and spoke data stores.
Every model needs only one hub and can contain two or more data stores that are assigned as spokes. All the spokes in the model will be synchronized with the hub.
For information on creating a data store, see [Creating a Data Store](#).
2. Create the data entities.
Every data store contains a set of data entities.
For information on creating data entities for each data store, see [Creating a Data Entity](#).
3. [Create the MDM model](#).

The components of the MDM model are ready.

You can [validate](#) the MDM hub data store, spoke data store, hub data entity, spoke data entity, sniffer, model, and state model after you create it.

Creating a data store

Before you begin:

- If you are creating a hub data store, Create an MDM Publisher and ensure that it points to the application database.
- Also ensure that the Hub Publisher option is selected while creating the service group.

A hub data store is central to data synchronization. All spoke data stores in the master data management system are synchronized with the hub.

To create a data source:

1. **Select a starting point** and click  (MDM Hub Data Store) to create the MDM hub data store, or click  (MDM Spoke Data Store) to create the MDM spoke data store. The <Untitled MDM (Hub or Spoke) Data Store> - MDM (Hub or Spoke) Data Store window opens.
2. **Provide the details of the Data Store.**
3. **Click Save.**

An MDM data store of type hub or spoke is created and added to the project content tree.

You can [validate](#) the created data store.

After you complete this task:

- You must add a hub data store to the MDM model along with the spoke data stores.

Configuring an MDM sniffer

MDM Sniffer tracks the changes in a data store. AppWorks Platform provides four sniffers. You can also create custom sniffers.

Before you use custom sniffers:

- Create a Java class for the sniffer that implements Interface sniffer, provided by MDM framework. The custom sniffer class is Java code that sniffs for changes in the data store.

To configure an MDM sniffer:

1. **Select a starting point** and click  (MDM Sniffer). The <MDM Sniffer> - MDM Sniffer window opens.
2. In **Implementation Class**, specify the path of the class name.
3. **Click Save.** The <MDM Sniffer> - MDM Sniffer dialog box opens.
4. Type the appropriate name and description for the MDM Sniffer.
5. To save the MDM sniffer at a specific location, click  and select a folder in the Select

Folder dialog box. If required, create a project, right-click it and create a folder.

6. Click **OK**.

An MDM sniffer is created and added to the project content tree.

You can [validate](#) the created MDM sniffer.

After you complete this task:

- You can add the sniffer to the required spoke data entity.

Creating a data entity

Before you begin:

- [Creating a data store](#).
- Create metadata for each data entity.
- [Configuring an MDM sniffer](#).

Create data entities to model the objects that you want to synchronize. The specification of the model involves specifying the structure of data, a sniffer to track the changes and Web services to perform operations on the data.

To create a data entity:

1. Right-click the appropriate data store and select the <Hub or Spoke> Entity option. The Untitled MDM <Hub or Spoke> Entity - MDM <Hub or Spoke> Entity window opens. See MDM Data Entity Interface in the AppWorks Platform Administrator's Guide for more details on the fields.
For hub data entity, the **Enable Business Object Lifecycle** option is enabled by default. Use this option should to define states, events, and transitions for a business object, using state model. You must clear the check box for performing the simple synchronization task; the Sniffer tab appears in place of the Business Object Lifecycle tab.
2. Click the **Web Services** tab to specify the Web services to perform operations on the data.
The Web services that you specify perform necessary operations on the data, at run-time.
3. Click the **Sniffer** or **Fields on the Business Object Lifecycle** tab per the requirement.
4. Click the **Advanced** tab to view the default batch size for the Web services to update, insert, or delete data, and the name of the sniffer audit table.
5. Click **Save**.

A data entity is created and added to the project content tree. You can [validate](#) the created data entity.

Creating an MDM subscription handler

MDM Subscription handler tracks the changes in a data store. The custom implementation class is a Java class containing a program that handles the data before it reaches the hub data store.

Before you begin:

- Create a Java class for the subscription handler that implements `ISubscriptionHandler.java` Interface provided by MDM framework.

To create an MDM subscription handler:

1. Select a starting point and click (MDM Subscription Handler) to create a subscription handler.
The Untitled <MDM Subscription Handler> - MDM Subscription Handler window opens.
2. In **Implementation Class**, specify the path of the class name.
3. Click **Save**.
The <MDM Subscription Handler> - MDM Subscription Handler dialog box opens.
4. Type the appropriate **Name** and **Description** for the MDM Subscription Handler.
5. To save the MDM Subscription Handler at a specific location, click . The Select Folder dialog box opens. Select a folder. If required, create a project, right-click it and create a folder.
6. Click **OK**.

An MDM Subscription Handler is created and added to the project content tree.

After you complete this task:

The Subscription Handler can be added to any subscription between the hub and spoke entities in the MDM Model.

Designing a state model

A state model is a set of states, events, and transitions. Transition is the transformation of a business object from one state to another. Events trigger the transition of a business object from one state to the next state.

To design a state model:

1. [Select a starting point](#) and click (MDM State Model) to create an MDM State Model.
The <Untitled MDM State Model> - MDM State Model window opens.
2. Click on the **States** tab to create states.
See [Creating a State](#), for the procedure to create states.
3. Click the **Transitions** tab to define transitions between states, and click on the tool bar.
The Transition - Enter Transition Details window opens.

4. In the **From State** list, specify the state from which the transition starts.
5. In the **To State** list, specify the state where the transition ends.
6. In Event Name, specify a name for event that triggers the transition.
7. If you want to specify a guard on the transition, select **Use Guard**.
Else, skip this step and go to the next step.
The Guard window opens.
 - a. In **Guard Name**, specify a name for the guard.
 - b. In the **Guard Type** list, select the type of guard.
Only type Rule is available. You can [Create Custom Guards](#), if required.
 - c. In **Rule Expression**, type the rule expression.
To view an example for a rule expression, see Example for Configuring a Rule Expression in MDM in the *AppWorks Platform API Guide*.
8. To show this event in the instance view of MDM Cockpit, select **Show Event in Cockpit**.
9. Click **OK**.
The transition is created.
10. Repeat the steps until all the required transitions are created.
11. Click **Save**.
The <Untitled MDM State Model> - MDM State Model dialog box opens.
12. Type the appropriate **Name** and **Description** for the MDM State Model.
13. To save the MDM State Model at a specific location, click and select a folder. If required, create a project, right-click it, and create a folder.
14. Click **Save**.

The state model is created.

You can [validate](#) the created state model.

To edit the attributes of a transition:

- Select a transition and click .

To delete a transition:

- Select the required transition and click .

To view the SCXML code of a state model:

- Click the **View SCXML** icon.

After you complete this task:

- Associate the state model with an MDM hub data entity in an MDM model.

Creating a state

A state model is a sequence of states.

To create a state:

1. Select a starting point and click  (MDM State Model) to create an MDM State Model. Alternatively, you can also open an existing state model and add states to it. The <MDM State Model> - MDM State Model window opens.
2. Click the **States** tab.
3. Click  above the State Name and Description table. The State - Enter State Details window opens.
4. Specify the **State** name and **State Description**.
5. To specify the initial state, select **Initial State**.
6. To specify the final state, select **Final State**.
 - If both options are selected, the state forms a continuous loop in the state model. This is useful in scenarios when the state model must continuously process business objects.
 - To edit the attributes of a state, select a state and click .
 - To delete the state, select a state and click .
7. Click **OK**.
8. To view the instances of this state from the MDM Cockpit, select **Show in cockpit**.
9. In the **Entry** and **Exit** tabs, specify the required action(s) that are to be performed when the business object enters or exits the state.
10. To add an action, click .
For information on the default actions that can be added, see [Types of Actions](#). You can [Create a Custom Action Type](#), if required.
11. To delete an action, select the action and click .
12. Click **Save**.
The state is saved in the state model. If the state model is already saved then the state is available for use in the state model.
13. If the state model is not saved, type the **Name** and **Description** for the MDM state model in the <Untitled MDM State Model> - MDM State Model dialog box.
14. To save the state model at a specific location, click  and select a folder. If required, create a project, right-click it and create a folder.

The required state is created.

After you complete this task:

Associate states with a state model.

[Creating a custom action type](#)

AppWorks Platform provides some standard action types for a State model. You can also create action types that match the requirements of your organization.

To create a custom action type:

1. Click  (Define Custom Action type) on the toolbar.
The Custom Action Types window opens.
2. Click  to add an action type.
A blank properties list appears for the new action type.
3. Type the **Name** of the action type.
4. Type the **Class** for the action type.
5. The path of the default property sheet assigned to this new action type is displayed under Property Sheet. Modify the file according to your requirements.
6. To add parameters for the action type, click .
7. To remove parameters, select the parameter in the list and click .
8. Click **Save**.

A new action type is created.

To edit an existing action type:

1. In the list of action types, select the action type.
The details appear below.
2. Make the necessary changes and click **Save**.

Types of actions

Consider an insurance claim scenario to understand the fields below:

Business Process	Parameters	Description of parameters
Select this option to invoke a business process model. A business process is modeled based on the sequence of activities in the insurance claim. This business process model can be triggered in the initial state when the request for the insurance claim is received. Before selecting this option, ensure that you draw a business process model and publish it. The namespace of the business process model must be of the format http://schemas.cordys.com/rnor/1.0	Process Name User	Select a published business process model. Select the user under whose context the business process model is invoked.

Mail	Parameters	Description of parameters
Select this option to send an e-mail. In the	To	Specify the recipient of

Mail	Parameters	Description of parameters
insurance claims example, you can use this option for any mail communication external to AppWorks Platform environment. You may have to send an e-mail to the customer to intimate the status of the insurance claim.	From Subject Message	the e-mail. Specify the sender of the e-mail. Specify the subject of the e-mail. Specify the message contents of the e-mail.

Notification	Parameters	Description of parameters
Select this option to send a notification. Use this option for any communication that is internal to AppWorks Platform. In the insurance claim example, send a notification from the business process model to inform the responsible person to verify the claims.	Send To Subject	Specify the recipient of the notification. Specify the subject of the notification.

Update to Hub Entity	Parameters	Description of parameters
Select this option to update the hub data entity.	Create Registry Link	Select this option to create a registry link between spoke and hub entity records.

Registry Services	Parameters	Description of parameters
Select this option to add the object to the list of identifiers.	Create Registry Link Remove Registry Link	Select this option to link or add the identifier of a data entity to the registry. Select this option to delink or remove the identifier of a data entity to the registry.

Synchronize Spokes	Parameters	Description of parameters
Select this option to synchronize the spokes	Synchronize Back To Source	Select this option, if you want to update the source with information that was not included when the source originally sent the data.

Synchronize Spokes	Parameters	Description of parameters
	Synchronize Shared Data	Select this option, if a record is already present in the hub, spokes and the keys in the registry, and you want to update the existing records. This option is not applicable for insert operation.
	Create Registry Links	Select this option, if you want to create a registry link between hub and spoke entity records.

Web Service	Parameters	Description of parameters
Select this option to use a Web service	Web service Request Transformation Response Transformation	Select an existing Web service or model a new Web service. Select an existing data transformation or model a new data transformation. The source schema of the selected data transformation schema must be the schema of the business object used in the hub entity. The target schema must be the selected Web service request with the root element being the method name. If such an XML schema does not exist, it must be created. Select an existing data transformation or model a new data transformation. The source schema of the selected data transformation must be the response schema of the selected web service with the root element being the response name of the method. If such an XML schema does not exist, it must be created. The target schema must be the schema of the business object used in the hub entity.

Transform	Parameters	Description of parameters
Select this option to map the attributes of a business object to another	Data Transformation	Select an existing data transformation or model a new data transformation. For information on modeling a new data transformation, refer to Modeling Data Transformation.

Match	Parameters	Description of parameters
Select this option to use the match data quality operation	Event	Specify the event name, present in the State Model that you want to fire, when the matches are present.

Cleanse	Parameters	Description of parameters
Select this option to use the cleanse data quality operation Select this action only when the data quality is enabled.	(Name, Value) pair	Specify the name and value parameters. These parameters are accessed at run-time when the State Model processes this action.

SignalStateModel	Parameters	Description of parameters
Select this option to fire an event on a different State Model	(Name, Value) pair	Specify the name and value parameters. These parameters are accessed at run-time when the State Model processes this action.

Send Event	Parameters	Description of parameters
Select this option to trigger an event to continue with the state model execution.	Event Name	Specify the name of the event. This event name should be mapped to any of the event names of the outgoing transitions defined on that state.

Synchronize Source Spoke	Parameters	Description of parameters
Select this option to synchronize the data with the source spoke.	-	-

Business process model input for state object model

```
<tuple>
<new>
<businessobject>
<BUSINESS_OBJECT>
<COL1>val1</COL1>
<COL2>val2</COL2>
<COL3>val3</COL3>
```

```

</BUSINESS_OBJECT>
</businessobject>
<mdmmetadata>
<userdn>PARAMETER</userdn>
<dbevent>PARAMETER</dbevent>
<hub>
<keys>PARAMETER</keys>
<backend>PARAMETER</backend>
<entity>PARAMETER</entity>
</hub>
<spoke>
<keys>PARAMETER</keys>
<backend>PARAMETER</backend>
<entity>PARAMETER</entity>
</spoke>
</mdmmetadata>
</new>
</tuple>

```

Creating a transition

A transition is a transformation that moves the object from one state to the next state in a state model.

To create a transition:

1. Select a starting point and click  (<MDM State Model>) to open an existing MDM state model.
2. If you have the MDM state model already opened, then perform Step 3.
3. Click the **Transitions** tab.
4. Click  above the Source and Target table.
The Transition - Enter Transition Details window opens.
5. In the **From State** list, select the state from which the transition starts.
6. In the **To State** list, select the state where the transition ends.
7. In **Event Name**, specify a name for event that triggers the transition.
8. To ensure that certain conditions must be met before the object moves to the next state, select **Use Guard** to specify a guard on the transition.
The Guard section appears.
9. Specify a **Guard Name**.
10. Select a guard from Guard Type list. Select the default guard type, Rule, or Create Custom Guards.
11. Type the Rule Expression if you have selected the default guard type Rule. To view an example for a rule expression, refer to Example for Configuring a Rule Expression in MDM.
12. Click **OK**.

The required transition is created.

To edit the attributes of a transition:

- Select a transition and click .

To delete the transition:

- Select a transition and click .

Creating a custom guard type

MDM provides some standard guard types for a State model. You can also create guard types that match the requirements of your organization.

To create a custom guard type:

1. Click  (Define Custom Guard Type) on the toolbar.
The Custom Guard Types window opens.
2. Click  to add a guard type.
A blank properties list appears for the new guard type.
3. Type the **Name** of the guard type.
4. Type the **Class** for the guard type.
5. The path of the default property sheet assigned to this new guard type is displayed under **Property Sheet**. You can modify the file according to your requirements.
6. Click  to add parameters for the guard.
7. To remove any parameter, select the parameter in the list and click .
8. Click **Save**.

A new guard type is created.

To edit an existing guard type:

1. Click the guard type in the list of guard types and its details appear below.
2. Make the necessary changes and click **Save**.

Creating a sequence generator

MDM Sequence Generator creates a unique sequence identifier for the data entity. The Sequence Generator class is a Java code that extends the Interface `com.cordys.datagovernance.registry.idgenerators.IRegistryIDGenerator` and generates the next unique identifier for the Registry table.

Before you begin:

- Create a Java class for the Sequence Generator.

To create a sequence generator:

1. Select a starting point and click  (MDM Sequence Generator).
The <MDM Sequence Generator> - MDM Sequence Generator window opens.
2. In **Implementation Class**, specify the path of the class name.
3. Click **Save**.
The <MDM Sequence Generator> - MDM Sequence Generator dialog box opens.
4. Type **Name** and **Description** for the MDM Sequence Generator.
5. *Optional.* To save the MDM Sequence Generator at a specific location, click  and select a folder in the Select Folder dialog box. If required, create a project, right-click and create a folder.
6. Click **OK**.

An MDM Sequence Generator is created. You can view it in the Workspace Documents window.

After you complete this task:

- Add the Sequence Generator to the required hub data entity or spoke data entity.

Creating a data quality plan

Create a data quality plan to integrate third-party data quality management tools with MDM.

To create a data quality plan:

1. Select a starting point and click  (Data Quality Plan).
The <Untitled Data Quality Plan> - Data Quality Plan window opens.
2. Type the **Name** and **Description** of the data quality plan.
3. Click  corresponding to Business Object.
The Select Business Object window opens.
4. Select a relevant schema fragment of the business object that you can use while creating the hub entity.
5. Click **OK**.
The selected schema fragment appears in the Business Object box.
6. *Optional.* Click **Clear** to delete the schema fragment.
7. Click **Save**.
The data quality plan is created.

After you complete this task:

- You can right-click the newly-created Data Quality Plan to create a Data Quality Service to configure the services of third-party data quality management tool.
- You must associate the data quality plan with an MDM hub data entity to achieve data quality.

Creating a data quality service

A Data Quality Service contains information related to the property sheet and details of implementation class of the third party data quality tools.

To create a data quality service:

1. Open the context menu of Data Quality Plan and click  (DataQuality Service) .
The <Untitled Service> - Service window opens.
2. Select any one of the following Service types.

Custom	Select to provide a custom Service configuration. The Vendor Name, Version, Data Quality Service Identifier, Implementation Class, and Select Property Page fields appear.
Default	Select to use the default Service configuration that the DataQuality application provides.

3. Select **Human Inference** from the Select DataQuality Vendor list.
The Vendor Version, Data Quality Service, and Implementation Class fields auto-fill with the Human Inference default values.
4. For **Custom** type of a Service, specify the **Vendor Name, Version, Data Quality Service Identifier, Implementation Class**, and **Select Property Page** fields.
5. Click **Save**.

The Data Quality Service is created.

Using business identifiers

Business identifiers

In an application it is necessary to identify the process instances using specific identifiers. These types of identifiers are called Business Identifiers which can be defined at a project level and be linked to one or more process models. When a single business identifier is linked to multiple process models, then each instance of that model will have one instance of the linked business identifier value. Process developers should ensure that proper values are assigned to these Business Identifiers. Assignment of business data to Business Identifiers is possible using the message map. However, the values stored in the business identifiers cannot be used to assign values to the input messages of an activity. If values larger than 250 characters are assigned, they are truncated at run-time. Once the values are linked and assigned to these identifiers, they enable the query of process instances across different process models with a single business identifier.

Business identifiers must have a name, a description, and a type.

- The description of a business identifier is translatable. An Auto-suggest menu is available for the Description field, using which you can select from the existing text

identifiers.

- To translate a description, right-click the required business identifier, and select Translate. The Translation Editor is displayed, where you can specify the translation languages and the translated descriptions.

Business identifier data types

Business identifiers are type safe. Business Identifiers can have three data types:

- **String** - The value of the string identifier must not be more than 250 characters.
- **Numeric** - Precision must be specified.
- **Date** - The value must be in XSD format. If XSD format is not used, the process instance throws an error during the execution. For example, XSD Date format 2001-10-26T21:32:52.12679.

The following tasks can be performed using the Business Identifiers:

- [Creating a Business Identifier](#)
- [Associate business identifiers to a business process model](#)
- [Filter the process instances in Process Instance Manager at runtime using Business Identifiers](#)

Chapter 13

Creating a business process category

A business process category (also called business functions) contains all the business processes that belong to a functional group. Examples of business process categories include all production related processes, all sales related processes, all finance related processes and so on.

See [Business process category properties interface](#).

To create a business process category:

1. Select a starting point and select  (Business Process Category).
The Untitled Business Process Category-Business Process Category window opens.
2. Click the **Attached Processes** tab and click .
The Select Business Process dialog box opens displaying a list of existing business process models.
3. Select the required business process and click **OK**.
The selected business process model is attached to the current business process category.
4. Click the **Annotation** tab and type any additional notes or comments on the business process category.
5. Click **Save**.
The new business process category is added to the project content tree in Workspace Documents.

You have successfully created a business process category.

Business process category properties interface

The Business Process Category - Business Process Category Properties helps in viewing and setting the properties of a business process category at the time of modeling a business context.

The fields on the **General** tab are as follows.

Description	Provide a description for the business process category. This is a mandatory field.
-------------	---

Font	Select a font size from the list.
Color	Click to select required color.

The fields on the **Business Process Categories** tab are as follows.

Name	Displays name of the business process category.
Description	Displays description of the business process category.
Path	Displays the path where the selected business process category is located.

The fields on the **Business Processes** tab are as follows.

Name	Displays name of the business process model.
Description	Displays description of the business process model.
Path	Displays the path where the selected business process model is located.

The fields on the **Annotation** tab are as follows.

Annotation	Provide any additional notes or comments on the business process category.
------------	--

Part II

DEVELOPING APPLICATIONS

Chapter 14

Developing applications

After you design a Business Process Model that captures your Company's requirements, you are ready to start creating a Web application around it.

A Business Process Model captures the processes and business logic that your application must reflect. A Web application in turn comprises various artifacts (such as Web Services and XForms) bound in a logical sequence according to the Business Process Model. These artifacts cater to specific requirements of the application, for example, capturing user inputs, retrieving data, and performing backend operations according to a defined logic.

The topics in this section provide information on the functionality provided by AppWorks Platform to create and work with [WS-AppServer](#), [Web Services](#), [Web Applications](#), [Application Connectors](#), and [Tasks](#).

Chapter 15

Collaborative workspace - Team development setups

Setting up an application development environment where multiple developers work together to develop applications has its own needs and challenges. Through Collaborative Workspace (CWS), AppWorks Platform offers a unified working environment where applications can be built, tested, and packaged for deployment to production systems.

In AppWorks Platform, the Collaborative Workspace (CWS) is an Integrated Development Environment (IDE) where all stakeholders of a project, even if not co-located, can carry out tasks to create a deliverable and its supporting artifacts.

The core capabilities of CWS are:

- Centralized information management, enabling sharing of design-time content
- Web-based, shared project workspaces that work across geographical boundaries
- Single repository based on a unified metamodel
- Configuration control of shared artifacts
- Automatic dependency tracking among documents
- Single-click building and deployment of content to runtime

CWS accommodates all the requirements of team application development. It provides a single interface at design time to develop the application content. It facilitates easy creation and modification of the documents that form the building blocks of an application. All developers developing an application can access their own workspace through a browser on any computer, and can work simultaneously on the same application content.

The following scenarios are some examples where you can set up AppWorks Platform development teams in CWS to collaboratively develop an application:

- Separate developer stations that are configured to a Source Control Management (SCM) system
- Separate workspaces, where a central development server is configured to an SCM system
- Multiple development servers that are configured to an SCM system
- A single Workspace with a central development server

Separate developer stations that are configured to a Source Control Management (SCM) system

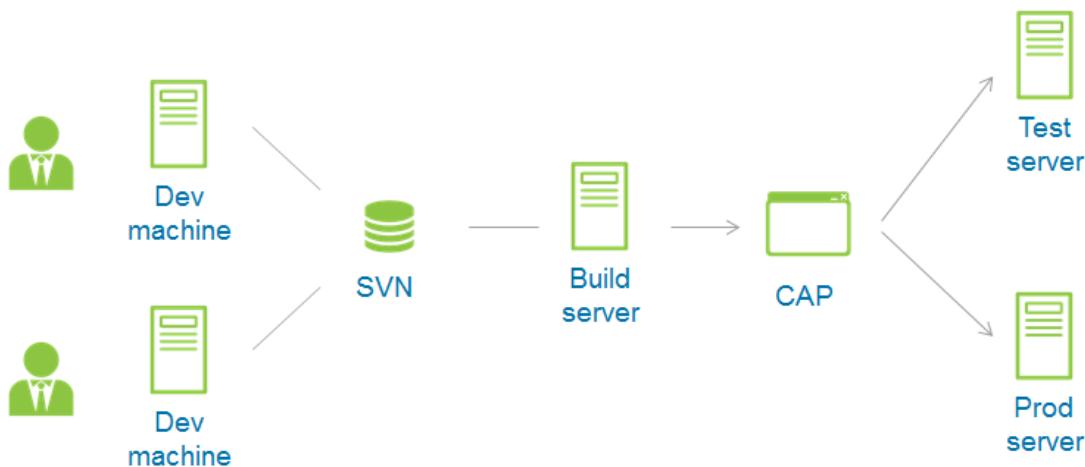
In this scenario, every developer has AppWorks Platform on their computers, and an individual workspace. The workspaces on the different computers are configured to a common SCM system. Developers can create and update artifacts on their computers, and synchronize their content with the SCM system.

Build or deployment, test, and production activities are done on separate servers connected to the same SCM system.

This scenario has the following features:

- One development server for each developer.
- Isolation of development content.
- Application content is shared with other developers through the SCM system.
- Versions of the developed content are maintained in the SCM system.
- If required, workspaces can be removed without affecting the content stored in the SCM system.

The following image illustrates this type of development scenario.



Separate workspaces, where a central development server is configured to an SCM system

In this scenario, every developer connects to a private workspace on the central or shared development server. The workspaces on this development server are linked to the same SCM repository. Developers can add and modify content within their respective workspaces.

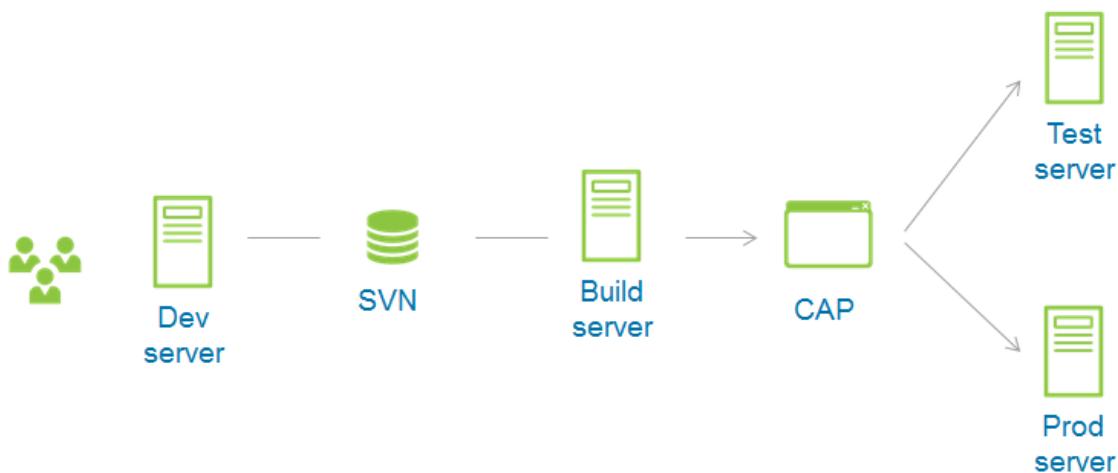
They can synchronize their content with the SCM system by using the option available in their workspace.

The build or deployment server receives all the updates from the SCM system. On demand or at regular intervals, the build master incorporates all changes to this workspace and packages all projects into application packages. These packages can be loaded on test or production servers for testing or production.

This scenario has the following salient features:

- Every development server supports multiple development workspaces.
- Development content is isolated.
- Application content is shared with other developers through the SCM system.
- Developers can belong to more than one workspace. This provision of shared workspaces facilitates functional grouping of developers within an organization, for example, all business users can work in one workspace.
- Versions of the developed content are maintained in the SCM system.
- If required, workspaces can be removed without affecting the content stored in the SCM system.

The following image illustrates this type of development scenario.



Multiple development servers that are configured to an SCM system

In the previous scenario, all developers worked on a central development server. It is also possible to set up multiple development servers, so that the developers are spread across these servers. The workspaces on the different development servers can still be connected to the same SCM system.

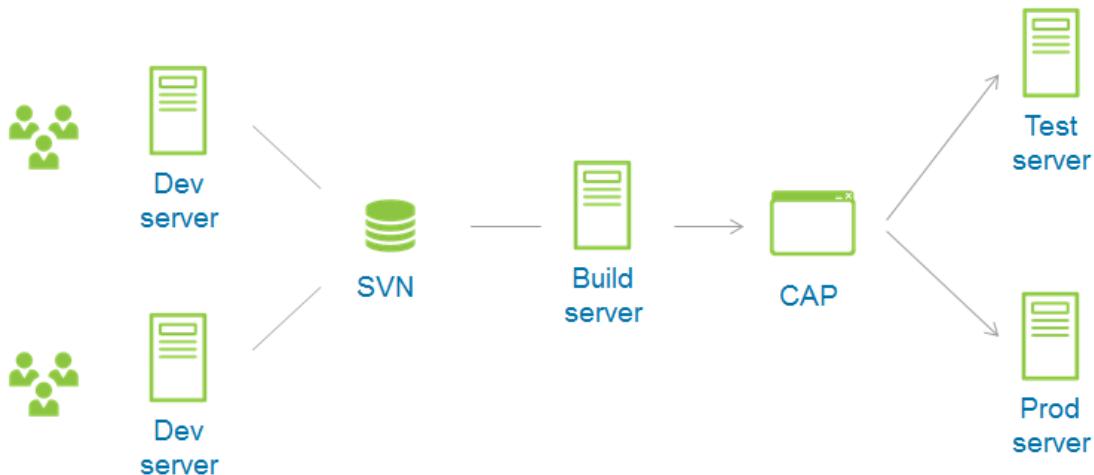
The build and deployment process is the same as in the previous scenario.

This scenario has the following salient features:

- Every development server supports multiple development workspaces.
- Developers can be spread across several development servers. Doing so helps in scaling up the number of developers working on the same project when reaching the limits of a single development server. You can also choose the geographical locations of the different development servers. Doing so enables you to minimize the network latency for remote development teams.
- Development content is isolated.
- Application content is shared with other developers through the SCM system.
- Versions of the developed content are maintained in the SCM system.
- If required, workspaces can be removed without affecting the content stored in the SCM system.

If multiple developers work on the same application, but through different development servers, these development servers must run the same version of the software.

The following image illustrates this type of development scenario.



A single Workspace with a central development server

In this scenario, for an application, all developers work on a central or shared development server and in the same workspace. All developers use only a single workspace. Collaboration can occur even without configuring the workspace to an SCM system. In this scenario, it is recommended to configure the workspace to an SCM system to ensure that the availability of the sources is not dependent on the existence of this single workspace. Using the SCM integration also ensures that the sources are version controlled.

Additionally, if all developers work in the same workspace, there is an increased chance of multiple developers working on the same models. If one developer decides to test changes, changes of other developers are likely to be published as well. This makes it very hard for individual developers to determine the effect of their isolated changes.

If the workspace is configured to an SCM system, application building and packaging can be done on a separate build server, and the package can then be delivered to separate test and production servers. If the workspace is not configured to an SCM system, the application must be built from that same workspace either manually or automatically. The delivery of the package to test and production servers is then similar to the other scenarios.

Setting up the CWS development environment

The following high-level topics contain information that help you set up the CWS development environment:

- [Creating a Workspace](#)
- [Creating a Solution](#) (optional)
- [Creating a Project](#)

Creating a workspace

A workspace forms the core of the AppWorks Platform development environment. It is the starting point for any application development, where all design-time documents are created and maintained. Any number of developers can connect to a workspace.

Note: It is recommended that the developers have their own workspace(s), instead of sharing workspaces among themselves.

A workspace stores all its content in the form of XML documents inside XML Document Store (XDS). It is an isolated development environment. Therefore, modifying its contents will not affect the content of other workspaces.

Important: Do not use long names to create projects, folders, and documents because very long names interfere with the synchronization process. On the file system, the synchronized file name takes into account the entire path, starting from the installation directory including the drive letter (for example C:). Since it is difficult to prescribe specific number of characters for a project name or document name, it is better to use as few characters as possible while providing names.

Before you begin:

- Be sure that you have the role of Developer to create a Workspace.

To create a workspace:

1. On the Welcome page, click  (Expand Menu) and from the list of applications, click  (Workspace Documents).
The Organize Workspaces window opens.

2. Click  (Add).
The Development Workspace wizard opens.
3. Provide the Name, Description, and Annotations and select one of the following Software Configuration Management type options:

No source control	If you do not want to associate the workspace with any Source Control Management (SCM) system. If you select this option, skip Step 4 of this procedure and go to Step 5.
SVN for team development	If you want to associate the workspace with an SCM such as SVN. Use discretion in setting up SVN for the team. This option is only a provision and not a recommendation. For more information, see Setting Up a Collaborative Workspace . If you select SVN for team development , an additional screen will be shown later in the wizard for defining SVN details, and the title of the current screen changes to New Workspace - Step 1 of 4.

4. Click **Next** and enter the following information on the SVN properties page:

URL	URL of the SVN repository to which you want to configure the workspace. In the URL, you also specify the protocol for network access of SVN. The four most common protocols that are used are: <ul style="list-style-type: none">■ http://repos■ https://repos■ svn://repos■ svn+ssh://repos CWS supports only the first two protocols (http and https).
User Name	To access the repository. As much as possible, provide the fully qualified username. If you are a user in a domain, provide the name in the <domain name>\<user name> format. If you are creating the workspace using a local system account, provide the name in the format <computer name>\<user name>. However, in specific cases, the SCM authentication manager might be able to handle user names without domain or computer names.
Password	To access the repository. Optionally click Test connection to test access to the repository.

5. If the SVN repository can only be accessed by using a proxy server, activate the option **Use a proxy server** and enter the following information:

- Host of the proxy server
- Port of the proxy server
- Username with which the proxy server can be accessed
- Password that belongs to the previous username

6. Click **Next** and enter the following information about the project in the New Workspace - Step 3 of 4 screen.

A project is a container for all documents that together form an application package. It provides a single design-time view of the development content and can be validated, published, and packaged. A project can be shared by multiple solutions within the same workspace.

- **Name of the project** to be added to the workspace
- **Package Owner** to specify the owner of the application
- **Name of the product** to specify the name of the application

7. Click **Next**.

The New Workspace - Step 4 of 4 screen is displayed. It displays a summary of the information provided in the earlier screens.

8. Click **Create**.

A message is displayed confirming the creation of your workspace.
You can click **Details** to view the workspace information in the Details tab.

9. Click **Close**.

The Workspace Documents opens with the default view set to Explorer. However, you can [change the view of Workspace Documents](#) to Tag Search or My Recent Documents views, depending upon your development needs.

After you complete this task:

- To begin application development, add [documents](#) to the project(s).
- To organize your workspace, you can add more [projects](#).
- To group projects logically, you can create new [solutions](#) or attach projects to an existing solution.

Working with workspace documents

Workspace Documents is a central interface especially designed for rapid application development, so that users are able to quickly create and modify documents, browse through solution/project structure, switch between workspaces and validate, publish, and package applications.

To access Workspace Documents:

1. On the Welcome page, click  (Extend Menu).
2. From the list of applications, click  (Workspace Documents).

Workspace Documents provides three different views through three tabs on the same window, the default being set to Explorer. Users can click the tabs to switch between these views as and when they need.

The usefulness of each view is described in the following table.

Tab/View	Description	Referred in Documentation as...
Explorer	Displays the structure of the workspace in a tree format. It shows how all the projects and the documents are arranged under the solution. This is the default view and the only view where you can switch between different solutions that exist in the workspace, using the option available on the toolbar.	Workspace Documents (Explorer)
My Recent Documents	Lists all the documents that were viewed in the workspace. The most recent document will be positioned at the top. This view displays only the documents and not their project or solution related information. It also does not show how the documents are organized in the solution.	Workspace Documents (My Recent Documents)
Tag Search	Lists all the documents based on the tags applied to a particular document. When you switch to this view, you can filter the documents using  (Lookup tab), based on the tag name.	Workspace Documents (Tag Search)

Procedural steps across the documentation have been written assuming that you are working in the Explorer view. It is because of the complete view of the Workspace Documents it provides in the form of a tree structure, besides offering a very familiar and convenient way of working. However, depending upon your preference you may choose to develop your application in any of the other two views.

Creating a solution

Important: Creating or using a solution is not required. This feature has been provided to enable users to logically group their projects that exist in the same workspace, if required. Hence, wherever a procedure on using a solution is mentioned in the documentation, the procedure has to be followed only if a solution was used in the application. Else, the solution part in the procedure can be ignored.

A solution is a "virtual" container for the projects in a workspace. A solution helps in functionally and logically grouping of projects that exist in the same workspace. A workspace may contain multiple solutions and each solution may contain several projects. Projects can be attached to multiple solutions. At any given point of time, you can only have one active solution.

To create a new solution:

1. On the Welcome page, click  (Extend Menu).
2. From the list of applications, click  (Workspace Documents).
You may be prompted to select a Workspace if you have not selected one already.
The Workspace Documents window opens.
3. On the toolbar, click  (Switch Solution) and select Manage Solutions.
The Solution window opens.
4. Click  (Insert).
The Create Solution wizard opens.
5. Enter **Name**, **Description**, and **Annotations** for the solution in the Create Solution - Step 1 of 2 screen.
6. Click **Next**.
The Create Solution - Step 2 of 2 screen opens.
7. Do one of the following:

Create a solution without any project Create new project(s) and add them to the solution Select from the existing projects	<ul style="list-style-type: none"> ■ Click Finish. <ol style="list-style-type: none"> 1. Click Next. The next page of the wizard opens. 2. Click  (Project) to create a project. The Project wizard opens. 3. On the Project wizard, enter Name, Description and Annotations for the project and click Finish. The project is created and is positioned within the space on the Create Solution wizard. 4. Repeat Steps 2 and 3 to create more projects and add all of them to the solution 4. Click Finish. <ol style="list-style-type: none"> 1. Click Next. The next page of the wizard opens. 2. Click  (Open). The available projects appear in a list. 3. Select the project. It is added to the solution and displayed in the space on the
--	--

	Create Solution wizard. 4. Repeat Steps 2 and 3 to add more projects to the solution. 4. Click Finish .
--	--

The solution is created along with the select project(s) and added to the workspace. By default, the Explorer view of the Workspace Documents is configured to show all projects as start points for navigating through the workspace.

To restrict the Explorer view to a specific solution:

1. On the Workspace Documents toolbar, click  (Switch Solution).
2. From the list, select the preferred solution.

To switch to the default Explorer view:

1. Click  (Switch Solution).
2. Select **Show All Projects**.

To switch between solutions:

1. On the Workspace Documents toolbar, click  (Switch Solution).
2. Select the Solution.

Creating a project

A project is a container for all documents that together give rise to an application package. It provides a single design-time view of the development content and can be validated, published, and packaged. A project can be attached to multiple solutions within the same workspace.

To create a project:

1. Open the AppWorks Platform User Welcome Page (using the `http://<machine name>:<port number>/cordys` URL).
2. From My Applications, click  (Workspace Documents).
You may be prompted to select a Workspace if you have not selected one already.
The Workspace Documents window opens.
3. On Workspace Documents, do one of the following:
 - On the toolbar, click  next to  and select Project.
 - Right-click <solution> and select **Add > New Project**.
This step is applicable only if you have created a solution to group your projects in the workspace.
4. On the toolbar, click , and select Project List.
A window opens, displaying the list of properties.

5. Click  on its toolbar.
The Project window opens.
6. Enter **Name**, **Description**, and **Annotations** for the project.
7. Click **Finish**.

The new project is created and added to the Project List. When you create a project directly on a solution, it is directly added to that solution.

Creating a document

AppWorks Platform offers different ways in which you can create documents. Newly created documents are stored in the workspace. You can choose to either place the documents directly under the project or group them in folders in a folder structure.

This topic helps you to get started with document creation. However, it does not contain detailed end-to-end information pertaining to the creation of any specific document type, because each document has its own requirements. To know the details, refer to the procedure pertaining to that specific document.

Before you begin:

- You must have created a workspace and have the Developer role.

While creating documents, ensure that you provide unique names to all the documents so that they are distinct. Also, provide distinct names to folders that contain these documents. Do not repeat the name of a folder or document, even if creating it in a separate project. This is to avoid any kind of conflict pertaining to the location, during publish, or use at run time. For information about making folders and documents distinct, see topic on using Qualified Names.

To create a document:

1. Select one of the following starting points.
 - a. If your starting point is Welcome Page >  <Applications>:

Welcome Page >  > <Applications>	<ul style="list-style-type: none"> ■ Click  (New AppWorks Platform Document). <p>The New AppWorks Platform Document window opens displaying all the documents.</p> <ol style="list-style-type: none"> a. Click  (Workspace Documents). The Workspace Documents window opens. b. On the toolbar, click  next to  and select Other. The New AppWorks Platform Document window opens displaying all the documents. a. Click  (Workspace Documents).
---	---

	<p>The Workspace Documents window opens.</p> <p>b. Right-click Project or Folder and select New > Other.</p> <p>The New AppWorks Platform Document window opens displaying all the documents.</p>
--	---

b. If your starting point is **Existing Document**:

<p>Existing Document</p>	<p>1. Open an existing document.</p> <p>2. On the toolbar, click  (Quick Access Menu) > New.</p> <p>The list of documents appears.</p> <p>1. Open an existing document.</p> <p>2. On the toolbar, click .</p> <p>A new blank document of the same type opens.</p>
--------------------------	--

2. For each fresh session of AppWorks Platform, you are prompted to select a Workspace on the Organize Workspaces window, before you proceed with any other activity. This is an intermediate step. To bypass this step, select **Don't ask me again** on the Organize Workspaces window.
3. From the **New AppWorks Platform Document** window, select the document you want to create.
The associated editor for the document opens.

Available documents

The following table lists all the available documents.

Icon	Document name	Purpose
	Application Connector	To create a custom application connector.
	Business Calendar	To create a business calendar.
	Business Calendar Exceptions	To create business calendar exceptions.
	Business Event Response	To create a Business Event Response that keeps track of multiple events and compares those event parameters and outcomes against a set of business rules; and takes appropriate action if the Event falls within the predefined criteria.
	Business Identifier	To define and use Business Identifiers.
	Business Measure	To create a Business Measure that helps measuring an aspect of a business process or any external data.

Icon	Document name	Purpose
	Business Process Category	To create a business process category.
	Business Process Model	To design, debug, and run business process models.
	Cascading Style Sheet	To create the cascading style sheets.
	Case360 EIS Connection	To create an entity connection to an external Case360 system. See the AppWorks Platform Low-Code Design Guide on My Support.
	Calendar Exceptions	To create business calendar exceptions.
	Case Model	To design and run case models.
	Composite Control	To create a composite control that can be used in XForms.
	Content Map	To create a common repository to maintain pairs of source and target values.
	Data Transformation	To create a document that represents the transformation between the source and target data models.
	Database Metadata	To abstract database contents and create its metadata.
	Data Quality Plan	To create a data quality plan that helps to manage the quality of data.
	Data Quality Service	To create a data quality plug-in that helps to configure a data quality plan.
	Decision Table	To create a decision table that contains one or more rules acting on a business object.
	Dispatch Algorithm	To create a custom dispatch algorithm that determines to whom the message is delivered.
	Email Configuration	To configure email. See the AppWorks Platform Low-Code Design Guide on My Support.
	Email Model	To create Email model on a business object.
	Entity	To create a native entity. See the AppWorks Platform Low-Code Design Guide on My Support.
	External User Interface	To create interfaces for external web pages.
	History Log	To create a history log for an entity-based application. See the AppWorks Platform Low-Code Design Guide on My Support.
	Homepage Layout	To create a home page for an entity-based application.

Icon	Document name	Purpose
		See the AppWorks Platform Low-Code Design Guide on My Support.
	HyperText Markup Language	To create HTML files. Clicking this icon will open a text editor. You can create the required HTML code or copy an existing code in the editor. For help on the coding, see HyperText Markup Language .
	Java Archive Definition	To create a Java Archive (.jar) by compiling the Java sources (.java) available in the project.
	Java Class Metadata	To create a metadata of the existing Java Archive (.jar) or Java classes (.class) on which Web services can be generated.
	Java source	To create Java sources.
	Javascript source	To create JavaScript sources. Clicking this icon will open a text editor. You can create the required JavaScript code or copy an existing code in the editor. For help on the coding, refer to Javascript source .
	Javascript Message Bundle	To create JavaScript Message Bundles.
	KPI	To create KPI for monitoring and capturing process or external data source related information and triggering business actions and outbound events as specified.
	MBPM EIS Connection	To create an entity connection to an MBPM system. See the AppWorks Platform Low-Code Design Guide on My Support.
	MDM Hub Data Store	To create MDM hub data store.
	MDM Model	To create an MDM model containing hub and spoke data stores.
	MDM Sequence Generator	To generate a sequence identifier for a data entity. This ID uniquely identifies the business object.
	MDM Sniffer	To create a sniffer that tracks the changes in the data store
	MDM Spoke Data Store	To create MDM spoke data stores
	MDM State Model	To create an MDM State model
	MDM Subscription Handler	To create an MDM subscription handler

Icon	Document name	Purpose
	Process Monitoring Object	To create Process Monitoring Object; a set of associated attributes from a business process and the related contextual web services, which need to be monitored as a logical group.
	AppWorks Platform EIS Connection	To create an entity connection to a Cordys system. See the AppWorks Platform Low-Code Design Guide on My Support.
	Role	To create a role.
	Rule	To create a rule that defines the business logic of a business object.
	Rule Group	To create a rule group to organize functionally similar rules for better management.
	Schedule	To create schedules that can trigger time-based events or processes.
	Theme	To create custom theme for an organization. See the AppWorks Platform Low-Code Design Guide on My Support.
	User Interface	To create interfaces for your applications.
	Web Library Definition	To transform a HTML file as a web document and use it in AppWorks Platform.
	Web service	To create a container that holds generated Web service operations and their Web service interface.
	WS-AppServer Package	To create a WS-AppServer package that contains Java classes generated on database tables, custom classes, methods, attributes, schemas, and Web service operations.
	XML	To create an XML document in AppWorks Platform.
	XML Schema	To create an XML schema of a document in AppWorks Platform.
	XML Store Definition	To store XML documents as run-time objects in XML Store.

You may notice the following variation in the way the documents are created:

- You may need to create the document first using the associated editor and then save it.
- You may need to save the document first and later open it and work on it.

To save a document:

When you save a document, you will be prompted to select a location to store the document.

1. You can select an existing project or folder, or create a new project or folder, and save the document.
2. To distinguish the contents of one folder from another at run time, set Qualified Name (QN) for folders.
The QN attributes a unique name to the project contents. At run time, when you install the packaged application, the QN prevents content overwrite. For more details, see [Using the Qualified Name](#).
3. For the exact procedure relevant to the document you want to create, refer to the task associated with that document and follow the hyperlink in the table.

The document is created and you can view it in Workspace Documents either in Explorer view or My Recent Documents view.

Using the Quick Access Menu

To use the Quick Access Menu:

- Click  (Quick Access Menu) on the top-left corner of any editor.
A menu displays the available options.

Using these options, you can perform several generic tasks. The actions that you perform here are similar to what you perform through Workspace Documents either on the Welcome page or in Classic View. It just provides another convenient point of development.

The following table lists each item of the Quick Access Menu and its use.

New	Creates a new document.
Open	Opens an existing document available in the Workspace Documents.
Save As	Saves the current document with a different name at the same location or a different location.
Validate	Builds the document and validates if it is fit to be published.
Publish	Publishes the document to an organization.
Used By	Indicates the shared reference details of the document (other documents that have references to a particular document). It identifies and displays the documents that use the contents of a project or a folder. This information is useful and serves as a notification when you delete a project or a folder that is a reference to external documents.
Properties	Displays the properties of the document, which can be edited.

This is a standard set of options. However, you may see additional options depending upon the editor you open. They are specific to that document type.

Chapter 16

Working with web services

Web service is a software system designed to support machine-to-machine interaction over a network. This describes a standardized way of integrating web-based applications using XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data; SOAP is used to transfer the data, WSDL to describe the services, and UDDI to list the services available.

Web services allow different applications from different sources to communicate with each other without consuming much time. As the communication is in XML, Web services are also platform independent. For example, Java-based applications can communicate with Perl-based applications, Windows applications can communicate with UNIX applications.

Web services do not require the use of browsers or HTML. Web services are sometimes called Application Services.

AppWorks Platform supports generation of Web services on the following models:

- [Business Process Model](#)
- [Database Metadata](#)
- [Object Template](#)
- [Java Classes](#)
- [Data Transformation](#)
- [Decision Table](#)
- [WS-AppServer Package](#)
- [Custom Logic](#)
- [External Web services](#)
- [Email Model](#)
- [HTTP Connector](#)

Generating web service operations on databases

AppWorks Platform facilitates working with database tables, views, and stored procedures and using the metadata to create Web services. You have an option to create a metadata of the database - a representation that contains details of the database tables, views and store

procedures. The created Database Metadata can be updated with any change that takes place inside the database.

You can create Web service operations on database in the following ways:

- [Generating standard web service operations on a database](#)
- [Generating custom web service operations on database details](#)

Important: WS-AppServer does not completely support the database with multiple schema for generating Web services. Database Metadata supports the option to load tables when the database has multiple schema.

When the Web service operations are generated over such databases, ensure that you do the following:

1. Open the Web service operation document in the CWS that belongs to the table which is not associated to the default schema.

2. Update the content and associate the schema to the relevant table name.

For example, if a table name 'T1' must be associated with a schema 's1', then update 'T1' with 's1.T1'. If this is not updated, the Web service generated on the tables that are associated to the non-default schema will result in the following error during execution:
`com.microsoft.sqlserver.jdbc.SQLServerException: Invalid object name '<TableName>'.`

Generating standard web service operations on a database

This option helps you to generate standard Web service operations directly upon the database tables. This option is useful when you need standard Web service operations to perform certain functions on the database tables, without applying any custom logic. All you need is to create a [Database Metadata](#) that internally uses a metadata service, and generate Web service operations using it.

Alternatively, you can generate Web service operations directly on a database. Both these ways result in a similar output.

Before you begin:

- WS-AppServer service must be configured and should be available as the metadata service.
- Database Metadata must be created on the metadata service.

To generate standard web service operations on a database:

1. [Select a starting point](#) and click  (Web Service) to create a Web Service.
The Web Service Wizard opens.
2. Provide the necessary details to generate the Web service operations on a Database.
3. Enter details on a couple of pages.
4. Click Next to proceed to the subsequent page and continue until you complete entering information on the wizard.

5. Click **Finish** on the wizard.

The Web Service Generation Wizard closes.

The Web service containing the Web service interface and Web service operations is generated and placed under the specified project/folder.

After you complete this task:

- After generating web services on stored procedures with input and output parameters containing large string data types (LOB), such as CLOB in ORACLE, TEXT, NTEXT, you must do the following before publishing it:

Open the implementation of the web service and remove the attribute `dt="string"` for the large data type parameters as shown in the example below:

Before:

```
<implementation xmlns:c="http://schemas.cordys.com/cws/1.0"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" xmlns="" type="DBSQL">
    <constructor language="DBRPC">
        <query>PROC_TESTCLOB1 (:PARAM1, :PARAM2)</query>
        <parameters>
            <PARAM1 dt="r8" ct="in" />
            <PARAM2 dt="string" ct="out" />
        </parameters>
    </constructor>
</implementation>
```

After:

```
<implementation xmlns:c="http://schemas.cordys.com/cws/1.0"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" xmlns="" type="DBSQL">
    <constructor language="DBRPC">
        <query>PROC_TESTCLOB1 (:PARAM1, :PARAM2)</query>
        <parameters>
            <PARAM1 dt="r8" ct="in" />
            <PARAM2 ct="out" />
        </parameters>
    </constructor>
</implementation>
```

- You must [publish](#) the generated Web service and its contents to the organization.
- For use at run time, [attach the Web service interface to the WS-AppServer Service](#).

Generating custom web service operations on database details

Using this option, you can generate custom Web service operations using a combination of tables, views and using a query that is built by a query builder. The advantage of this feature is that you can conveniently create a Web service that is much more closer to your business requirements, without depending upon any external tool.

Before you begin:

- WS-AppServer service must be configured and should be available as the metadata service.
- Database Metadata is created on the metadata service

To generate custom web service operations on database details:

1. In Workspace Documents (Explorer), open <solution> > <project> >  (Database Schema) <database metadata>. The Database Metadata expands and displays its contents.
2. Under Database Metadata, expand either Tables or Views, right-click a specific table or view, and select **Generate Custom Web Service Operation**. The DB Table window opens, displaying the selected table/view and its fields, all the expressions required to build the query, and options to check and validate the query.
3. Right-click the table/view from the Tables/Views list and select a relevant option:

View all Relations	To add all the related tables of the selected table to the Views/Tables list.
Choose Tables	To add new tables to the Views/Tables list for generating the query.
Choose Views	To add new views to the Views/Tables list for generating the query.

4. Build the query using information available in **Fields** and **Expressions**:
 - Select the field that needs to be added in the select Area.
 - Select the area where the field has to be added and click **Add**.
 - If the area is not selected, double-click the field at the end of the content in the select Area.
 - Go through the following references that will help you to build the query:
 - You can create a Composite Query by selecting more data from than one table. To do this, select the Composite Query check box. See [Types of Queries](#) for information on the types of queries that can be generated.
 - You will see several tags (in red) that need to be replaced with proper values and operators.
5. Click **Validate**, **Check Query**, and **View** to ensure that the query is accurate and complete in all respects.
The results appear in different dialog boxes.
6. Click **Generate Custom Web Service Operations**.
The Web Service Interface Generator dialog box opens.

7. Provide details such as the **Name** and **Description** of the Web service operation, the Name of the Web service interface, the location to save the Web service operation, and click **Generate**.

The Web service along with the Web service interface and Web service operation(s) is generated and placed at the selected location.

After you complete this task:

- You must [publish](#) the generated Web service operations to the organization and [attach the Web service interface to the WS-AppServer Service](#) before you can use them at runtime.

To edit the generated Web service operation:

1. Right-click it and select Edit Web Service Operation.
The Web Service Operation <table name> - Binding Operation window opens.
2. Modify the details and click Update Web service Operation to regenerate it.

To edit the implementation of the generated Web service operation:

1. Double-click the <Web service operation> and open the **Binding Operation Properties** window.
For example, change the value for Type, or rename the Parameter, and so on.
2. Ensure that whatever you are modifying is functionally correct and valid. Otherwise, the changes may cause the Web service operation to fail.

Generating application and business logic on relational data

WS-AppServer provides a framework for rapidly developing and deploying applications. Applications developed using this framework can leverage all the features of AppWorks Platform as well as the traditional application server. WS-AppServer includes a predefined set of functions such as database persistence management, inter-process communications, authorization, authentication, stability, scalability, transaction management, and performance management. It can also be used to extend the logic locked inside existing applications and to build composite applications. The logic that encompasses this functionality is exposed as a Web service and is versatile in its applicability when compared to the traditional applications.

WS-AppServer allows application developers to extend the logic of the database-generated models. The extended logic is automatically exposed as Web services, enabling easy remote access and composition in business process models.

WS-AppServer provides an easy and simple way to develop applications. It facilitates creation of containers (packages) to hold the database-generated logic. It provides a user-friendly interface to structure and store data. Support for automatic generation of Web service operations is an added advantage. WS-AppServer also lets you add custom business logic to the existing models.

The following illustration demonstrates the application logic generation process that takes place within WS-AppServer.



Generating application logic using WS-AppServer involves the following activities:

1. Create a [Database Metadata](#).
2. Do one of the following:
 - Create an empty package, load package with database details, and [generate Java code](#).
 - Create a Package containing XML Schemas and Java Code.
3. [Create Web service Interface](#)
4. [Attach the Web service Interface to the WS-AppServer Service](#).

The following table highlights the features of WS-AppServer and the associated benefits.

Features	Benefits
Manages applications based on databases with the help of the WS-AppServer business logic abstraction layer. Segregate application logic and database layers.	Segregate application logic and database layers.
Supports all major database systems, such as SQL	Greater flexibility in data

Features	Benefits
Server, Oracle, and PostgreSQL. Greater flexibility in data management and application development.	management and application development.
Exposes business objects from databases in the form of Web services and manages them through a comprehensive development tool. Easy management of business objects with the help of the development tool.	Easy management of business objects with the help of the development tool.
Auto-generates and manages services, thus enabling transaction management, remote access, and Web service exposure. Faster and lighter application logic using pre-built logic abstraction layer.	Faster and lighter application logic using pre-built logic abstraction layer.
Integrates with Rules to enable building dynamic business logic, in addition to the static compiled business logic. Enhanced agility in business by managing dynamic business logic in the form of rules.	Enhanced agility in business by managing dynamic business logic in the form of rules.
Integrates with XForms to allow the Web services generated by WS-AppServer to be exposed to the user interface, and enables quicker development of end-to-end solutions. This separates the business logic from the presentation layer and attaches it to the business logic layer. Enhanced developer productivity by integrating X-Forms with WS-AppServer.	Enhanced developer productivity by integrating X-Forms with WS-AppServer.
Provides extensible business logic with the help of extension class. Generation of Java extension class, facilitating the addition of ready-made simple business logic in the form of Java code snippets. Flexibility to provide custom business object implementation, which helps developers transcend the limitations posed by the tool, and express richer business objects and logic.	Generation of Java extension class, facilitating the addition of ready-made simple business logic in the form of Java code snippets. Flexibility to provide custom business object implementation, which helps developers transcend the limitations posed by the tool, and express richer business objects and logic.
Supports tracking changes in business objects using events and event listeners, and plug additional business logic into them for advanced functions. Intelligently managed services help developers to focus on business logic building.	Intelligently managed services help developers to focus on business logic building.
Supports handling business objects that are	Faster and efficient reuse of

Features	Benefits
composed of several database tables or non-database objects. Faster and efficient reuse of application services because business objects are wrapped as Web services.	application services because business objects are wrapped as Web services.
Delegates business logic in client-side applications to server-side through validation support. Smarter user interface. Thinner client applications because the entire logic is handled by WS-AppServer.	Smarter user interface. Thinner client applications because the entire logic is handled by WS-AppServer.
Supports auditing database tables of the application. Transaction auditing helps in preserving critical information and analyzing it whenever required.	Transaction auditing helps in preserving critical information and analyzing it whenever required.
Is easily configurable to be used in an embedded mode. Facility to embed WS-AppServer into applications, enabling rich server side functionality, where most of the standard services are auto managed.	Facility to embed WS-AppServer into applications, enabling rich server side functionality, where most of the standard services are auto managed.
Is XML compliant to facilitate advanced techniques of data communication. Event-based 4GL type of programming through Java, providing great flexibility to plug enterprise-specific business logic into the application.	Event-based 4GL type of programming through Java, providing great flexibility to plug enterprise-specific business logic into the application.
Exposes business object transaction management as Web services, thus enabling SOA and SOA-based business process modeling. Dynamic management of enterprise business data using custom business logic.	Dynamic management of enterprise business data using custom business logic.

Note: Before you start generating the application and business logic, it is important to configure the WS-AppServer Service. Administrators have the required privileges to perform this task.

By default, Container Managed Transactions is enabled for the WS-AppServer Service Container.

To disable Container Managed Transactions:

1. In the Service Container Properties page, select **Disable Container Managed Transactions**.
2. If **Disable Container Managed Transactions** is selected or cleared for the service containers, restart the corresponding service containers.

Generating the Java code for WS-AppServer models

You may choose to generate the Java code for the selected data sources as part of generating Web service operations on Database Metadata or while [creating the WS-AppServer package using Database Metadata](#). If you used neither of these options, you may still generate the Java code from the WS-AppServer Package Editor by following the procedure given here.

In the generated code, each Java class comprises a Base class and an Extension class. These represent tables in the database and contain methods that may be used during transactions. The BaseClass is an extended class of the StateBusObject class, which is an extension on the BusObject class. The BusObject class holds the generic functions such as insert(), update() and delete(). The BaseClass provides the standard generated get-functions (query). The Extension class is used to customize behavior and add custom methods and properties.

Note: WS-AppServer generates Java code based on the model elements inside the WS-AppServer package. Therefore, the model elements such as model mapped name, package name, model attribute mapped names, and method parameter names should inline with the [Java identifier](#) naming convention.

To generate the Java code for WS-AppServer models:

1. Double-click  <WS-AppServer Package> from either of the following locations:

- Workspace Documents (My Recent Documents) window.
- Workspace Documents (Explorer) > <solution> > <project>.

The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.

2. On the WS-AppServer Package toolbar, click  (Java Class).

The Generate Java Code on WS-AppServer Models dialog box opens.

3. Select classes as per the following options.

<input type="checkbox"/> Auto select related classes <input type="checkbox"/> Overwrite Extension Classes	<p>Used to select the classes that have relations/dependency</p> <p>When selected, all the extension classes will be overwritten. When it is cleared, the existing extension classes will not be overwritten. The Extension Classes for newly added WS-AppServer classes will still be generated. This option is cleared by default.</p> <p>If a WS-AppServer class for which the extension class was previously generated is modified, you must select this check box before generating the code in order to reflect the change. The classes are overwritten; therefore, the custom logic in the class will be lost. If required, you must keep a copy of the original code outside the CWS environment.</p>
--	---

Filter	Displays specific results (classes) depending upon the name or initials entered in the field. Used to narrow down search results
Java Archive Source	Enables selection of a specific Java Archive Source where you can store the generated Java code. <ul style="list-style-type: none">■ Click  (Lookup) to browse and select the WS-AppServer Java Archive Source available in the project.
Class name	<p>Lists all the classes available in the WS-AppServer Package.</p> <ul style="list-style-type: none">■ Select the topmost check box to select all the classes or select each check box to select individual classes.■ You can navigate through the list of classes using the navigation buttons on the interface. <p>If you have selected the Auto select related classes option, all the related classes will also get selected when you select a class.</p>

4. Click **Generate**.

The Java code is generated and is added to the project at the specified location within a folder Java Archive <WS-AppServer package name>.

After you complete this task:

- Publish the generated Java code so that the code is compiled and the Java classes are packaged in a .JAR file. By default, the .JAR file is stored at <AppWorks Platform_installdir>\<instance name>\bsf\runtime\deploy and this location is added to the classpath of the WS-AppServer Service.

The path details are mentioned against the `wsappserver.deploy.folder` property in the `wsapps.properties` file located at <AppWorks Platform_installdir>\<instance name>\components\wsappserver\config. An Administrator can modify the path, if required.

For example, in Windows,
`wsappserver.deploy.folder=C:\PROGRA~1\Cordys\DEFAUL~1\bsf\runtime\deploy`
and in Linux, `wsappserver.deploy.folder= <AppWorks Platform_installdir>\bsf\runtime\deploy`. While changing the path, keep the folder separators as they are in the default path.

Adding standard methods to a WS-AppServer model

When you import tables from the database (add classes), depending upon the tables selected, the attributes and standard methods are automatically generated. However, you can further add standard methods based on the data model that you want to create.

To add standard methods to a WS-AppServer model:

1. Double-click the  <WS-AppServer Package> from either of the following locations:

- Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>. The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. In the Models group box, right-click a model and select **Add > Standard Methods**. The Add Standard Methods to Class dialog box opens.
 3. From the Select Attribute table, select the attributes for which you want to generate the standard methods.
If you want to select all the attributes in the table, select the check box next to the table header.
 4. In **Object name used in the method**, provide a name for the object.
The name you provide is used by WS-AppServer to generate the methods. It is mandatory.
 5. Select the **Do the selected attribute(s) uniquely identify an object?** check box only if the selected attributes form a unique record and do not have duplication.
 6. Click **OK**.

The following four standard methods are generated and added to the Methods group box:

- `get<objectname>Object`
The `get<objectname>Object` method is generated only if you have identified the attributes as unique (see Step 5). The variable `<objectname>` refers to the name that you provided (see Step 4).
- `get<objectname>Objects`
- `getPrevious<objectname>Objects`
- `getNext<objectname>Objects`

Adding a parameter to a method

Each WS-AppServer method has parameters that are created by default. However, you can add more parameters based on the data model that you want to create.

To add a parameter to a method:

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>. The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. In the Models group box, click a model.
The Attributes, Methods, and Relations tabs are updated with the details of the model.
3. Click the **Methods** tab.
The Method group box and Method Properties group box open.

4. In the Method group box, click a <method>. The corresponding properties appear in Method Properties group box.
5. In the Method Properties group box, under Parameters, click  (Add). The Parameter Properties dialog box opens.
6. Provide the required details and click **OK**.

A parameter is added to the selected method and is displayed under Parameters in the Method Properties group box.

Adding custom methods to a WS-AppServer model

Using the data in the database tables, WS-AppServer generates standard methods. However, these methods may not meet all the requirements of implementing business logic in applications. To accommodate such needs, WS-AppServer facilitates adding custom methods as per your specifications.

To add custom methods to a WS-AppServer model:

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>.

The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.

2. In the Models group box, right-click a model and select **Add > Method**. The Method Properties dialog box opens.
3. Fill in the required details on the dialog box and click **OK**. The new method is created in the Methods group box.

The custom method is added to the class.

After you complete this task:

- The newly created method does not contain any parameters. To make it operational, add parameter(s) to it.

Adding an attribute to a WS-AppServer model

When you create a WS-AppServer package, the attributes and standard methods for the models are automatically generated. However, you can add more attributes based on the data model that you want to create.

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>.
- The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.

2. In the Models group box, right-click a model and select **Add > Attribute**.
The Attribute Properties dialog box opens.
3. Fill in the required details on the dialog box and click **OK**.
The new attribute is created under the Attributes group box.

An attribute is added to the WS-AppServer model and is displayed in the Attribute group box.

Extending business logic in WS-AppServer generated applications

WS-AppServer helps you to build applications that support end-to-end business of an organization. WS-AppServer creates models upon databases and Java sources, and generates Web service operations using those models. In other words, the sources are transformed and exposed as Web services. The exposed Web services are readily available to be used by any application based on SOA architecture. WS-AppServer also facilitates code customization so that the existing functionality of the application can be extended.

The business logic that is available in the form of database-generated classes and methods may fulfill the business requirements to a certain extent. However, if you want to implement additional logic to customize the application and tune it to business requirements, you will need to modify the way application works.

WS-AppServer provides you the necessary framework support with which you can implement the custom logic and design the applications to behave just as you want.

While modeling data and generating the application content, WS-AppServer builds both static and dynamic business logic. The Base class of the generated Java code contains the static business logic, whereas the Extension class is a structure with provision to include dynamic business logic in the generated business logic. The following table shows the basic differences between the types of business logic generated by WS-AppServer.

Static business logic	Dynamic business logic
Rigid logic	Dynamically changes based on several parameters and constraints
Directly mapped to the database	Facilitates addition of rules and constraints
Reliable and stable performance	Varied performance based on the logic executed
Single purpose	Customizable and reusable

If you want to create the entire business logic afresh, you can create [Custom classes](#). WS-AppServer also provides a very useful facility to build rules directly on the data models (both standard and custom) that are created and maintained in the WS-AppServer package.

You can extend the business logic in any of the following ways, depending upon the business requirement:

- Add custom logic to the [extension classes](#) of the generated Java code
- [Create custom classes](#)

- Create composite classes
- Build rules on WS-AppServer models

Implementing custom logic with Extension classes

In WS-AppServer, when Java code is generated and compiled it creates two Java class files for each model - Base (<classname>) and Extension (<classname>). To implement any custom logic, you have to do it in the Extension Java class. These are created within a folder named Java Archive <name of the package>.

To implement custom logic with extension classes:

1. In the **Workspace Documents (Explorer)**, open <solution> > <project> > **Java Archive <package name>** > **com** > <package name>, and double-click the generated extension class (<class>.java).
The <class>.java window opens, displaying the code of the selected extension class.
2. Append the custom logic to the existing code and save the file.

Your custom logic is ready to be used in the WS-AppServer applications.

After you complete this task:

- You need to validate the project to compile the modified Java code, so that it works during runtime.

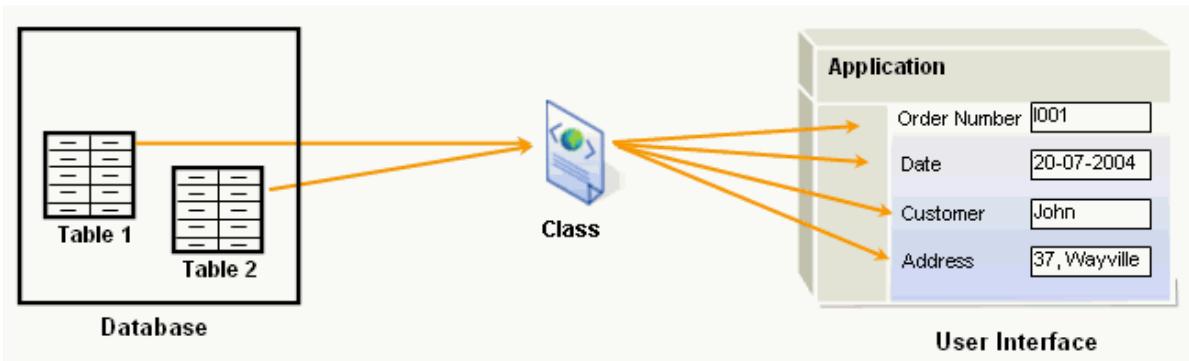
Extending the business logic using custom classes

A custom class is created by combining data from more than one database table and storing it in a single object. Unlike a standard class, a custom class helps in situations where data from a single database table is functionally not sufficient. This is better understood with the help of an example.

Example

There are two database tables - Order and Customer. The Order table contains the Order Number and the Date, whereas the Customer table contains the Customer's name and Address. If a class is created out of the Order table, it will contain only the Order Number and Date but not the customer details. To have details of both the Order table and the Customer table, you need to create a custom class from these two tables.

The following illustration represents a custom class that contains information from both the database tables.



WS-AppServer supports creating the following types of custom classes using Object Layout:

Pure Custom class

A pure custom class is a class that is created without using any other standard or custom class. To design this type of custom class, you need to explicitly implement read and write functionality (from database to Java, and vice-versa) for that class. For example,

```
<MyCompany>
<Director>string</Director>
<Manager>string</Manager>
<Department>string</Department>
<EmployeeID>i4</EmployeeID>
</MyCompany>
```

Derived Custom class

A derived custom class is a combination of one or more standard classes. For derived custom classes, WS-AppServer generates the read and update methods, which handle data exchange from the Java layer to the database.

Query-based Custom class

A query-based custom class is generated based on a SQL query created in WS-AppServer. For a detailed description, see [Creating a Query-based Custom Class](#).

Inherited Custom class

A class that is inherited from another class. For example, when you create an Inherited custom class called InheritedClass that inherits another class called Categories, you get the following layout.

```
<class>
<name>InheritedClass</name>
<type>inherits</type>
<inherits>Categories</inherits>
<attributes
xmlns="http://schemas.cordys.com/wsappserver/WSAppServerClass/1.0"
```

```

<xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"/>
<methods>
<method cursor="false" implementation="default:GetByAttributes"
soap="true" soapresult="tupleset" static="true">
<name>getInheritedClassObject</name>
<return>
<type occ="1">InheritedClass</type>
</return>
<parameters>
<parameter>
<name attribute="CategoryID">CategoryID</name>
<type>ui2</type>
</parameter>
</parameters>
</method>
<method cursor="true"
implementation="default:GetByAttributesRange" soap="true"
soapresult="tupleset" static="true">
<name>getInheritedClassObjects</name>
<return>
<type occ="*">InheritedClass</type>
</return>
<parameters>
<parameter>
<name attribute="CategoryID">fromCategoryID</name>
<type>ui2</type>
</parameter>
<parameter>
<name attribute="CategoryID">toCategoryID</name>
<type>ui2</type>
</parameter>
</parameters>
</method>
</methods>
</class>

```

Note: Custom classes can be either flattened or nested. A flattened custom class has all its attributes at a single level. In other words, its object layout does not have a nested structure. A nested custom class has a nested structure and its object layout contains more than one custom class.

Creating a custom class

A custom class can be either a pure custom class, a derived custom class, or an inherited custom class. Each type of class is created in a different way, using different options that are available. However, the procedure for creating a custom class is common for all the three types.

To create a custom class:

1. Double-click  (<WS-AppServer Package>) from either of the following locations:

- Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>. The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. On the WS-AppServer Package toolbar, click  (Create Class from Object Layout). The Create Class from Object Layout dialog box opens.
 3. On the Layout Tree tab, in the Class/Attribute Information box, provide a name for the class and fill other details.
 4. In **Class Layout**, right-click the <root class> and do the following:
- | | |
|----------------------------|---|
| To add a class within | Select Add > Class , and provide a name for it when prompted. |
| To add an attribute within | Select Add > Attribute , and provide a name for it when prompted. |
5. In **Class/Attribute Information**, provide other details to define the attribute. The object layout is formed in the tree structure.
 6. Depending upon your business logic requirements, you can add as many number of classes and attributes to the root class.
 7. You can view the object layout in XML on the Layout XML tab.
 8. You can also modify the XML structure there and see the changes reflecting in the tree layout.
 5. Click **Create**.

The custom class is created under the Models group box, bearing the  icon next to it.

After you complete this task:

- Select the custom class that you created and generate Java code and Web service operations for it.
- If you need to modify the Custom class, right-click the class in Models group box and select **Edit Object Layout**. The Object Layout Editor opens, displaying the XML structure on the Layout XML tab.
- If you need to delete a Custom class, right-click the class in Models group box and select **Delete**.

Creating a query-based custom class

This option helps you to build a custom class using a query. Using the query-builder of WS-AppServer, you can bind database objects together in a query string and make it executable. Methods generated on such classes run in the form of a query and thus automate data retrieval.

To create a query-based custom class:

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>.The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. On the WS-AppServer Package toolbar, click  (Create Class from Query).
The Create Class from Query window opens.
3. From **Select Database Metadata**, browse and select the Database Metadata.
The tables available in the selected Database Metadata appear under DB Object.
4. Select the required tables and click **Next**.
5. Use the navigation buttons to see all the tables. You can select more than one table or all the tables, to build the query.
6. To view database in its view format, select **Views**
It supports SQL query structure, where you can provide the relevant details depending upon the field selection, to build the query. It is mandatory to provide a valid query.
6. Click **Next** to proceed and do one of the following:
 - If it is a derived class, select the class from **Derived From**.
 - If it is not a derived class, provide a name for the class in **View Class Name**.
7. Click **Finish**.

The query is built and the custom class based on the query is created in the Models group box.

Example

You want to create a query custom class that will retrieve shipping location details of a particular order for a customer and also retrieve the date of placing the order. The important fields that will help in getting this information are CustomerID, OrderID, OrderDate, and ShipCity. Since this information is not available in a single table, you have to select two tables - Customers and Orders.

To create the query custom class on this logic:

1. On the Create Class from Query window, from the DB Object table, select **Customers and Orders** and click **Next**.
2. In the space for the query, type the following query and click **Next**.

```
SELECT Customers.CustomerID, Customers.City, Orders.OrderDate,  
Orders.ShipCity From Customers,  
Orders WHERE Customers.CustomerID= Orders.CustomerID
```

`CustomerID` and `City` are from the `Customers` table, and `OrderDate` and `ShipCity` are from the `Orders` table. Also, `CustomerID` exists in the `Orders` table as Foreign Key.

3. Under Derived From, select **Customers** because that contains the primary key `CustomerID` which will be used to retrieve the details.
`CustomersView` is displayed in the View Class Name field.
4. Click **Finish**.
A message is displayed indicating successful creation of the query custom class.
5. On the WS-AppServer Package editor, under Models, search for the `CustomersView` query custom class (indicated by ) and select it.
The attributes (`City`, `CustomerID`, `OrderDate`, and `ShipCity`) of that class appear on the Attributes tab.
6. Under **Models**, right-click the `CustomersView` query custom class and select **Add > Standard Methods**.
7. On the Add Standard Methods dialog box, select the attribute `ShipCity` (because you need the shipping location details) and provide a name for the Object.
8. Select **Does the selected attribute(s) uniquely identify an object?** if you want to treat the object as unique.
If you select this, you can retrieve details of an object based on the selected attribute(s) alone.
9. Click **OK**.

The methods are generated and displayed on the Methods tab.

Now that you have generated the `CustomersView` query custom class and have added Standard Methods to it, you have to [generate Java code](#) followed by [Web service operations](#) for the `CustomersView` query custom class, to make it functional.

After you complete this task:

- Select the query-based custom class that you created and [add methods](#) to it.
- Subsequently, [generate Java code](#) and [Web service operations](#) on it and [publish](#) the Web service to an organization, so that you can use them at run time.

Creating and using composite classes

In complex applications, standard classes derived from single database tables may be deficient in providing support to all the functions within the application. In such situations, there is a need for classes that contain information from more than one database table.

These are called Composite classes, which are created by clubbing two or more objects together. A virtual reference is created between the primary object and the secondary object such that the request is framed reading data from both the objects. The structure of the composite class is created using the Object Layout provided by WS-AppServer. These classes are created in a nested structure.

Example

The following is an example of creating a composite class using the Orders and Order Details tables from the SQL-Server Northwind's database. Using the Object Layout provided by WS-AppServer, a custom class called FullOrder is defined, which aggregates a Header element and Lines element. The Header element is represented by the Orders object, while the Lines element is represented by a collection of Order Details objects (sequenced).

The format used to club one object with another is <Name of the Package in WS-AppServer>::<name of the standard class in that package>. In the Object Layout, the custom class would form the following structure:

```
<FullOrder>
<OrderID unique="true">i4</OrderID>
<Header occ="1">HB Northwind::Orders</Header>
<Lines occ="*">HB Northwind::OrderDetails</Lines>
</FullOrder>
```

In the above code, the package is Northwind, and Orders and OrderDetails are the standard classes. Based on this layout, the composite class would be framed as follows:

```
<FullOrder>
<OrderID>11049</OrderID>
<Header>
<OrderID>11049</OrderID>
<CustomerID>XYZ</CustomerID>
<EmployeeID>3</EmployeeID>
<OrderDate>1998-04-24T00:00:00.0</OrderDate>
<RequiredDate>1998-05-22T00:00:00.0</RequiredDate>
<ShippedDate>1998-05-04T00:00:00.0</ShippedDate>
<ShipVia>1</ShipVia>
<Freight>8.34</Freight>
<ShipName>XYZ Shipment</ShipName>
<ShipAddress>Av. Brasil, 442</ShipAddress>
<ShipCity>Campinas</ShipCity>
<ShipRegion>SP</ShipRegion>
<ShipPostalCode>04876-786</ShipPostalCode>
<ShipCountry>Brazil</ShipCountry>
</Header>
<Lines seqId="1">
<OrderID>11049</OrderID>
<ProductID>12</ProductID>
<UnitPrice>38</UnitPrice>
<Quantity>4</Quantity>
<Discount>0.2</Discount>
</Lines>
<Lines seqId="2">
<OrderID>11049</OrderID>
<ProductID>2</ProductID>
<UnitPrice>19</UnitPrice>
<Quantity>10</Quantity>
```

```
<Discount>0.2</Discount>
</Lines>
</FullOrder>
```

Note: Before the composite class is used in a SOAP request, it has to be compiled. The compilation will be successful only if the Java code of both the objects is available.

Building a rule on WS-AppServer models

The default business logic embedded in the Java code and Web service operations generated by WS-AppServer are static in nature. However, they can be customized to embed dynamic business logic that would help address complex computations and requirements of a dynamic business environment. You can achieve this by defining rules in a WS-AppServer application.

WS-AppServer provides a facility to build rules on its data models and leverage the dynamic business logic building facility provided by the Rule Engine. By applying rules, you extend the available business logic, and enable the application to perform complex tasks. Once you define a rule on a WS-AppServer model, it will be executed during runtime, while running the corresponding Web service operation.

Before you begin:

- WS-AppServer Package should contain data models. In order to build a rule, you must ensure that the **Enable Rules** option is selected in the WS-AppServer tab of WS-AppServer Service Container.

To build a rule on WS-AppServer models:

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>.
 The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. In WS-AppServer Package, right-click a model and select Create Rule from the menu. The Rule editor opens.
3. Build the rule as described in [Creating a Business Rule](#).

The rule is created on the WS-AppServer Model and is now a part of the WS-AppServer application. It is triggered during runtime when the Web service operations from this application are executed.

After you complete this task:

- [Publish](#) the WS-AppServer package and the Web service operations generated on it to run-time.

Creating or updating web service interface on WS-AppServer models

A Web service interface is a container for Web service operations generated by WS-AppServer. The grouping is done at the model level and not at the level of individual Web services. Therefore, to create a Web service interface, you need to select the models that contain Web service operations. WS-AppServer provides you the option to create new Web service interface for different models and also to add models to the existing Web services.

To create or update a Web Service interface:

1. Double-click  <WS-AppServer Package> from either of the following locations:
 - Workspace Documents (My Recent Documents) window.
 - Workspace Documents (Explorer) > <solution> > <project>.The WS-AppServer Package window opens, displaying the data models and their corresponding attributes, methods, and relations.
2. On the WS-AppServer Package toolbar, click  (Web Service).
The Generate Web service Interface on WS-AppServer Models dialog box opens.
3. Select classes in the following ways and click **Next**.

Auto select related classes	Enables selection of all the related classes in a single click.
Search	To filter classes based on the characters typed in the field.
Class name	The table header. Enables selection of all the classes.
<class name>	To select an individual class. You can navigate through the list of classes using the buttons above the table, and select individual classes. If you have selected the Auto select related classes option, all the related classes will also get selected, as you select each class.

You proceed to the next page.

4. Under **Generate to Web service Interface**, select one of the following options:

New	Creates a new Web service interface. If you select this option, do the following: <ol style="list-style-type: none">1. Enter the name of the new Web service in the Web Service Name field.2. Enter the name of the new Web service interface in the Web service Interface Name field. By default, it displays the package name as the Web service interface.
-----	---

	<p>3. Click  (Lookup) to browse and select a folder to store the Web service. The name of the folder appears in the Web services Folder Name field.</p>
Existing	<p>Updates an existing Web service interface.</p> <p>If you select this option, click  (Lookup) to browse select the Web service interface from the list. The name of the Web service interface appears in the Web service Interface Name field.</p>

5. Click **Generate** to either generate a new Web service interface or to update an existing Web service interface.

A WS-AppServer Class inherits its namespace from the Ws-AppServer package it belongs to. An Application Developer can change the namespace of this class. In such a case, mapping of objects and namespaces is as follows:

- The Web service Operations defined on the Ws-AppServer class belong to the Package Namespace.
- The Ws-AppServer classes that appear in the SOAP request or the response belong to the Class Namespace.

A new Web service interface is created or the existing Web service interface is updated.

After you complete this task:

- You must [attach the generated Web service interface to the WS-AppServer service](#).

Attaching the web service interface to the WS-AppServer service

This topic describes the procedure to attach the Web service interface to the WS-AppServer Service (metadata service).

Before you begin:

- You must have the role of an Administrator to perform this task.
- You must have published the Web service interface to an organization.
- Ensure that the WS-AppServer Service and the Web service interface are in the same organization. Otherwise, create a WS-AppServer Service in the organization where you have published the Web service interface.

To attach the Web Service interface to the WS-AppServer service:

1. In My Applications, select  (System Resource Manager).
The System Resource Manager window opens.
2. Click  (Show All Service Groups) on the toolbar.
The Service Groups pane opens showing the available Service Groups.
3. Double-click the Service Group (containing the WS-AppServer Service) to which you intend to attach the Web service interface.
The related properties are displayed in the ServiceGroup Properties - <Service Group Name> palette window. It also shows the already attached Web service interfaces listed under Web Service Interfaces.
4. Under Web Service Interfaces, click  (Add).
The Attach Web Service Interfaces dialog box opens.
5. From the Organization/Package list, select the Organization/Package to which the generated Web service interface was published.
The published Web service interfaces appear in the table below.
6. Select the check box next to the Web service interface that you want to add and click **Add** below the table.
The selected Web service interface is added to the existing list of Web service interfaces shown in another table below.
7. Click **Done**.
The Attach Web Service Interfaces dialog box closes and the Web service interface that you added appears under Web Service Interfaces.
8. Click .

The Web service interface is successfully attached to the WS-AppServer Service.

An easier way to attach the Web service interface

An easier way to attach the Web service interface to the WS-AppServer Service is through the Web Service Interface - Web Service Binding window.

1. Double-click the Web service interface and on the Web Service Binding window, select the relevant Service Group from the list.
2. Save the Web service interface.

The Web service interface is attached to the selected WS-AppServer Service.

Setting access control on tables and views of a database

While building an application, you may grant certain roles complete access or partial access to a database table or view, or you may not provide access to a particular database table or view.

The Security feature helps you set access control on a table or view of the database metadata. There are four levels at which you can set the control: Read, Insert, Update, and Delete. The extent to which a user (bearing that role) is able to work with a database

depends upon the level at which access control is set. For instance, the Read access allows the user to retrieve data from a table or view, but does not allow the user to store data into the table or view. The Delete access enables the user to retrieve data, to modify data, to insert data, and to delete data from the table or view.

The Web service operations generated on a particular table or view inherit the access control setting and behave accordingly at runtime.

Before you begin:

- Create a Database Metadata configured to a WS-AppServer Service.
- Create roles and publish them to the organization.

To set access control on tables and views of a database:

1. Perform one of the following steps:

- In Workspace Documents (Explorer), open <solution> > <project> >  (Database Metadata), and do one of the following:
 - a. Expand Tables, right-click  (Table Name) and select Define Runtime Security.
 - b. Expand Views, right-click  (View Name) and select Define Runtime Security.
- On the Database Metadata window, in the Children pane, do one of the following:
 - a. Place the pointer on  (Table Name), click  (Extend Menu) and select **Actions > Define Runtime Security**.
 - b. Place the pointer on  (View Name), click  (Extend Menu) and select **Actions > Define Runtime Security**.

The Security Editor window displays the name of the selected database table or view on its title bar.

2. In the Roles pane, click  (Add).

The Select Role dialog box displays the roles that you created.

3. Select a role for which you want to set access control.

The selected role is shown in the Roles pane.

4. To define the access for the selected role, in the Permissions pane, select **Read, Update, Insert, or Delete**.

5. Click .

The access control is set on the database table or view and is associated to the specific role.

Tip: To set access control on all the database contents, define Runtime Security directly on the Database Metadata in the same way as it is done for an individual table/view.

Setting access control on database procedures

While building an application, you may allow users with certain roles to access a particular database procedure or completely prevent them from accessing it. The Security feature helps you set access control on a procedure of the database metadata. By default, access is given to everyone using the workspace, which contains the database procedure.

Before you begin:

- Create a Database Metadata configured to a WS-AppServer Service.
- Create roles and publish them to the organization.

To set access control on database procedures:

1. Do one of the following:
 - In Workspace Documents (Explorer), open <solution> > <project> >  (Database Metadata) > Procedures, right-click  (Procedure Name), and select Define Runtime Security.
 - On the Database Metadata window, in the Children pane, point to  (Procedure Name), click  (Extend Menu), and select Actions > Define Runtime Security. The Security Editor window displays the name of the database procedure on its title bar.
2. In the Roles pane, click  (Add).
The Select Role dialog box displays the roles that you created.
3. Select a role for which you want to set access control. The selected role is shown in the Roles pane.
4. To block access to the procedure, in the Permissions pane, select **Blocked**.
The selected role will not have access to the procedure.
5. Click .
The access control is set on the database procedure and is associated to a specific role.

Tip: To set access control on all the database contents, define Runtime Security directly on the Database Metadata in the same way as it is done for an individual procedure.

Generating a web service operation on a business process model

You can generate a Web service operation on a published process model in AppWorks Platform based on ExecuteProcess using the Web service generator wizard. This Web service can be used to trigger a process.

For a short-lived process, the Web service has both an input and an output message, whereas for a long-lived process the output message is the instance ID of the instantiated process. If you are generating a Web service for a short-lived process, ensure that there is only one End event with a message or multiple End events with the same message.

Before you begin:

- [Create and Design](#) a Business Process Model.

To generate a Web service by creating a New AppWorks Platform document and using a business process model as source:

1. In Workspace Documents right-click <Project> > **New** > **Other**.
The New AppWorks Platform Document dialog box opens.
2. Select **Web Service**.
The Untitled Web Service - Web Service wizard is initiated.
For information on the fields that display in the Untitled Web Service - Web Service wizard, see [Web Service Interface](#).
3. In **Select the source**, select **Business Process Model**.
The Namespace field is displayed.
4. Enter the **Name**, **Description**, and **Namespace**.
5. Click **Next**.
The next screen of the Untitled Web Service - Web Service wizard opens displaying the Business Process field.
6. To select the business process model basing which you want to generate a Web service, click  associated with the Business Process field.
The selected BPM is displayed in the Business Process field and the Web Service Interface Name, Web Service Operation Name, Activity Name, Monitoring, and Execution Priority fields appear. For information about these fields, see [Web Service Interface](#).
7. Provide relevant information in all the fields.
8. Click **Finish** to exit the wizard.

To generate a Web service on a business process model in the project content tree through Business Process Execution:

1. Right-click required business process model on which you want to generate the Web service, and select **Execution** > **Generate Web Service**.
The <businessprocessmodelname> - Web Service Generation Wizard opens.
By default, in the Select Web Service Definition Set group box, the **New** option remains selected. However, select **Existing** if you want to generate the Web service using an existing Web Service Definition Set.
2. Depending on whether you select (**New** or **Existing**), relevant fields appear for associating a Webservice Definition Set to the Web service that you are generating.
See the [Web Service Interface](#) for more information on the fields within.

3. Click  (Lookup) associated with each of the Webservice Interface Name, Instantiation Source description, Webservice Operation Name fields to select appropriate artifacts.
4. Select relevant information in the Process monitoring and Process Execution Priority fields.
5. Click **Finish** to exit the wizard.

A Web service for triggering the process is generated and is attached to the project content tree.

After you complete this task:

- Attach the Web service to the Business Process Management Service Container and [publish the Web service](#) to organization.

Generating a web service operation on a data transformation model

You can generate Web service operations on a data transformation model in AppWorks Platform using the Web service wizard and use them in a process or a Web page to offer specific functionality to users.

Method Sets are referred to as Web service Interfaces and Methods are referred to as Web service Operations.

Before you begin:

- You should already have a [data transformation model](#) for generating a Web service.

To generate a web service operation on a data transformation model:

1. From the Workspace Documents (Explorer), right-click your project and select **New > Other**.
The New AppWorks Platform Document window opens.
2. Click  (Web Service) from the New AppWorks Platform Document window.
The Untitled Web Service - Web Service wizard is displayed.
3. In **Select the Source**, select **Data Transformation** from the list.
4. Provide the appropriate name, description, and namespace for the Web service operation in the respective text boxes of the wizard.
5. Click **Next**.
The next page on the wizard opens.
6. Click  (Lookup) next to the Select a Data Transformation field.
7. Select the required data transformation from the Select the Data Transformation dialog box that opens and click **OK**.
The name of the selected data transformation is displayed in the Select a Data Transformation field.

8. Type the names for Web service interface, Web service operations in the Web Service Interface Name, and Web Service Operations Name in the respective fields of the Web Services Interface Information group box and click **Finish**.

A Web service Properties page opens, displaying the target namespace, its prefix, and the location URI in the Target Namespace, TNS Prefix and Location URI fields.

The data transformation model can now be used as a Web service operation and is displayed under Workspace Documents (Explorer) > <project> folder.

To use the data transformation Web service:

1. Select one of the following options:
 - Open Workspace Documents (Explorer) > <project>, and double-click the Web service Interface. In the Web Service Binding Properties pane that opens, select Data Transformation from the Service Group list. This will bind the Web service interface to the Service group.
 - Alternatively, you can attach the Web service to the Data Transformation Service Group at runtime.
2. Right-click the required <Web service operation> in Workspace Documents (Explorer) and click **Publish to Organization** to make it available to an organization. If you do not specify the organization, it is published to the default organization that is set for the authenticated user.

Generating web service operations on Java classes

This option facilitates generating Web service operations on a Java Class Metadata that is mapped to a JAR file or Java classes that were created outside AppWorks Platform environment. This feature helps you to import any application logic in the form of Java files that you think is important or relevant to the application being built. The Web service operations that you create on these files can be published in the same way you create and publish Web service operations on a database.

Before you begin:

- Ensure that the **Java Class Metadata** model is available.

To generate web service operations on Java classes:

1. Select one of the starting points:
 - In Workspace Documents (My Recent Documents) window, point to  (Java Class Metadata), click  (Extend Menu) and select Actions > Generate Web services.
 - Open Workspace Documents (Explorer) > <solution> > <project>, right-click <Java Class Metadata> and select **Generate Web services**.

The Java Class Metadata window opens, displaying the name of the Java Class Metadata on its titlebar.

2. In the Java Class Metadata Explorer, expand the Java Class model and select the  (Java Class) on which you want to generate the Web services. You can only select Static methods. The methods of the selected Java Class appear on the right-side pane.
3. On the **Methods** tab, select the methods upon which you want to generate the Web service operations. You can also perform the following optional activities:
 - Change the name of the Web service operation to suit your functional needs.
 - For a selected Web service operation, click  (Lookup) and do the following on the Advanced dialog box:
 - a. If you want to treat a parameter with i4 data type as XML in a request, select the **XML Node** check box.
The check box is displayed only if the data type is i4. XML Node is not applicable for other data types.
 - b. Modify the parameter name depending upon your functional needs.
The implementation will display this name instead of the default parameter name. Even if you change the name, the values will remain intact.
 - c. Depending upon whether you want the parameter in the request or in response or in both, set the Scope to **in** (only in request), **out** (only in response), or **inout** (in both request and response).
 - d. Click **OK** to save the changes and close the dialog box.
4. On the Web service Interface Details tab, do one of the following:
 - Generate Web service operations under a new Web service interface. You may change the names in the Web service, Web service Interface Name and the Namespace fields if you do not want the default ones.
 - Click  (Lookup) to browse and choose a Location to create the Web service interface.
 - Select **Existing** to generate the Web service operations within an existing Web service interface.
 - Click  (Lookup) to browse and choose the Web service interface.
5. Click **Finish**.
The Java Class Metadata window closes.

The Web service operations for the selected Java Class are generated. The Web service bearing the Web service Interface and Web service Operations for the selected Java Class (es) is stored at the same location where the Java Class Metadata is stored.

After you complete this task:

1. In the Web service Interface properties, select the Service to which this Web service will be attached.

2. Publish the generated Web service operations to the organization and attach the Web service interface to the WS-AppServer Service before you can use them at runtime.
3. The JAR location should be added to the classpath.
4. Restart the WS-AppServer service.

Generating a web service operation on an object template

You can generate Web service operations on an object template in AppWorks Platform using the Web service wizard and use them in a process or a web page to offer specific functionality to users.

Object Template is deprecated in AppWorks Platform BOP 4.1. Use [WS-AppServer Custom class](#) instead. See Using [WS-AppServer Custom Class as an Alternative to Object Template](#) for more information on the procedure and guidelines to use WS-AppServer Custom Class as an alternative to Object Template.

Before you begin:

- You must already have [created an object template](#).

To generate a web service operation on an object template:

Method Sets are referred to as Web service Interfaces and Methods are referred to as Web service Operations.

1. From the Workspace Documents (Explorer), right-click your project and select **New > Other**.
The New AppWorks Platform Document window opens.
2. Click  (Web Service).
The Untitled Web Service - Web Service wizard opens.
3. In **Select the Source**, select **Object Template**.
4. Provide **Name**, **Description**, and **Namespace** for the Web service operation.
5. Click **Next**.
The next page on the wizard opens.
6. From Select an Object Template, click  (Lookup).
The Object Template dialog box opens.
7. Select the required object template and click **OK**.
The name of the selected object template is displayed in the Select an Object Template field.
8. Type the names for Web service interface and Web service operations in the Web Service Interface Name and Web Service Operations Name in the respective fields of the Web Services Interface Information group box and click **Next**.
The next page of the wizard opens.

In the **Method Definition** section, the name of the Web service operation is displayed automatically in the **Name** field.

9. Select a **Version**.
 - If you want the version tag to be included in the SOAP request of the Web service, select **True**.
 - Including the version tag in the SOAP request will enable you to specify the space (Organization, User, or ISV) from where the objects need to be retrieved.
 - If you set the version to **False**, the SOAP request will not contain the version tag, and objects will be searched in all the three spaces. By default, this value is False.
 10. Select **Metadata** option.
 - If you want the metadata of the object to be included in the response of the SOAP request, select **True**.
By default, this value is False.
 11. Build the required expression for the Web service operation by dragging the required special attributes and operators from the Special Attributes and Operators folders to the Implementation text box.
 12. Click **Finish**.
A Web service Properties page opens, displaying the target namespace, its prefix and the location URI in the Target Namespace, TNS Prefix and Location URI fields.
- The object template can now be used as a Web service operation and is displayed under Workspace Documents (Explorer) > <project>folder.
- To make it available to an organization:**
- Right-click the required <Web service operation> in Workspace Documents (Explorer) and click **Publish to Organization**. If you do not specify the organization, it is published to the default organization that is set for the authenticated user.
 - Open Workspace Documents (Explorer) > <project>, and double-click the just created Web service Interface. In the Web Service Binding Properties pane that opens, select CoBOC from the Service Group list. This will bind the Web service interface to the Service group.
 - Alternatively, you can attach the Web service to the CoBOC Service Group at runtime.

Generating a web service operation on a decision table

You can generate Web service operations on a decision table in AppWorks Platform using the Web service definition wizard and use them in a process or a web page to offer specific functionality to users.

Method Sets are referred to as Web service Interfaces and Methods are referred to as Web service Operations.

Before you begin:

- You must already have [built a decision table](#) to generate a Web service operation.

To generate a web service operation on a decision table:

1. From the Workspace Documents (Explorer), right-click your project and select **New > Other**.
The New AppWorks Platform Document window opens.
2. Click  (Web Service) from the New AppWorks Platform Document window.
The Untitled Web Service - Web Service wizard opens.
3. In Select the Source, select **Decision Table**.
A Namespace field is added to the page.
4. In **Name** and **Description**, type the appropriate name and description for the Web service operation.
5. In **Namespace**, type the namespace of the Decision Table and click **Next**.
The next page on the wizard is displayed.
6. In Select a Decision Table, click  (Lookup), select the required decision table from the Select the Decision Table dialog box that opens, and click **OK**.
The name of the selected decision table is displayed in the Select a Decision Table field.
7. Type the names for Web service interface and Web service operations in the Web service Interface Name and Web service Operations Name in the respective fields of the Web services Interface Information group box and click **Finish**.
A Web service Properties page opens, displaying the target namespace, its prefix and the location URI in the Target Namespace, TNS Prefix, and Location URI fields.

The decision table can now be used as a Web service operation and is displayed under Workspace Documents (Explorer) > <solution> > <project> folder.

To make it available to an organization:

1. Right-click <Webservice operation> in Workspace Documents (Explorer) and click [Publish to Organization](#) to make it available to an organization.
If you do not specify the organization, it is published to the default organization that is set for the authenticated user.
2. Open Workspace Documents (Explorer) > <solution> > <project> >, and double-click the just created Web service Interface.
In the WebService Binding Properties pane that opens, select Rule Management from the Service Group list. This will bind the Web service interface to the Service group.
3. Alternatively, you can attach the Web service to the Rule Management Service Group at runtime.

Generating web service operations on custom logic

This option helps you to generate Web service operations on custom logic that was pre-built or is in use in some application. Such an option facilitates reusing the existing logic to generate Web service operations and expose them. You can also provide the logic while creating the Web service operations.

Alternatively, you can [create custom classes in WS-AppServer](#), [add methods to them](#), and [generate Web service operations from the WS-AppServer Package editor](#) for those classes. Both these ways result in a similar output.

Before you begin:

- The method implementation of custom logic should be available.

To generate web service operations on custom logic:

1. [Select a starting point](#) and click  (Web Service) to create a Web service. The Web Service Wizard opens.
2. Provide the [necessary details to generate the custom Web service operations](#). You need to enter details on couple of pages.
3. Click **Next** to proceed to the subsequent page and continue until you complete entering information on the wizard.
4. Click **Finish** on the wizard. The Web Service Generation Wizard closes and the Web service containing the Web service interface and Web service operations is generated and placed under the specified project or folder.
5. Open the  (Web service operation) to which you want to add the logic. The Binding Operation window opens, displaying the name of the Web service operation on its title bar and in the Name field.
6. Provide the method implementation in the Implementation text area and click .
7. Close the Binding Operation window.

The custom Web service operation is created.

After you complete this task:

- You must [publish](#) the generated Web service and its contents to the organization. For use at run time, attach the Web service interface to an appropriate Service.

Generate Web Service Operations on Custom Logic - Wizard interface

The Generate Web Service Operations on Custom Logic - Wizard interface opens when you create a Web service at a project or folder level and select Web Service for New AppWorks Platform Document. This interface contains the following fields.

Field	Description
Description	Description of the Web service.
Name	Name of the Web service
Namespace	Namespace that is unique to the Web service
Select the Source	Source on which the Web service is generated. Select Custom Web Service from the list.

Field	Description
Web service Interface Name	Name of the Web service interface.
Implementation Class	Class that contains the application connector information to communicate with the relevant service. Click  (Lookup) to browse and select an application connector. The name of the implementation class is displayed.
Web service Operations	Names of the Web service operations that are part of the Web service interface. Click  (Add) to add a row, and then type a name for the Web service operation. Repeat this to add more Web service operations.

Generating web service operations on a WS-AppServer package

This option helps you to generate Web service operations on the data models and the Java logic that exist in a WS-AppServer Package. Unlike generating standard Web service operations on a Database Metadata, this option helps you to generate Web service operations even on the custom logic that you may have added to a WS-AppServer Package.

Alternatively, you can [generate Web service operations from the WS-AppServer Package editor](#). Both these ways result in a similar output.

Before you begin:

- You must [create a Database Metadata](#) and [generate WS-AppServer Package](#) using it.

To generate web service operations on a WS-AppServer package:

- Select a starting point and click  (Web Service) to create a Web service. The Web Service Wizard opens.
- Provide the necessary details to generate the Web service operations on a WS-AppServer Package.
You need to enter details on couple of pages.

3. Click **Next** to proceed to the subsequent page and continue until you complete entering information on the wizard.
4. Click **Finish** on the wizard.

The Web Service Generation Wizard closes.

For the Web service operations that are associated to the tables that have non-default schema associated:

- You must manually associate the schema name to the table name. Double-click the Web service operation and associate the schema name to the table name in the Implementation Details that appear.

The Web service containing the Web service interface and Web service operations is generated and placed under the specified project/folder.

After you complete this task:

- You must [publish](#) the generated Web service and its contents to the organization. For use at run time, [attach the Web service interface to the WS-AppServer Service](#).
- If required, you can delete a Web service operation from the Ws-AppServer modeler. However, deleting the Web service operation from WS-AppServer modeler does not delete the corresponding Web service operation in the CWS <project> or <folder>. You must manually delete it from the <project> or <folder> in the workspace.

Generate Web Service Operations on WS-AppServer Package - Wizard interface

The Generate Web Service Operations on WS-AppServer Package - Wizard interface opens when you create a Web service at a project or folder level, and select Web Service document for New AppWorks Platform Document. This interface contains the following fields.

Field	Description
Select the Source	Source on which the Web service is generated. Select WS-AppServer Package from the list.
Name	Name of the Web service.
Description	Description of the Web service.
Namespace	Namespace that is unique to the Web service.

Field	Description
Select Package	Name of the WS-AppServer package. Click  (Look up) to browse and select the WS-AppServer

Field	Description
	package.
Web service Interface Name	Name of the Web service interface.
WS-AppServer Class	Classes in the WS-AppServer package. Select the topmost check box to select all the classes. Otherwise, scroll through the list and select specific classes.

Generate Web Service Interface on WS-AppServer Models - Wizard interface

The Select Classes section contains the following fields.

Field	Description
Auto select related tables	Option to select related database tables that share a relation with the selected table.
Search	Filters model names. Enter the name or initial letters of the model to filter model names and enable quicker selection.
Class Name	Displays the models available in a WS-AppServer package Select the topmost check box to select all the models, or navigate through the list of models using the buttons and select specific models.

The Web services section contains the following fields.

Field	Description
Generate to Web service Interface	Web service interface being created. Default selection: New. Leave the default selection as New to create a Web service interface or select Existing to select from created Web service interfaces.
Web service Interface Name	Name of the Web service interface that displays when New is selected for the Generate to Web service Interface option. The interface name defaults to the package name. Type a different interface name if necessary.
Web service Interface	Existing Web service interface to store the Web service operations that display when Existing is selected for the Web service Interface option.

Field	Description
	Click  (Look up) to select an existing Web service interface from Workspace Explorer.
Web service Name	Name of the Web service that contains the Web service interface and its Web service operations. This field displays when you generate a new Web service interface. Type a name for the Web service.
Web services Folder Name	Name of the folder where the Web service interface is placed. The field displays the package name as part of the folder name by default. This field displays when you select New for the Generate to Web service Interface option. Type a different folder name if necessary.

Executing a web service operation using web services explorer

Web service operations or services can be executed using the Web Services Explorer. The user can view the requests, the parameters required for executing the requests, and view the responses after the request is executed. While executing the request, the user can also view the WSDL URL of a Web service operation.

To execute a web service operation:

1. In the Address field of the browser window, type: `http://<Computer Name>:<port number>/cordys/services`.
If the port number specified in the Address field is the default HTTP port, then specifying the port number is optional.
2. Select **<Web service interface> > <Web service operation>**.
3. Click **Test** for the selected <Web service operation>. The WSDL URL is displayed in the right frame and the corresponding request is composed in the Request text box.
4. Select the <organization> from the Select Organization list.
5. Fill in the required parameters in the composed request and click **Send** to send the request to the appropriate service group for executing it.
The response is displayed in the Response area.

The Web service operation is executed and the response is displayed.

Generating Web Service operations on database metadata

You can generate standard Web service operations directly on the database metadata. This option is useful when you need standard Web service operations to perform specific operations on the database tables without applying any custom logic. You must create database metadata and generate Web service operations on it. See *Creating database metadata* in the *AppWorks Platform Advanced Development Guide*.

1. Select one of the starting points:
 - In the Workspace Documents (My Recent Documents) window, click  (Database Metadata), and then select **Actions > Generate Web service Operations**.
 - Open **Workspace Documents (Explorer)** > **<solution>** > **<project>**, right-click **<database metadata>**, and then select **Generate Web service Operations**.

The Web Service Generation wizard window opens. The name of the Database Metadata is displayed on the title bar.
2. Enter the necessary details to generate the Web service operations, and then click **Finish**. See [Generate Web Service Operations on Database Metadata interface](#).

The Web service containing the Web service interface and Web service operations is generated and placed under the specified project/folder.

After you complete this task:

You must publish the generated Web service and its contents to the organization. See [Publishing a document to an organization](#). For use at runtime, attach the Web service interface to the WS-AppServer service. See [Attaching the web service interface to the WS-AppServer service](#).

Generate Web Service Operations on Database Metadata interface

The Generate Web Service Operations on Database Metadata interface opens when you generate a Web service on database metadata. This interface contains the following fields.

Field	Description
Web Service Name	Name of the new Web service. By default, the field displays the name of the database metadata as part of the Web service name. Type a different Web service name if necessary.
Web Service Interface Name	Name of the Web Service interface that contains all the Web service operations.

Field	Description
	Type a name for the Web Service Interface. If you selected the Generate One Web service Interface per Table/View check box, this field name changes to Prefix for Web Service Interface Name. In that case, type a prefix to attach to each Web service interface name that is generated.
Web Service Folder Name	Location where the Web service is created. The default folder is the current folder where the database metadata exists. Click  (Lookup) to browse and select a project or folder to place the Web service.
Views	Option to view database objects in a view format.
Tables	Option to view database objects in a tabular format.
Namespace	Namespace that is unique to the Web service. Type a different namespace name if necessary.
Generate One Web service Interface per Table/View	Option to generate a separate Web service interface for each table or view and display it in the Web Service Interface Name column. This option is selected by default. Clear the check box to generate a single Web service interface for all the tables. The Web Service Interface Name column is no longer visible.
Database Table/View	Tables or views in the database that display based on the earlier selection. Select the topmost check box to select all the tables or views. Otherwise, scroll through the list and select specific tables or views. For each table or view, you can generate different types of operations: <ul style="list-style-type: none">■ Pagination Operations: Generates the previous and next operations.■ CUD Operations: Generates the update operations (create, update, and delete) for each selected table.■ Referential Data Retrieval Operations: Generates operations that govern the relations between selected tables. For each table or view, select the corresponding check box in these columns to generate the required operations.

Working with external web services

Web services are self-contained, modular applications that can be invoked over the Web. These services range across different businesses, partners, customers and so on catering to various business activities. A Web service that is exterior to AppWorks Platform environment is referred as an external Web service. These Web services enable you to exchange services on the Internet. They are created and published by AppWorks Platform and consumed by external parties or vice versa.

The service could be as simple as World Time Service that gives the time for a city, a credit card transaction, or a full blown business service. A complex service would include other services within.

Generally, a Web service like the World Time Service is created, exposed on the web and maintained by a company. Such companies are called the Providers. A partner or a customer would use the service to find time for different cities as required. These users are called Consumers. The consumers can integrate these services into their business services. A web based travel booking service can use the World Time Service.

A consumer can use an exposed Web service only when he understands how to invoke it. Every Web service needs a contract between the provider and consumer. In the World Time Service the provider has to mention that the consumer has to send a city name and the country name. The provider would also mention what the consumer would receive in return, which is here the time. The provider may mention that the format of the time is 24 hours with day light savings. This information is shared in the form of a definition called WSDL. The interface definition clearly states what the provider wants and what it returns. The interface also contains other information related to invocation. Sending a city's name asking for its time is the Request and returning the time is the Response. Thus, a service sends responses to all the requests it receives from consumers. The request and the response are Messages.

To define a standard format for the messages that flow between requests and Web services, Web services use the extensible markup language (XML) standard. XML is an industry-standard, platform-independent syntax for describing and structuring data.

Universal Description, Discovery, and Integration (UDDI) is an XML and SOAP-based specification for publishing and locating information about Web services. UDDI provides a framework that enables business entities to describe and classify their services, and provide the technical details about the interfaces of their Web services. It also allows organizations to search for and locate required Web services.

AppWorks Platform provides a Web service wizard that allows you to generate Web service operations for the external Web services.

This section contains the following topics:

- [Searching for External Web services](#)
- [Generating AppWorks Platform Web service Operations for External Web services](#)
- [Configuring Access to External Web services](#)

- [Personal External Access](#)
- [Publishing Web services to an External UDDI Registry](#)
- External Web services over SSL/TLS

To access External Web services, set the following property in the browser:

1. Navigate to **Tools > Internet Options > Security > Internet** (or **Local Intranet**) > **Custom level > Miscellaneous**.
2. Set Access data sources across domains option to **Enable or Prompt**.
3. Select Internet or Local Intranet based on the way you are accessing AppWorks Platform.

For information about intercepted SOAP requests, see the *AppWorks Platform API Guide*.

Searching for external web services

You can search business entities and services in a UDDI registry. External Web services that are located in this manner can be used by AppWorks Platform, by generating Web service operations for them.

To search for external web services:

1. **Select a starting point** and click  (Web Service).
The Untitled Web Service - Web Service window opens.
2. In **Select the source**, select **Import WSDL**.
3. Type the appropriate **Name** and **Description**.
4. Click **Next**.
The Web Services Generation Wizard page opens.
5. In this page, type the WSDL URL to locate and read the WSDL in the URL field and click **Show Services**.
6. To search for the URL in the UDDI registry, click  (Lookup).
The Explore Web Services - UDDI Browser dialog box opens.
7. Select a <registry> whose inquiry URL you want to access from the From Registry list.
8. Type the keyword or the search term that is related to the URL you are looking for in the Keyword field.

You can refine your search by using following wild card characters:

- Use percentage (%) to search for a string of characters (Example: get% returns all the services starting with get).
 - Use the question mark (?) to search for a single character (For example: Por? returns all the 4 lettered words containing "Por").
8. Choose one of the following options for the **Find** field:

- If you want to search Web services by business entity name, select **Business Entities**.
 - If you want to search Web services by providing business service details, select **Business Services**. Provide the unique identifier of the Web service in the Business Key field. Refine your search for Web services using the [Advanced Search Options](#).
9. Click **Search**.
- Results are displayed in the Search Results tab of the Explore Web Services - UDDI Browser page. Depending on your selection in the previous step, the <business entity> or <business service> information is displayed as a table in the left pane.
- If you search for a business service, the entity name to which the service belongs is shown as a table caption above the Business Service table.
10. Select a Business Entity and the corresponding services are displayed in the right pane.
11. Select a Service and the corresponding operations are displayed in the bottom pane. The Operations pane denotes the Web service operation name, model key, and End Point URL.
12. Select a Web service operation and click **Select Service**.

The dialog box closes and the corresponding End Point URL is ported to the URL field of the Generating Web Service Operations page.

Generating AppWorks Platform web service operations for external web services

Web services are extensively used in the AppWorks Platform. Web services are standards-based and have a well-defined contract; they play an important role in modeling the application in the design time and provide a simple way to invoke them in the runtime.

Web services can be generated not only on standard AppWorks Platform models such as BPM or Database tables but also on external Web services to make them available on the AppWorks Platform.

The following procedure explains how to use the wizard to generate Web services on top of external services.

Before you begin:

- The standard application AppWorks Platform CWS External Web Service Model must be installed.
- Create the UDDI Service Container and ensure that it is running.
- [Create a project](#) in CWS.

To generate web service operations for external web services:

1. [Access Web Service](#) (Web Service) to generate Web service operations. The Web services Generation Wizard opens, displaying Untitled Web Service - Web Service page.

2. From the **Select the Source** list, select **Import WSDL**.
3. Type the appropriate **Name** and **Description** and click **Next**.
The next page of the Web services Generation Wizard opens.
4. Type the URL that points to the WSDL in the URL field (for example, <https://secure.jajah.com/api/SMSService.asmx?wsdl>).
5. If you want to use an external Web service from AppWorks Platform Process Factory (CPF), use the following procedure to obtain its URL:
 - a. Open the MashApps Composer window.
 - b. Select **Build Web services > All WebServices**.
It lists all available Web services.
 - c. Double-click the required Web service to view its details.
 - d. Copy the URL provided in the WSDL URL field.
5. If reading the WSDL requires authentication, select **Authentication Details** and type the credentials.
These login details will allow you to read the WSDL.
6. Click **Show Services**.
7. To [search the business entity or business service in the UDDI registry](#), click  (Lookup).
 - If the WSDL XML is invalid or returning content with invalid Content-Type property, the notification message "WSDL/Schema document is invalid or not present. Make sure the server is returning the content in proper format" is displayed.
 - If a request is unsuccessful, the message "Could not read the HTTP URL. The server has returned HTTP Error <code>" is displayed at times.
 - To access an external Web service from the local intranet zone, you must change the security settings in your browser.

7. Select the required WSDL type from the WSDL Type list.

The WSDL types are as follows:

Implementation	You can invoke the Web service operations directly, because the implementation is readily available for each operation. Hence, for these WSDLs, you can select the required Web service operations and proceed with the generation of Web services.
Interface	<p>In this type of WSDLs, the Web service operations do not have an implementation. Hence, you must implement all the Web service operations on the AppWorks Platform.</p> <ul style="list-style-type: none">■ To implement Web service operations, you need to select Contract in the Properties sheet, while defining the properties of the business process model.■ To generate Web service operations for this type, the operations

	under all the Port Types are selected by default and you must generate operations for all of them.
--	--

8. If invoking WSDL Web service operations requires authentication, do the following:
 - a. Select **Requires authentication to invoke**.
 - b. Provide the following details:

Type	Select the authentication type, Basic, Digest, NTLM, or SAML 1.1/Basic. In case of Basic authentication, the server authorizes a request only if it can validate the user name and password for that request. Digest authentication is similar to Basic but the difference is that the password is not sent in clear text. NTLM is an authentication protocol used in various Microsoft network protocol implementations. The SAML 1.1/Basic authentication combines Basic authentication and SAML 1.1. SAML is the Security Assertion Markup Language open standard as defined by OASIS . For Cordys Process Factory (CPF) Web Services, SAML 1.1/Basic is the authentication mechanism to be used.
Username	Type the username.
Password	Type the password.

9. From **Web Services**, select a **Port Type**.

The Web Service Operations defined in the WSDL for the selected port type are displayed in the Web Service Operations list. Namespace is automatically picked up from the WSDL.

9. Click **Finish**.

The Web service is listed in the project menu.

- If the imported Web service definition sets exist in the workspace, they will be updated with the latest content. If they are not found in the workspace, the newly created imported definition sets will be stored in the imported wsdls folder under the project.
- If the imported schemas exist in the workspace, they will be updated with the latest content. If they are not found in the workspace, the newly created schemas will be stored in the imported schemas folder under the project.
- However, there will be no change in the view of the definition set hierarchy as the imported schemas and WSDLs are displayed under the definition set.

11. Attach a service group to a Web service interface.

- If you do not specify the organization, the Web service gets published to the default organization set for the authenticated user.
- You can only consume WS-I Basic Profile compliant Web services.

Caution: It is recommended not to use the Web service Generator for generating Web services for internal (AppWorks Platform) Web services. This can cause a namespace to be attached to two different Service Groups (example LDAP and UDDI), which may result in request being routed to the wrong Service Group.

After you complete this task:

- [Publish](#) the Web service.

Configuring access to external web services

External Services Configuration enables you to specify which credentials to use for invoking the external web service and change the service endpoint if required.

Before you begin:

- Deploy the UDDI Application.
- Create the UDDI Service Container.
- Be sure that you have the role of UDDI Administrator.

To configure access to external web services:

1. On the Welcome page > My Applications, click  (External Services Configuration). The External Services Configuration window opens and lists all the Web service interfaces with UDDI, as their implementation class type.
2. From the **Web Service Interface** list, select a Web service interface. The Endpoints, Operations, Roles, and Properties appear in their respective panes.
3. In the **Roles** pane, select a role for which credentials are to be specified. Only functional [roles](#) are displayed. Related properties are displayed in the Properties pane.
4. In the Properties pane, select **Authentication Type**.
5. Assign credentials in the **User Name** and **Password** fields.

The authentication types supported are Basic, Digest, NTLM, or SAML 1.1/Basic. In case of Basic authentication, the server authorizes a request only if it can validate the user name and password for that request.

Digest authentication is similar to Basic but the difference is that the password is not sent in clear text. NTLM is an authentication protocol used in various Microsoft network protocol implementations.

The SAML 1.1/Basic authentication combines Basic authentication and SAML 1.1. This is typically used when integrating with the AppWorks Platform Process Factory. SAML is the Security Assertion Markup Language open standard as defined by [OASIS](#).

With this the UDDI service container will handle the retrieval and refresh of SAML assertions from the remote system. SAML 1.1 is used to authenticate at the remote system, this is done with the provided username and password here or in the [Personal External Access](#).

- For AppWorks Platform Process Factory (CPF) Web Services, SAML 1.1/Basic is the authentication mechanism to be used.
- The SAML 1.1/Basic authentication protocol implementation is not generically usable for integrations using SAML 1.1.

Because credentials are specified at role level, different roles can use different credentials.

You need not select a role to modify the service URL implying that the service endpoint change is not at role level.

5. Click .

If you intend to edit the XML directly instead of using the UI:

- Click **Edit XML** and make the necessary changes.

To apply a customization (for an Endpoint URL):

- Select **Apply to All Interfaces** to apply a customization (made for an Endpoint URL) to all the Web service interfaces that point to the same Endpoint URL.
This will avoid the need of customizing every Web service interface.

Attaching service group to a web service interface

This topic describes the procedure to attach a Service Group to a Web service interface.

Before you begin:

- [Create a Project](#).

1. In the Explorer view of the Workspace Documents window, go to **<Project> > <Web service Definition Set> > <Web service interface>**.
For information about changing the view in Workspace Documents window, see [Working with Workspace Documents](#).
2. Right-click the Web service interface and select **Open**.
The <Web service interface>-Web service Binding page appears.
3. Select a Service Group and click .

Publishing web services to an external UDDI registry

The UDDI registry provides a mechanism to advertise and discover Web services. For a service requestor to discover a service, a service provider must first publish a business entity and at least one business service in a UDDI registry.

Before you begin:

- Load the UDDI Application.
- Create the UDDI Service Container.

- Be sure that you have the role of either UDDI Developer or UDDI Admin to perform this task.
- Add the UDDI 2.0 registry. For more information on this task, see [Managing Registries](#).

To publish web services to an external UDDI registry:

1. Search for the Web service interface or operation you intend to publish.
The relevant Web service operations are displayed in the Search Results box.
 2. Right-click a Web service interface or operation and select **Properties**.
The Web Service Interface Properties - <Name of the Web Service Interface> dialog box opens. This box displays the Web service interfaces and operations in their corresponding App palettes.
 3. To publish to a Web service interface, in the Web Service Interface App Palette, click  and select Publish to UDDI Registry.
The Publish to UDDI Registry dialog box opens.
 4. To publish to a Web service operation, in the Web Service Operation App Palette, right-click an operation and select **Publish to UDDI Registry**.
The Publish to UDDI Registry dialog box opens.
 5. Select the registry where you want to publish the business entity.
You can select a registry through one of the following options:
 - **Default Registry** - Displays the registry that is chosen as default while Managing Registries. In this case, the list is disabled and the default registry is populated.
 - **Select a Registry** - Allows you to select a registry other than the default one from the list.Only the registries that have been added to UDDI registry are displayed in the Registry list.
5. To search for an existing entity:
 - a. Click **Search**.
The Search for entities and services dialog box opens.
 - b. In Find, select **Business Entities**.
 - c. Type the Keyword and specify the number of results to be displayed per page.
 - d. Click **Search**.
The results are displayed in the Search Results section.
 - e. Select the required entities and click **Use for Publish**.
 2. To create a new entity for the selected registry:
 - a. Select <create a new entity> from **Entity**.
Fields that have to be filled with the new entity information are displayed in the right pane. The registry hierarchy is displayed in the left pane.
 - b. You can enter information for the new entity.

A new service can be created within this new entity by selecting the New Service node appended to New Entity in the registry hierarchy.

6. You can select a business service in the following ways from the Service list.

Important: If you have selected an existing entity in the previous step, you can either create a new service under it or search for an existing service. While, if you have created a new entity, a new service can be created.

To select an existing service.	<ol style="list-style-type: none"> 1. Click Search. The Search for entities and services dialog box opens. 2. In Find, select Business Services. 3. Type the Keyword and specify the number of results to be displayed per page. 4. Click Search. The results are displayed in the Search Results section. 5. Select the required services and click Use for Publish.
To publish a new service for the selected registry.	<ol style="list-style-type: none"> 1. Select <create a new service> from Service. Empty fields to be filled with new service information are displayed in the right pane. 2. In the left pane, the New Service node is appended to the selected entity in the registry hierarchy. 3. You can enter information for the new service in the respective fields. However, this is not mandatory.

7. Click **Publish**.

The Web service is published to an external UDDI registry.

You can update tModel information as follows:

1. From the hierarchy in the left pane, select <registry> > <entity> > <service> > <binding template> > tModel.
2. Modify the required tModel information.

You can update binding template information as follows:

1. From the registry hierarchy in the left pane, select <registry> > <entity> > <service> > <binding template>.
2. Modify the required binding template information.

You can modify the entity, service, tmodel and binding information for a published Web service. For information on this, see [Modifying a Published Web service](#).

Managing registries using the UDDI Registry Manager

A UDDI Registry is a repository of business entities and services. To search for the available Web services, you must configure a registry. Using UDDI Registry Manager, you can add, modify, and validate registries.

Before you begin:

- Be sure that you have the role of a UDDI Administrator.
- Create the UDDI Service Container and ensure that it is running.

To manage registries using the UDDI registry manager:

1. On the Welcome page > My Applications, click  (UDDI Registry Manager).
The UDDI Registry Manager window opens.
2. You can perform one of the following tasks:

Add the registration details for a new UDDI registry. When the UDDI Registries are created, the registry names cannot be modified.	<ol style="list-style-type: none">1. Click  (Add). The Add New Registry Details dialog box opens.2. Enter the details in the Add New Registry Details dialog box and click  (Save). <p>The Inquiry URL and Publish URL must belong to the same UDDI registry.</p> <ol style="list-style-type: none">1. Click the row containing the registry name. The registration details for the selected registry are displayed.2. Modify the required information and click  (Save).
Delete the registration details of a UDDI registry from AppWorks Platform.	<ol style="list-style-type: none">1. Select the check box next to the name of the registry you want to delete.2. Click  (Delete).
Validate if the registry is up and running.	<ol style="list-style-type: none">1. Select the check box next to the name of the registry you want to test.2. Click  (Validate). <p>You cannot validate a registry, if it does not contain UDDI Service Group.</p>

Modifying a published web service

You can update the entity and service details of published Web services. You can also modify tModel and binding template information.

Before you begin:

- You must have loaded the UDDI Application.
- You must have created the UDDI Service Container.
- You must have either the UDDI Developer or the UDDI Admin role to perform this task.
- You must add the UDDI 2.0 registry, you want to use for publishing Web services to AppWorks Platform. For more information on this task, see [Managing Registries](#).

To modify a published web service:

1. On the Welcome page > My Applications, click  (Publish Services).
The Publish Services window opens.
2. Perform one or a combination of the following actions:

Modify the Entity details	<ol style="list-style-type: none"> 1. In the Registry list, select the registry whose entity details you want to modify. You can either continue with the Default Registry or Select a Registry. 2. Click Search. The Search for entities and services dialog box opens. 3. In Find, select Business Entities. 4. Type the Keyword and specify the number of results to be displayed per page. 5. Click Search. The results are displayed in the Search Results section. 6. Select the required entities and click Use for Edit. Fields with the entity information are displayed in the right pane. The registry hierarchy is displayed in the left pane. 7. Modify the <entity> details and click Save Entity.
Modify the Service details	<p>Select the registry whose entity details you want to modify from the Registry list. You can either continue with the Default Registry or Select a Registry.</p> <ol style="list-style-type: none"> 1. Click Search. The Search for entities and services dialog box opens. 2. From the Find list, Select Business Services. 3. Type the Keyword and specify the number of results to be displayed per page. 4. Click Search. The results are displayed in the Search Results section. 5. Select the required services and click Use for Edit. Fields with the entity information are displayed in the right

	<p>pane. The registry hierarchy is displayed in the left pane.</p> <p>6. Modify the <service> details and click Save Service.</p> <p>You can enter information for the <service> by selecting the '<service>' node appended to <entity> in the registry hierarchy.</p> <p>1. From the registry hierarchy in the left pane, select <registry> > <entity> > <service> > <binding template> > tModel.</p> <p>2. Modify the required tModel information and click Save tmodel.</p> <p>1. From the registry hierarchy in the left pane, select <registry> > <entity> > <service> > <binding template>.</p> <p>2. Modify the required binding template information and click Save binding.</p> <p>See Binding Template Details Interface.</p>
Update tModel information	
Update binding information	

Personal external access

Personal External Access enables the end user to set the user name and password to be used by the UDDI connector when trying to invoke an external web service on behalf of the user.

Before you begin:

- All OpenText AppWorks Platform Users have access to this task.

To specify user specific credentials, do one of the following:

- Use the Personal External Access UI
 - Specify the service end point URL, user name and password
- Use the Credentials dialog box
 - This dialog box opens when the user is working with the application and the external web service invocation fails because of invalid or missing credentials.
 - Specify the user name and the password for the service end point URL.
 - The AppWorks Platform framework will resend the request.

Notes:

- Credentials specified using the External Services Configuration by the administrator take precedence over credentials specified by the user using Personal External Access UI.
- Credentials dialog box opens only when the administrator has not specified any credentials for the user's role
- Credentials dialog box opens only when Authentication type is SAML 1.1./Basic.

Viewing the WSDL of a web service operation

WSDL is the foundation for creating a Web service operation. As a developer, you must know the WSDL based on which you are developing the Web service operation. By reading the WSDL, you will know whether the Web service operation that you created complies with the communication standards that are laid down in the WSDL. It helps you in diagnosing a problem, if any.

After checking the WSDL, you can [test](#) the Web service operation before using it at run time.

Before you begin:

- Create the Web service operation.

To view the WSDL of a web service operation:

1. In Workspace Documents (Explorer), open <solution> > <project> >  >  , right-click  (<Web service operation>) and select Show WSDL.
The WSDL window opens, displaying the Web service definition in XML.
2. Read the WSDL for communication ports, Web service location, input and output SOAP messages, binding, and operation.
3. Close the WSDL window.

Updating web services

After you generate a Web service, you may feel the need to enhance its functionality by adding more Web service operations to it or by updating the existing Web service operations. This can be achieved through the Web service generation wizard.

Caution: When you edit a Web service, ensure that the target namespace for all the Web service interfaces that you add or update remains the same.

To update web services:

1. Select one of the following starting points:
 - In Workspace Documents (My Recent Documents), point to  (Web Service), click  and select Actions > Add/Update Operations.
 - In Workspace Documents (Explorer), open <solution> > <project>, right-click  (<Web service>) and select Add/Update Operations.
The Web Service Generation wizard opens, displaying the name of the <Web service> on its title bar.
2. On the wizard, from the **Select the source** list, select the type of Web service operation you want to generate.
For details on each type that is listed, see [Working with Web services](#).

3. If you want to add or update Web service operations to the existing Web service interface, select **Overwrite to existing Web service interface** and select the relevant Web service interface from the Select a Web service interface list. Leaving the check box cleared creates a new Web service interface under the same Web service.
4. Click **Next** and provide relevant information depending upon the type of Web service operation that you intend to update/generate.
5. You need to enter details on several pages. Click **Next** on each page to move to the subsequent one and continue until you complete entering information on the wizard.
6. Click **Finish**.

The Web service Generation wizard closes.

The existing Web service operations are generated or new Web service operations are generated and added to the Web service.

After you complete this task:

- You must [publish](#) the generated Web service operations to the organization and attach them to the related Service so that they can be used at run time.

Testing web service operations

Testing Web service operations helps you to know whether a Web service operation works properly or not. It is better to check the functionality of a Web service operation before using it in run time.

Testing is done using a SOAP request. It checks whether the Web service operation generates correct SOAP request and response, and properly tries to send and receive SOAP messages. The feature facilitates composing and editing the SOAP request, if required.

Before you begin:

- Create and publish the Web service operations.

To test a web service operation:

1. In Workspace Documents (Explorer), open <solution> > <project> >  > , right-click  (<Web service operation>) and select Test Web Service Operation. The Operation Test Tool window opens, displaying the SOAP request for that Web service operation.
2. Depending upon the data that the request needs to process, enter the values replacing PARAMETER in the SOAP request.
3. Click **Invoke**.
The request and response appear in the Result Messages area.
4. Click the response message to check if the retrieved details are accurate.

5. Click **Clear Messages** to clear the request and response messages from the Result Messages area.
6. Repeat the test with modified parameters, if required.

Setting access control on web service interface and web service operations

In your application, you may want to restrict the accessibility of some Web service interfaces or Web service operations to certain roles. You can restrict or grant access to the Web service interface and Web service operations while developing the application. The Security feature helps you set access control on a Web service interface or Web service operation at design time, where you can determine the accessibility options that suit your application's design.

Access control considers the hierarchy between Web service interface and Web service operations and this information is passed on to the Access Control Engine that imposes the defined access control. Access control set on a Web service interface extends to all the Web service operations under it, but access control set on a Web service operation does not apply to the Web service interface. Therefore, you can provide the required permissions to a role to access the entire Web service interface and at the same time restrict the role from executing a particular Web service operation.

Before you begin:

- Generate Web service interface and the Web service operations.
- Create roles and publish them to the organization.

To set access control:

1. Select one of the following options:
 - In Workspace Documents (Explorer), open <solution> > <project> >  (Web service interface), right-click  (Web service interface), and select Define Runtime Security.
 - In Workspace Documents (Explorer), open <solution> > <project> >  >  (Web service operation), right-click  (Web service operation), and select Define Runtime Security.

The Security Editor window opens, displaying the name of the Web service interface or the Web service operation on its title bar.
2. In the Roles pane, click .

The Select Role dialog box opens with the roles that you created.

3. Select the role for which you want to set access control.

The selected role is shown in the Roles pane.

4. In the Permissions pane, select **Execute**.

If the Execute check box is not selected, the role will not have access to that Web

service operation. It means, users possessing that role will not be able to perform a task driven by that Web service operation.

5. Click .

Access control set on that Web service interface or Web service operation is associated with a role.

After you complete this task:

- After you set access control on a Web service interface or Web service operation, publish it to run time. To do that, you need to have the role of an Administrator or have administrative privileges.

Generating a web service operation on an email model

You can generate web service operations on an E-mail Model using the Web service definition wizard and use them in a Business Process Model.

In the run time, Web service Interfaces are referred to as Method Sets and Web service Operations are referred as Methods.

Before you begin:

- Create an E-mail Model to generate a Web service Operation.

To generate a web service operation on an email model:

1. From the Workspace Documents (Explorer), right-click your project and select **New > Other**.
The New AppWorks Platform Document window opens.
2. Click  (Web Service).
The Untitled Web Service - Web Service wizard opens.
3. Click **Select the source** and select **Email Model** from the list.
4. Type the **Name** and **Description**.
5. Type the namespace for the Web service operation and click **Next**.
6. Click  (Lookup) next to Select an Email Model.
The Select Email Model dialog box opens.
7. Select the required Email Model and click **OK**.
The name of the selected Email Model is displayed in Select a Email Model.
8. Provide the names of the Web service Interface and Web service Operation and click **Finish**

A Web service Properties page opens, displaying the target namespace, its prefix and the location URI in the Target Namespace, TNS Prefix, and Location URI fields.

The E-mail Model can now be used as a Web service operation and is displayed under Workspace Documents (Explorer) > <project> folder.

After you complete this task:

- Right-click the E-mail Model <Web service operation> in Workspace Documents (Explorer) and click **Publish to Organization** to make it available to an organization.
- To work with Web services generated on E-mail Models, the Notification Service Group needs to be available in that organization.
- If you encounter the 'Service Group Lookup Failure' error on executing a Web service generated on the E-mail Model, you can do one of the following:
 - Add the corresponding Method Set (Web Service Interface) manually to the Notification Service Group.
 - Alternatively, you can [attach the Notification Service Group to the Web service Interface](#) in the project.

Viewing inline schema of web services

Inline schemas are the schemas in which you can include the schema within a WSDL file rather importing the schema.

Before you begin:

- Create the Web service operation.

To show inline schemas:

1. In Workspace Documents (Explorer), open <solution> > <project> >  > The Schemas folder displays the list of Inline Schemas in WSDL.
2. Expand the required schema to view the Schema fragments and the XSD References.

To hide inline schemas:

- In Workspace Documents (Explorer), open <solution> > <project> >  > The Schemas folder does not display the Inline Schemas of WSDL. It contains the Schemas that have a separate schema location.

Hide Inline Schema is available only for the schemas in which the previous setting is **Show Inline Schema**.

Chapter 17

Working with web applications

A Web application constitutes a set of Web pages bound together by the business logic defined for it. This section focuses on developing the interfaces for your application.

A Web page is an interface that is used to capture user information in an application. Given the possibilities of today's Web environment, you can create Web pages that display and process information combined from varied sources. To cater to such requirements, AppWorks Platform provides support for creating application interfaces using URLs of existing JSP or HTML pages that you may already have and that are external to the AppWorks Platform.

AppWorks Platform also supports the creation of user interfaces based on XForms. It provides an XForms Designer that offers advanced features to design and create XForms-based user interfaces. These user interfaces are also referred to as XForms.

The following topics describe the functionality available to create Web pages in AppWorks Platform.

- [Creating an Interface Using a Web Page URL](#)
- [Creating XForms](#)
- [Managing XForms](#)
- [Working with XForms](#)
- [Customizing Interface Styles](#)
- [Managing XForms Translation](#)
- [Adding a Workflow Library to HTML Applications](#)
- [Working with Bi-directional Support in User interfaces](#)
- [Accessing AppWorks Platform with a specific Locale](#)
- [Working with HTML5 SDK](#)
- [Organization Level Styling and BASE Relative URLs](#)

Working with XForms

After you create an XForm, you may need to customize it according to your specific requirements. This section contains topics that describe the features available to you in AppWorks Platform XForms.

Depending upon applicability, information is grouped as follows:

- [Customizing Controls](#): for information about how to modify a control.
- [Customizing XForms](#): for information about how to modify an XForm.
- [Working with Data](#): for information about how to use various data models in an XForm.
- [Working with Composite Controls](#): for information about how to create composite controls.
- [Integrating with WS-AppServer](#): for information about how to integrate an XForm with WS-AppServer.
- [Reusing XForms](#): for information about using an XForm as a template.
- [Interceptor Support](#): for information about how to manipulate the generated HTML content that goes to the client.

Note: For the XForms that are created using Cordys BOP 4.2 CU1 or later versions, the inter-version support is available. However, you can continue to use your existing XForms in the current AppWorks Platform version.

Creating a user interface using a web page URL

User interfaces built for your applications in AppWorks Platform can be accessed from the Welcome page. By creating an interface for a Web-based application, you can define the authorization or access level for a user on that application. This access control setting enables or disables activities for that user. You can create an interface for external applications such as HTML and JSP.

Before you begin:

- [Create a project](#) in CWS.

To create a user interface using a web page URL:

1. Select a starting point and click  (External User Interface) to create a User Interface using Web pages.
The Untitled External User Interface - External User Interface window opens.
 2. Provide the **Name** and **Description** of the interface.
 3. Specify the [relevant details](#).
 4. Click  (UI Task Properties) to configure the [properties](#) of the external user interface.
 5. To create a delivery model, click  on the toolbar, and click  (Add) in the Children - Untitled External User Interface pane.
For more information on the procedure to create a delivery model, see [Creating a Delivery Model](#).
 5. Click .
- A user interface is created for the intended Web page. You can view it in the Workspace Documents window.

AppWorks Platform relies on the native authentication mechanism, if any, of the external user interface.

6. You must [publish](#) the user interface to deploy it in the current organization. The user interface is published and is available as a task for the runtime assignment.
7. Assign this task to the user.
The user interface is displayed as a task on the My Applications App Palette.

A user interface is created using the URL of a web page.

To delete the external user interface, do any of the following:

- In the My Recent Documents view - Mouse over the user interface and click  (Extend Menu) and then select Delete on the context menu.
- In the Explorer view - right-click the user interface and select **Delete**.

Creating XForms

You can use AppWorks Platform XForms to build the user interface of your application. AppWorks Platform XForms offers varied functionality with which you can easily create application interfaces using the [AppWorks Platform Web Service Operations](#) that are available with the application. You can also use Java API to [create XForms dynamically](#) or use [XML Schema Fragments to create XForms](#) and make them available for mapping in the message map.

All XForms, once created, automatically participate in application level logging, as part of the composite application logging functionality. To enable application level logging, an XForm is first registered to the logging framework. The complete path of the XForm is used for registration. For example, to register file `file1.caf` the complete path used can be `/home/<organization name>/folder1/folder2/file1.caf`, which includes the XForm name.

In case of errors involving an XForm, a log is saved along with corresponding registration information of the XForm. This makes it easier for an administrator to search for the log related to an XForm. You can identify the log by the XForm name in the complete path.

You can use the related references provided in this topic for conceptual information about AppWorks Platform XForms.

Creating XForms using web services

AppWorks Platform supports the use of Web services to create XForms. A Web service is a simple, open-standard, and XML-based Web application that allows interoperability between various platforms. A Web service may contain multiple Web service operations to provide a wide range of services.

Before you begin:

- Ensure that the [Web service](#) to be used to create the XForm is published and available for use.

To create XForms using web services:

1. Select a starting point and click  (User Interface) to create XForms using Web Services.
The Untitled User Interface - User Interface App Palette is displayed and the XForms Designer opens.
2. Drag the appropriate Web service to the XForm from the **WebserviceBindingOperation** group in the Insert tab.
Alternately, right-click the Designer Area of the XForm, select Insert > Insert Web Service, and drag the Web service from the Web Services dialog box that opens. This right-click menu option is available for a split area, an XForm, and for the Group, Groupbox, and Tab Page grouping controls.
You can also use external Web services to create XForms. For information about using external Web services in AppWorks Platform, see [Working With External Web services](#). The Generate Input and Output UI dialog box opens.
3. Make appropriate modifications in the Generate Input and Output UI dialog box and click OK.
 - Select **Generate UI for Input message** to generate an input UI.
 - Select **Generate UI for Output message** to generate an output UI.

For details about generating UI, see [Generating Input and Output User Interfaces](#).
The UI is displayed in the Designer Area. You can modify the XForm as required.
4. Click .
The Untitled User Interface - User Interface dialog box opens.
5. Enter the **Name** of the XForm.
The name of the XForm must begin with a letter of the alphabet or an underscore, and can contain only letters, numbers, and underscores.
A name that begins with an underscore must contain at least one letter or number. The user interface name cannot contain multi-byte characters.
Caution: Ensure that the names that you specify are not the same as the names of the [predefined XForms](#) in the application.
6. Enter a description for the XForm in the Description field.
The description can contain only letters, numbers, and special characters such as periods (.), commas (,), parenthesis (()), single quotation marks ('), hyphens (-), and underscores (_).
7. In the Location field, select  (Lookup).
The Select Folder dialog box opens.
8. Specify the location where the XForm must be saved, and click **OK**.
The Select Folder dialog box closes.
9. Select **Save** in the Untitled User Interface - User Interface dialog box.
The Untitled User Interface - User Interface dialog box closes.

A new XForm is created and saved.

To use the preview feature to preview the XForm:

- Click  (Preview) in the toolbar.

Note: If the schema of a Web service that is being used in an XForm is changed, the change will not be automatically reflected in the XForm. In this case, you must again add the modified Web service to the XForm.

After you complete this task:

- After creating an XForm, you can customize it and translate it in various languages, as required. You can also publish it to use it in an application.

Creating XForms dynamically

In addition to using the XForms Designer, you can use Java API to create XForms. Using AppWorks Platform XForms, it is possible to view and dynamically generate the user interface of XForms from Java classes. You can use the API exposed by AppWorks Platform XForms to generate a .caf file for Java class.

To create XForms dynamically:

1. **Select a starting point** and click  (User Interface).
The Untitled User Interface - User Interface App Palette is displayed and the XForms Designer opens.
2. Click **Click here to create Dynamic XForm**.
3. Type the fully qualified class path of the implementation class in the Qualified Class Name field.
An example of the fully qualified class path is
`com.cordys.test.dynamic.DynamicXformGenerationone`. This specifies the Java class using which you can create an XForm.

The XForms service container loads the specified Java class dynamically, if the Java archive is present in the `<AppWorks_Platform_installdir>/components/xfruntime/dynamicxforms` folder. In this case, you do not need to restart the XForms service container. However, if the Java archive is not present in the mentioned folder, specify the location of the Java archive in the JRE Configuration of the XForms service container and restart the service container in order to load the Java class.

Important: To be able to use an implementation class to create a dynamic XForm, the IXFormsDefinition interface definition must be implemented for the class. This makes the implementation class eligible for generating dynamic XForms. You can use the `getXFormsDefinition` and `getWSDLDefinition` Web service operations to develop a dynamic XForm. The `getXFormsDefinition` Web service operation enables the acceptance of parameters from the URL. You can pass the parameters supported by the `getXFormsDefinition` and `getWSDLDefinition` Web service operations through the URL.

4. Click **OK**.

When you click **Cancel**, the Untitled User Interface - User Interface App Palette opens without saving the qualified class name you entered.

The XForm is automatically generated from the specified Java Class, and is displayed in the XForms Designer in the read-only mode. It is not possible to modify the XForm from the XForms Designer.

You can also view the properties of the dynamic XForm. The property sheet is displayed in the read-only mode and is not available for modifications. Additionally, after creating a dynamic XForm, it is possible to use the URL to pass parameters to the XForm (.caf file). These parameters, if any, are stored in a Hashtable with the parameter names as key. After the XForm is created, you can [publish](#) it to the organization and use it in a business process. You can also translate a dynamic XForm. Dynamic XForms are not cached.

It is possible to view the WSDL of a current dynamic XForm, if the `getWSDLDefinition` Web service operation is implemented. To view its WSDL, type the URL of the XForm followed by `?WSDL` as a suffix.

Example

The following example code demonstrates the implementation of a Java Class using `IXFormsDefinition` and the use of the `getXFormsDefinition` and `getWSDLDefinition` Web service operations.

Prerequisite: The following jars should be in CLASSPATH before compile the given sample java code.

- <AppWorks Platform_installdir>/cordyscp.jar
- <AppWorks Platform_installdir>/components/xfruntime/xfruntime.jar
- <AppWorks Platform_installdir>/components/xfruntime/dynamicxforms

```
package com.cordys.test.dynamic; import java.util.HashMap; import
com.eibus.xforms.XFormsDef; import
com.eibus.xforms.persistence.IXFormsDefinition; import
com.eibus.xml.nom.Document; public class DynamicXformGenerationone
implements IXFormsDefinition { public void genXForm(XFormsDef xf) { int
root = xf.get MainForm(); xf.setXFormsVersion("2"); xf.addInput(root,
"Employee", "empID", " "); xf.linkStyleSheet
("/cordys/wcp/style/MyCss.css"); } public int getXFormsDefinition( HashMap
parameters ) { Document document = NOMDocumentPool.getInstance
().lendDocument(); //This node must be either a valid Xforms definition or
zero. Framework will take care of this when it is zero. int
xformsDefinition = 0; XFormsDef xf = null; try { xf = new XFormsDef
(xformsDefinition); genXForm(xf); } finally { NOMDocumentPool.getInstance
().returnDocument(document); } return xf.getXFormsDefinition(); } public
int getWSDLDefinition( HashMap parameters ) { return 0; } }
```

Creating XForms using XML schema fragments

AppWorks Platform supports the use of Schema fragments to create user interfaces (for outputs) in XForms.

This is useful in cases where XForms are not associated with a WSDL, making it difficult to map them for outputs in a process. In such cases, you can add a schema fragment to XForms, making them visible in the message map for mapping. While generating an HTML definition, AppWorks Platform XForms interprets the schema for a business object and combines any validation defined for it. When you enter a value in the XForm at run time, the value is validated and an alert is displayed if the validation fails.

To create XForms using XML schema fragments:

1. Select a starting point and click  (User Interface).
The Untitled User Interface - User Interface App Palette is displayed and the XForms Designer opens.
2. Drag the appropriate schema fragment to the XForm from the Schema Fragment group in the Insert tab.
Alternately, right-click the Designer Area of the XForm, select **Insert > Insert Schema Fragment**, and drag the Schema Fragment from the Select Schema Fragment dialog box that opens. This right-click menu option is available for a split area, an XForm, and for the Group, Groupbox, and Tab Page grouping controls.
The Generate Output UI dialog box opens.
3. Select **Generate UI for Output** to generate an output UI.
The UI is displayed in the Designer Area. You can modify the XForm as required. A model is created for the schema fragment added to the XForm.
4. Click .
The Untitled User Interface - User Interface dialog box opens.
5. Type the name of the XForm in the Name field.
The name of the XForm must begin with a letter of the alphabet or an underscore, and can contain only letters, numbers, and underscores. A name that begins with an underscore must contain at least one letter or number. The user interface name cannot contain multibyte characters.
Ensure that the names that you specify are not the same as the names of the predefined XForms in the application.
6. Type a description for the XForm in the Description field.
The description can contain only letters, numbers, and special characters such as periods (.), commas (,), parenthesis (()), single quotation marks ('), hyphens (-), and underscores (_).
7. In the Location field, select  (Lookup).
The Select Folder dialog box opens.
8. Specify the location where the XForm must be saved in the Select Folder dialog box, and click **OK**.
The Select Folder dialog box closes.

9. Select **Save** in the Untitled User Interface - User Interface dialog box.
The Untitled User Interface - User Interface dialog box closes.
An XForm is created and saved.
10. To preview the XForm, click  (Preview) in the toolbar.

Note:

- If the schema of a schema fragment that is being used in an XForm is changed, the change will not be automatically reflected in the XForm. In this case, you must again add the modified schema fragment to the XForm.
- You can also add a schema fragment to an XForm using the Model Properties dialog box.
For details, see [Model Properties](#) dialog box.
- For information about creating schema fragments, see [Creating Schema Fragments](#).

After you complete this task:

- After creating an XForm, you can [customize](#) it and [translate](#) it into various languages, as required. You can also [publish](#) it to use it in an application.

Customizing controls

After creating an XForm, you can add controls to it and customize those controls to further enhance their functionality.

This section groups topics that describe the various ways in which you can [add controls to an XForm](#) and customize controls on an XForm. The topics in this section address the ways in which you can [position](#), [resize](#), [align](#), [anchor](#), and [convert](#) controls. You can also [align labels of grouping controls](#) on your XForm, [add element bars](#) and [set the properties](#) of a control; and [split a control or an XForm](#).

Additionally, you can:

- [Add Columns to Table Controls](#)
- [Bind Data to Controls](#)
- [Customize Element Bar and Toolbar Buttons](#)
- [Customize the Find Dialog Box](#)
- [Format and Validate Controls](#)
- [Hide Controls in Tables and Groupboxes](#)
- [Hide Groupbox Headers](#)
- [Insert Lookup Pages](#)
- [Position Labels](#)
- [Prefill List and Select Controls](#)
- [Set Synchronized Dialogs](#)
- [Specify Data Values for Check Controls](#)

- [Use Message Map](#)
- [Use Web Components in XForms](#)
- [Write Scripts for Events](#)

Adding controls to an XForm

While designing an XForm, you can add various controls to it to perform actions such as entering and displaying information.

To add controls to an XForm:

1. [Open the XForm](#) in the XForms Designer.
2. Drag the control from the Toolbox tab to add it.

The control is inserted and positioned depending on the layout of the XForm, and is set to focus.

The size of the added control depends on the size and [layout](#) of the parent control.

- In case of vertical layouts, the control occupies the full available width.
- For horizontal layouts, two controls are arranged in a row, by default.
- For Free layouts, the control is inserted at the position you drag it to and retains the default size.

You can resize the control manually, or can use the Set Default Size context-menu option to specify its default width.

For Vertical and Horizontal layouts, placeholders appear on the XForm to help you insert the control.

When a control is added to an XForm, an auto-generated ID is assigned to it. You can view this ID in the control's property sheet. AppWorks Platform XForms supports the copying of existing controls within an XForm and to other XForms. However, two controls in an XForm cannot have the same ID. If a control is copied to the same XForm, then as two controls in the XForm cannot have the same ID, the copied control is automatically assigned a new unique ID. When copied to another XForm, the ID of the copied control is retained or changed depending on the existing IDs of the controls in that XForm.

Adding columns to table controls

When added to an XForm, a Table control displays a column containing check boxes and an empty column, by default. You can modify the Table control and add additional columns as required.

To add columns to table controls:

1. Open the XForm in the XForms Designer.
2. Drag a Table control from the Toolbox tab to the open XForm.
A table is displayed in the Designer Area of the open XForm.
3. Drag the appropriate control from the Toolbox tab into the Table control.
A new column is created and the default control name appears as the header of the column.
 - A placeholder is shown to help you position the control.
 - You can drag the Input, Password, Output, Textarea, Check, Select, List, and Image controls to form columns.
4. To edit the name of the column, click it to view the name in the editable mode, type the new name, and press **Enter** or click outside the editable field.
The column header displays the new column name.

The new column with the modified header is added in the Table control. You can add multiple columns to the table and rename them as required.

Adding element bars to controls

You can use element bars to manage data in controls grouped under grouping controls, such as the Table, Groupbox, and Tab Page controls.

Element bar comprises a control bar and a pagination bar. Control bar comprises buttons used to add, delete, or refresh control data, while pagination bar comprises buttons used for searching and navigating between records.

To add element bars to controls:

1. Open the XForm in the XForms Designer.
2. Right-click a control in the XForm, select **Control bar**, and select the appropriate option from its sub-menu:

To add a button to the control bar	Select Add Item .
To add standard buttons to the control bar	To add standard buttons to the control bar such as  (Add),  (Delete), and  (Refresh), select Add Standard Items .
To delete a control bar	Select Delete control bar .

The control bar displays buttons depending on the option you select.

- It is possible to access the above options from the context menu of the control bar, after it is added to the control.
- You can also add the control bar to a control by selecting Control bar and Pagination bar in the Element bar options pane of the control's property sheet.

3. Right-click the control in the XForm, select **Pagination bar**, and select appropriate options from its sub-menu:

To add standard buttons to the pagination bar	To add standard buttons to the pagination bar such as (First), (Previous), (Find), (Next), and (Last), select Add Standard Items.
To add the (Show All) button to the pagination bar	Select Add 'Show All'. This option is available only for the Table control.
To delete it	Select Delete pagination bar . The pagination bar displays buttons depending on the option you select.

It is possible to access the options from the context menu of the pagination bar, after it is added to the control.

4. Click .

The changes made to the element bar of the control are saved. The element bar is added to the control.

Aligning controls

AppWorks Platform XForms provides the functionality to align multiple controls with respect to a control. You can thus define relative positions for specific controls in an XForm.

To align controls:

1. Open the [XForm](#) in the XForms Designer.
2. Select the controls to align.
3. Right-click the control to which the other selected controls need to be aligned, select **Align**, and select the appropriate option from the sub-menu:

Align Left	To align the selected controls to the left of the control.
Align Center	To vertically center-align the selected controls with the control.
Align Right	To align the selected controls to the right of the control.
Align Top	To align the selected controls to the top of the control.
Align Bottom	To align the selected controls to the bottom of the control.
Make Same Width > Labels >, or Both	To modify the width of labels, controls, or both for the selected controls with respect to the control.
Make Same Height	To modify the height of the selected controls with respect to the control.
Make Same Size	To modify the size of the selected controls with respect to the control.

Distribute Horizontal	To specify equal horizontal spacing between selected controls.
Distribute Vertical	To specify equal vertical spacing between the selected controls.

- For vertical and horizontal layouts, only the **Make Same Width**, **Make Same Height**, and **Make Same Size** options are available in the context menu.
- It is not possible to align multiple selected controls with respect to the Button, Image, Tab Page, or Radio controls.

Depending on the option you select, the selected controls are aligned with the specific control.

Aligning and resizing labels of grouped controls

While designing applications, you can use grouping controls, such as Groupbox, Group, and Tab Page, to group multiple controls.

Controls comprise a label that contains descriptive text and a user interface element that is used to input or display information. Depending on the text that is specified in the label of each control, the width of the labels varies for different controls. Due to this, the controls in the grouping control appear out-of-alignment and irregular. In such cases, it is possible to adjust and align the size of labels so that the controls appear properly aligned.

To align and resize labels of grouped controls:

1. [Open the XForm](#) in the XForms Designer.
 2. Select the controls for which the label size needs to be aligned.
 3. Position the mouse pointer on a control edge.
The resize handler appears, using which you can resize the labels.
 4. Drag the resize handler to the required position.
This resizes the width of all selected labels.
- Alternatively, select all controls that must be aligned, right-click a control and select **Align > Make Same Width > Labels** to modify the width of all labels according to the label of the control.
- The labels are resized and aligned with the selected label.

The extent of resizing of a label depends on the white space available in it. You cannot resize a label that does not contain white space beyond the minimum width of its textual content. Multi-worded text with white space will wrap when label width is reduced.

Anchoring Controls

Using the anchoring feature, you can specify absolute positions for freely positioned controls in an XForm.

To anchor controls:

1. [Open the XForm](#) in the XForms Designer.
2. Click  (Show Anchor Bars) on the XForms Designer toolbar.
Alternatively, right-click a control and select Anchors.

Note:

- The Show Anchor Bars icon is available only in the advanced toolbar of the XForms Designer. Check Advanced in the toolbar, to view it. This enables the anchor display mode.
- The anchor display mode is disabled when you first open an XForm in the XForms Designer. Subsequently, AppWorks Platform XForms saves your preferences, and the anchor display mode is enabled or disabled accordingly when you access the XForms Designer the next time.

3. Click a control.
The anchor bars of the control appear. Use the anchor bars to specify the position of the control with respect to the edges of the Designer area.
4. Place the cursor on the anchor bars of the control to display an editable field (containing a default value), and make appropriate modifications:
 - To anchor a control at an equal distance from all sides of the parent control, place the cursor on the anchor bar at the center of the control, and type the value and the unit of measurement in the editable field that is displayed.
 - To anchor a control to any side of the parent control, place the cursor on the corresponding anchor bar, and type the appropriate value in the editable field that is displayed.
 - To anchor a control such that it expands to fill the parent control, click the anchor bar at the center of the control.
 - To remove an anchor from a control, click the anchor bar to clear it.
The control is repositioned according to the options you specify.

5. Click .

The anchor settings for the XForm are saved.

Important:

- Open Text recommends that you use **%** as the anchor-size unit for controls that are anchored on all sides and are expected to resize with reference to the parent.
- If you use units other than % to specify the fixed anchor size for controls, when reducing the XForm size, the controls that are anchored on all sides may become too small to be viewed properly.
- Open Text recommends using **em** as the unit to specify the anchor size in terms of the text size used in the browser. This is suitable for scenarios where XForms are viewed in various resolutions.

Binding data to controls

You can associate controls with models to display data (parameter values of model's business objects) in an XForm.

To bind data to controls:

1. Open the [XForm](#) in the XForms Designer.
The XForm must be associated with at least one model to enable binding of data. For information on how to associate models with an XForm, see [Adding and Associating Multiple Models](#).
2. Drag a control from the Toolbox tab to the Designer Area to add it to the XForm.
3. Right-click the control and select **Properties**.
Alternatively, double-click the control.
The <control> window opens.
4. From **Model**, select an option to specify the model from which to bind data.
This option is not mandatory if a parent control is associated with a model. In such a case, the control is automatically associated to the parent control's model.
5. In References, click  (Lookup).
The Response - <model> - References dialog box opens, displaying the data nodes available for association in the specified model.
6. Select the data node to associate with the control and click **OK**.

Restriction: For non-transactional models and for transactional models returning complex XML data (containing multiple levels of nodes), you must manually specify the complete XPath in the References field. For information about referencing non-transactional data see, [Referencing Transactional and Non-transactional Data](#).

The selected data node is displayed in the References field.

7. Select an option from the Parent list to specify the parent view from which data is displayed in the control. This option is available only for the Table, Tab Page, Groupbox, and Group grouping controls. This step is optional. If you specify a parent view, the Reference XPath is evaluated on the parent view active business object. The reference is picked with respect to the specified parent view, instead of considering the complete XPath.
8. Click .

The settings specified for binding data to the control are saved.

Data bound to composite controls is displayed in its child controls, as relevant. However, data bound to primitive controls is displayed only in the specific control; for example, data bound to a Button control is displayed in it as text.

Creating a task part for a control

While working with AppWorks Platform, when you create a user interface, a corresponding task is automatically created for it. By [assigning](#) these tasks to user roles, you can exercise control over user access to functionality in an application.

AppWorks Platform XForms takes access control a step further by enabling the creation of task parts for each control in an XForm. In addition to the access control possible on a Web Page, you use these task parts to specify access for each control in a Web page. If at run time, you do not have permissions to access the task part, the associated control will be hidden or disabled.

To create a task part for a control:

1. Double-click the control to display its property sheet. Alternatively, right-click and select Properties.

The <control name> dialog box opens.

2. Select **Task Part**.

The Task Part Details pane opens.

3. Click  (Lookup) in the Associated Web Services field and select the Web services to associate with the control from the Set Web Services dialog box that opens.

The  (Lookup) button of the Associated Web Services field is disabled if no model with a Web service is present on the XForm. You must add a Web service to the XForm to associate it with a control in the XForm.

The Web service is displayed in the Associated Web Services field.

4. To specify the name of the task part, select **Name** or **Existing** and type the appropriate **Name**.

It is possible to specify the same task-part name for multiple controls. This groups the controls such that the configuration settings applicable to the task-part name apply to all grouped controls.

5. To specify the display mode of the control when its task part is not available for a user role, select **Hide** or **Disable**.

For example, on selecting Hide, the control is not displayed in the XForm. On selecting Disable, the control is not available for editing.

The task part is created for the control.

Creating ClickChoice relations

A major purpose of Web applications is the display of data to users. In their first interaction with users, most applications provide relevant information only to an optimal level of complexity. Additionally, navigation options are made available to users to enable them to delve further into additional detailed information.

The ClickChoice feature of AppWorks Platform XForms provides you a similar functionality. This feature enables you to define context-menu options that are used to drill down to details of the information visible in a control. Depending on the various kinds of related information that need to be made accessible to users, you can define multiple ClickChoice relations for a control.

At run time, each ClickChoice relation is displayed as a context menu option on the right-click of the control. Selecting an option, opens a window that displays the detailed data.

The ClickChoice feature is available for the Input, Output, Text area, and Code Snippet controls. You can define ClickChoice relations for these controls through their [property sheet](#).

Before you begin:

- Ensure that the application to be displayed as a ClickChoice relation is created, published, and available for use.

To create ClickChoice relations:

1. Open the [XForm](#) containing the controls for which you need to define the ClickChoice relations.
2. Double-click the control to display its property sheet.
Alternatively, right-click and select **Properties**.
The <control name> dialog box opens.
3. Click **ClickChoice Relation** to expand the group box.
It displays a table comprising the Caption column.
4. Click  (Add) in the toolbar to add a ClickChoice relation.
5. Click  (Lookup) in the Caption column.
The Application Definition dialog box opens.
6. In **Menu Caption**, type the text that must display as an option in the context menu of the control.
This text will also display under ClickChoice Relation in the property sheet.
7. Click  (Lookup) for the URL field to specify a ClickChoice application.
The Select a URL dialog box opens.
8. Select a URL and click **OK**.
The selected URL is displayed in the URL field of the Application Definition dialog box.
9. In the ClickChoice Application Definition Properties area, provide other application definition details for the application that will display when this ClickChoice relation is selected.
10. Click **OK** to save the changes and close the Application Definition dialog box.
The Menu Caption is displayed in the Caption column under ClickChoice Relation in the property sheet.

To add multiple ClickChoice relations to a control:

- Click  (Add) in the ClickChoice Relation group box and repeat Steps 5 to 7.
The ClickChoice relations will be displayed as context-menu options of the control at run time.

Note: ClickChoice is not supported for the controls in XGrid.

Converting controls

You can convert a control to another control without affecting the properties and events associated with it.

To convert controls:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click a control, select **Change To**, and select the appropriate option from its submenu.

Depending on the option you select, the control is converted to another control.

- You cannot convert a primitive control (such as Input, Password, Output, and Textarea) into a grouping control (such as Table and Groupbox) and vice versa.
- The Code Snippet, Button, Image, and Frame controls cannot be converted to other controls and vice versa.
- When you convert a Check control into a Radio control, two radio buttons display to indicate the True and False values of the Check control.
- When you convert grouping controls, the constituent primitive controls are converted appropriately. For example, on converting Tab Pages in a Tab Group to a Table control, the fields (primitive controls) are displayed as columns in the table.

Customizing element bar and toolbar buttons

You can edit the properties of a button in the element bar or toolbar to customize it as required.

To customize element bar and toolbar buttons:

1. [Open the XForm](#) in the XForms Designer.
2. Double-click an element bar or toolbar button, or right-click it and select Properties. Depending upon the button selected, the Elementbar Button_or_Toolbar Button window opens.
3. Apply the appropriate modifications:
 - Type a unique identifier in the ID field.
A message is displayed confirming that all references are updated to the ID in the script.
 - Click **Yes** to update the script and proceed.
 - By default, the application assigns a unique identifier to the Find button and displays it in the ID field. You can edit this identifier, if required.
 - You cannot edit the ID of the Save button () in a toolbar.
 - Type a value in the Tab Index field to specify the order at which the button should be highlighted using the Tab key.
 - Type the text to be displayed as a tool tip in **Tooltip**.
 - Type the URL of the image to be displayed in the button in the Image field.
 - Select **Task Part** to create a task part for the button.
Selecting the check box displays the [Task Part Details](#) pane, where you can specify relevant details.

4. Click **Events** to expand the pane and select the Web service operation to be executed for each event. Alternatively, you can click  (Script Editor) for each event, and specify the appropriate Web service operation in the Script Editor.

The options available in the Events pane vary according to the button selected.

To specify the Web service operation to be executed before inserting a record	Select an option from Before Insert
To specify the Web service operation to be executed after inserting a record	Select an option from After Insert
To specify the Web service operation to be executed before the Find dialog box is opened	Select an option from Before Find
To specify the Web service operation to be executed after the search is complete	Select an option from After Find
To specify the Web service operation to be executed before the modified data in the XForm is saved	Select an option from Before Save
To specify the Web service operation to be executed after the modified data in the XForm is saved	Select an option from After Save

The properties of the selected button in the element bar or toolbar are updated.

To delete a button:

- Right-click it, and select  (Delete).

Customizing the Find dialog box

You can customize the Find dialog box in various ways.

Customizing search parameters and controls

You can use the Script Editor to modify programmatically the search parameters of the [Find feature](#). It is also possible to modify the input and output controls that appear in the Find dialog box at run time.

Customizing find methods

By default, the Find feature uses the Get, Next, and Previous events to retrieve and display records. However, you can edit the Find feature using the Script Editor to specify alternate methods.

Using a custom find feature

Apart from the Find feature provided by AppWorks Platform XForms that is accessible through the Element bar, you can define alternate mechanisms to search for records.

You can create a suitable user interface and, using the Script Editor, associate appropriate methods with it to search data.

Additionally, you can also use the property sheet to [customize the Find button](#).

Formatting and validating controls

Each data record that is displayed in a control comprises a description and a value. The description is what is displayed in the control during run time. The value associated with each data record is used at the backend for transactions related to the data record.

It is possible to format the description and value of a control, and specify validation rules for data-entry controls. For a data-bound control, the formatting and validation settings defined for its reference field in the WSDL or XSD are automatically applied. However, in some cases, you may need to set the formatting of a control at run time. It is possible to do so using the format, getFormat, and setValue (run time) methods.

During design time, you can specify the format and validation for a control as follows. This feature is available for the Input, Textarea, Output, List, and Select controls.

To format and validate controls:

1. [Open the XForm](#) in the XForms Designer.
2. To format and validate a control, right-click the control and select **Format**.
The Formatting Options dialog box opens. It displays the Control ID and Data Type fields, and the Format and Validation tabs.
3. From **Data Type**, select the type of data value supported for the control:

Integer	A numeric data type that can hold exact integer values. An integer is a 32-bit, signed (positive or negative) whole number without decimals. The Integer data type includes the XSD-defined types such as i1, ui1, i2, ui2, i4, and i8.
Float	A numeric data type used to store floating-point values. This type is useful for applications that need large numbers but do not need precise accuracy. This data type includes the XSD-defined types such as r4 and r8.
Double	A numeric data type used to store floating-point values. It offers greater precision and can store larger numbers than the Float data type.
Decimal	An exact numeric data type. It can accurately represent very large or very precise decimal numbers. This type is useful for applications (such as accounting) where rounding-off errors must be avoided.
Amount	A numeric data type that represents currency.
String	Contains a sequence of characters such as letters, numbers, and punctuation marks.
Date	A data type used for date and time.

At run time, you can specify the date by entering the number of days prior to or ahead of the current date. For example, enter '+1' to specify tomorrow's date, or enter '-7' to specify a date from the week gone by.

HexBinary	Represents arbitrary, hex-encoded binary data. The value space of HexBinary is a set of finite-length sequences of binary octets.
base64Binary	Represents Base64-encoded arbitrary binary data. The value space of base64Binary is a set of finite-length sequences of binary octets.
AnyURI	Represents a Uniform Resource Identifier Reference (URI). An AnyURI value can be absolute or relative, and may have an optional fragment identifier (that is, it may be a URI Reference). This type can be used to specify the intention that the value fulfills the role of a URI. The selected option is displayed in Data Type.

4. To set the field as mandatory, select **Required**.
5. Make appropriate modifications in the Format tab:
 - Select an option from **Locale** to specify your language and country.
If not defined, the settings of the client computer are used as Locale settings.
Changing the Locale also changes the date and currency settings accordingly.
 - Select a format for currency values from **Currency**.
This option is enabled only for the Amount data type.
 - Select an option from **Time Zone** to specify your time zone.
This option is enabled only for the Date data type.
 - Specify the number of decimal places to allow in an input value in the Number of Decimal field.
This option is enabled for the Float, Double, and Amount data types.
 - Select a predefined format for the specified data type from **Predefined Format**.
[Predefined formats](#) are available for the Integer, String, and Date data types.
 - Specify a format in **Specific Format** to use a [custom format](#).
 - Select a data-alignment option from **Data Alignment**.
 - Select a display format for positive values from **Positive Pattern**.
This option is enabled only for the Amount data type.
 - Select a display format for negative values from **Negative Pattern**.

Formatting is supported for various data types such as Integer, Float, Amount, String, and Date.
6. Make appropriate modifications in the Validation tab:
 - Specify the minimum value allowed in **Minimum Inclusive Value**.
 - Specify the maximum value allowed in **Maximum Inclusive Value**.
 - Specify the minimum length allowed in **Minimum Length**.

- Specify the maximum length allowed in **Maximum Length**.
- Specify the characters that are allowed in **Legal Character Set**.
You can use regular expressions to specify the characters.
For example:
 - a. To allow a string that starts with only an alphabet, you can use `^[a-zA-Z]`.
 - b. To allow only alphanumeric values, you can use `^[a-zA-Z0-9]+$`.
 - c. To allow a number that is three characters in length, you can use `^\d{3}$`.
- Specify the characters that are not allowed in the **Illegal Character Set** field. You can use regular expressions to specify the characters. For example:
 - a. To avoid space, you can specify `\s`.
 - b. To avoid specific special characters, you can specify `[@#$%^&*()]`.

You can use the `xforms-onvalidate` event to programmatically create additional validations for a control.

7. Make the appropriate modifications in the Messages tab.

- Type the message to be displayed, if the validation of the corresponding constraint fails, in the Message column.

The insertions specified as `{0}` in a message will be replaced at run time with the respective property of the control, depending on the constraint. When a translated XForm is viewed, the insertions in a message (`{0}`, `{1}`) are translated as well.

For example, consider a mandatory control with Data Type as 'integer' and a validation message that states 'Enter a non-empty value for `{0}`'. Here, the insertion will be replaced with the translated label at run time.

All default messages and tooltips are bundled in the `cordys.Xforms.messages.xml` message bundle. The Platform Administrator can specify translations for the message bundle by creating files such as `cordys.Xforms.messages_<language code>.xml`. For example, for translation to the Dutch language, the translated message bundle is `cordys.Xforms.messages_nl_NL.xml`.

8. Click **OK** to save all changes.

Clicking **Set default from XSD** restores the default settings specified for the control's reference field in the WSDL or XSD. This button is enabled only for data-bound controls.

The formatting and validation conditions for the control are set as specified.

Note:

- For the Select and List controls, the Validation tab is not available in the Format dialog box. You can only format the description for these controls.
- If, at run time, you enter a value exceeding 15 digits (including integer part and decimal part digits), then the value is not exact and is rounded off to the nearest number. This is applicable for all numerical data types.

- If, at run time, you enter a value exceeding 21 digits, the value is not formatted and is displayed as an exponential notation. Also, for a data-bound field, the value is not validated against the facets pattern, totalDigits, and fractionalDigits defined for it in the WSDL or XSD.
- AppWorks Platform XForms does not support the formatting and validation of controls that have attributes as references. Also, schema-based validations are not supported at run time.

Hiding controls in tables and Groupboxes

Grouping controls such as Group, Groupbox, Tab Page, and Table can contain multiple controls. In some cases, you may need to hide some controls in a grouping control and display only selected ones. The Column Chooser feature enables you to specify the controls to display during run time, without having to delete the controls.

You can thus customize an existing XForm according based on your requirements. The Column Chooser feature is available at both run time and design time for the Table control, and at design time for the Group, Groupbox, and Tab Page controls.

To hide controls in tables and groupboxes:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the Groupbox or Table control and select Column Chooser. The Columns dialog box opens and displays all constituent controls. By default, all controls are selected and display a check mark.
3. Make appropriate modifications:
 - Click **Checkboxes** to hide or display the check boxes in a Table control. This option is available only at run time for a Table control. Alternatively, for the Table control, you can clear the Checkboxes check box in the Table - Properties dialog box to hide the column.
 - Click a control to select or clear the control. Clearing the check mark hides the control. It is not possible to hide all columns of a Table control at run time, and an alert is displayed if you try to hide the columns. Also, at run time, the Column Chooser of the Table control displays only the columns that were selected during design time.
 - Click **Columns** to select or clear all controls. This option is enabled only at design time. The controls are hidden or displayed depending on the options you select.
4. Click .

The settings specified for the controls are saved and the selected controls are displayed.

Hiding Groupbox headers

When added to an XForm, the Groupbox control displays a header by default. If required, you can hide the header.

To hide groupbox headers:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the Groupbox control, and select the appropriate option:

Hide header	This option is available if the header is displayed in the Groupbox. You can use the move handle that appears on hiding the header to drag the control in the Designer Area.
Show header	This option is available if the header is already hidden in the Groupbox. The Hide header and Show header context-menu options toggle depending upon the status of the header of the Groupbox control.

3. Click .

Inserting lookup pages

You can associate two XForms using the lookup feature so that clicking the lookup button of a control on an XForm, opens the other in a separate lookup window.

Using this feature, you can display data in a control, while its detailed information is displayed in a separate lookup window. The lookup feature is supported for the Input, Output, Textarea, Code Snippet, List, and Select controls.

Before you begin:

- Create an XForm and publish it to run time to use it as the lookup page.

To insert lookup pages:

1. [Open the XForm](#) in the XForms Designer.
2. Double-click the control in the XForm. Alternatively, right-click the control and select Properties.
The <control name> window opens in the XForms Designer.
3. Click **Zoom Properties** in the <control name> window to expand the pane.
4. Make appropriate modifications to specify zoom properties:
 - Type the URL of the lookup page (XForm) in the Zoom URL field.
Alternatively, click  (Lookup), select the XForm in the Select URL dialog box that opens, and click **OK**.
The URL is displayed in the Zoom URL field. This field is mandatory.
 - Type the XPath of the business attribute to be returned from the zoom page in **Zoom Field**.
This field is mandatory.
 - Select an option from **Before Zoom** to specify the event handler (onbeforezoom) to be executed before the lookup page is opened.

Alternatively, click  (Lookup) to display the Script Editor and type the script for the event.

- Select an option from **After Zoom** to specify the event handler (onafterzoom) to be executed after the lookup page is closed.

Alternatively, click  (Lookup) to display the Script Editor and type the script for the event.

This sets the zoom properties for the control.

When you click the lookup button in the preview or during run time, the event specified in the Before Zoom list is executed, and the lookup page opens.

If you select an option or a check box in an open lookup page and click OK, the page closes, and the event specified in the After Zoom list is executed. The data associated with the selected option is retrieved and the application runs validations, if any, specified for the control. A message is displayed if the retrieved data fails validation. If the data clears the validation, the lookup window closes and the data is displayed in the control.

Note:

- By default, the lookup page opens in a separate window at run time. However, you can modify the application definition to display the page in any framework position. AppWorks Platform recommends that you do not modify the URL of the lookup XForm while exposing the application definition.
- To preserve the positioning of controls during run time, ensure that the layout of the lookup XForm is vertical and that the controls are not anchored.

Positioning controls

On adding a control to an XForm or a composite control, it is automatically positioned according to the layout specified for the parent control. It is possible to modify the control such that you can position it independent of the parent control's layout.

To position controls:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the control and select an option from the **Position** sub-menu.
Alternately, select a control, click  (Extend Menu) and select an option from Position.

Inherit	Apply the layout defined for the parent control or XForm. If the control is dragged to another parent control, it inherits the layout from the new parent control and is repositioned automatically.
Vertical	Position the control vertically in the parent control or XForm.
Horizontal	Position the control horizontally in the parent control or XForm.
Free	Freely position the control in the parent control or XForm

If you select Vertical, Horizontal, or Free, the specified position overrides the layout that applies to the control.

The control is repositioned according to the selected position.

Note:

- Specific features are available for each layout option of an XForm. After modifying the position of a control, it is possible to use the functionality specific to the position independent of the parent control's layout.
- When you modify the layout of an XForm or composite control, the layout overrides the position settings of the constituent controls.

Positioning labels

Labels are intended to describe the type of content displayed in controls. Some specific controls in AppWorks Platform XForms are associated with labels, by default. On adding them to an XForm, the labels display the control name such as Input1 and Input2. To rename it, you can click on a label and type the new label.

By default, labels are placed at the top of controls. However, you can specify different display positions for labels with respect to the control.

To position labels:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the control, select **Label**, and select the appropriate option from its submenu:

Hide	Hide the label.
Top	Position the label at the top of the control.
Side-Left	Position the label to the left of the control, and align the text to the left edge of the label, away from the control element.
Side-Right	Position the label to the right of the control, and aligns the text to the right edge of the label, towards the control element. The label is aligned according to the option you select.

3. To select multiple controls and position labels simultaneously, press **CRTL**, and select the required controls.
4. Click .

Depending on the option you select, the label is repositioned with respect to the control.

Pre-filling list and select controls

You can use the Select and List controls to input data on an application page (XForm). These controls can display multiple input values, from which you can select the appropriate option as the input. The multiple options that display in the Select and List controls can be defined

manually or retrieved for display from the backend (for example, from a database). Using the controls' property sheets, you can pre-fill these controls either manually or from the backend, as required.

Each option displayed in the controls is associated with a description and a value. The description of the option is displayed in the controls at run time, while the value of the option refers to the actual value to which the description is mapped internally. The value is not visible at run time and is only used in the backend. When prefilled from the backend, description and value refer to business object nodes of the dataset used.

The procedure to pre-fill both the Select and List controls is similar.

To pre-fill list and select controls:

1. Open the XForm in the XForms Designer.
2. Double-click the control displayed in the Designer Area.
The <control name> window opens in the XForms Designer.
3. Click  (Lookup) in the Dataset field.
The Pre-fill data source for <control> dialog box opens.
4. Select an option from **Dataset Source**:

Existing Model	Prefill the control using an existing model in the XForm.
Custom Content	Manually define the options with which to prefill the control.

Depending on the option you select, the Existing Model or Custom Content area is displayed in the Prefill data source for <control> dialog box.

In the Existing Model area, you can define the dataset to prefill the control. At run time, the specified model fetches the data from the dataset and prefills the control.

5. To specify the model from which to prefill the control, select an option from **Model**. Ensure that you do not select the model specified in the Model field of the <control> dialog box.
6. If the existing models are not suitable, click  (Lookup) from Model to create a model. The Model Properties dialog box opens, where you can define the new model.
7. Click  (Lookup) for the Node Set field to display the Model References dialog box.
8. Select the appropriate option and click **OK** to display it in the Node set field. This field is optional and corresponds to the Reference|initialize()|parameter of the initialize() method used to programmatically prefill the control. At run time, the selected node is used as the starting point for data references, that is, the control is pre-filled using the constituent nodes under the selected node.
9. Click  (Lookup) for the Description field to display the Model References dialog box.
10. Select the appropriate option and click **OK** to display it in the Description field. At run time, values from the selected node are pre-filled as descriptions in the control.
11. Click  (Lookup) for the Value field to display the Model References dialog box.

12. Select the appropriate option and click **OK** to display it in the Value field.
At run time, values from the selected node are prefilled as values in the control.

In the Custom Content area, you can manually specify the options with which to prefill the controls at run time.

13. Click  (Add) to display an empty row, and type the description and value of the option under the Description and Value columns, respectively.
14. Select the description and value of an option and click  (Delete) to delete it.
15. Click **OK** in the Pre-fill data source for <control> dialog box.

This saves the pre-filling settings to be used for the control at run time.

You must use the initialize() method to manually prefill the Select and List controls with non-transactional or complex transactional data (containing multiple levels of nodes).

Resizing controls

While designing XForms, you can resize controls vertically or horizontally to fit the required layout.

To resize controls:

1. Open the **XForm** in the XForms Designer.
2. Place the cursor on the vertical or horizontal edge of the control.
The resize handle is displayed.
For controls that cannot be resized vertically, such as the Input, Password, Output, Select, and Check controls, the resizing handle is displayed only on the vertical edges.
3. Drag the edge of the control to resize it.
Alternatively, select the control, and press Shift and the arrow keys.

The control is resized accordingly.

- You can also select multiple controls and resize them. To select multiple controls, press the CTRL key and select the required controls.
- Right-click the control and select **Set Default Size** to reverse the changes made and display the default size of the control. This option is available only for vertical layouts and resizes the control according to the size of the parent control.

Consider the following while resizing controls:

- The Groupbox and Group controls automatically resize when controls are added to them.
- Grouping controls can be resized diagonally with the help of the diagonal resize handler located at the bottom right corner of the grouping control.
- If you decrease the size of the Input control to less than the label size, the label shifts from the left to the top of the control.
- You cannot resize the toolbar, toolbar buttons, or the control bar and pagination bar buttons.

Setting synchronized dialogs

In XForms where you need to display data in multiple fields of a Table control, the user interface may look cluttered and become difficult to navigate. To avoid this, you can structure the Table control to display only the essential information and use synchronized dialogs to display the details of each record in the Table control.

Synchronized dialogs are XForms that are invoked on clicking a record in a table. Before specifying synchronized dialogs, you must create the XForm containing the relevant fields and publish it.

Before you begin:

- Ensure that the XForm to be specified as the synchronized dialog is published to run time.

To set synchronized dialogs:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the Table control in the XForm, and select **Properties**.
Alternatively, double-click the control. The Table window opens in the XForms Designer.
3. Click **Synchronized Dialog Options** to expand the pane.
It displays various options to specify the synchronized dialog.
4. Type the URL of the XForm to be used as the synchronized dialog in the Dialog URL field.
Alternatively, click  (Lookup) to open the Select URL dialog box, select an XForm, and click **OK**.
The URL of the XForm is displayed in the Dialog URL field.
5. Select an option from **On Dialog Open** to specify the event handler to execute before opening the synchronized dialog.
Alternatively, click  (Script Editor) to open the Script Editor and specify the event handler programmatically.
The specified event handler is displayed in the On Dialog Open list.
6. Click **Dialog Properties** to expand the pane, and make appropriate modifications:
 - Type a unique ID in the ID field for identifying and referring the synchronized dialog.
 - Type the text to be displayed in the XForm titlebar and tab caption in the Caption field.
 - Type the text to be displayed in the XForm title bar and as the XForm name in the navigational tree in the Description field.
If the Dialog Caption and Dialog Description are different, both are shown in the titlebar as <dialog caption> - <dialog description>. If they are the same, only the Dialog Caption is displayed, and the text is not repeated in the title bar
 - Select **Docked** to specify whether the synchronized dialog must be docked in the AppWorks Platform Classic view.

- Type an integer value in **Left** to specify the position of the left edge of the synchronized dialog.
- Type an integer value in **Top** to specify the position of the top edge of the synchronized dialog.
- Type an integer value in **Width** to specify the width of the synchronized dialog.
- Type an integer value in **Height** to specify the height of the synchronized dialog.

The unit for the values that you enter in the Left, Top, Width, and Height fields is taken as px (pixel), by default. Also, these options are available only if the synchronized dialog is specified as undocked.

7. Click .

Caution: Ensure that the XForms used to create synchronized dialogs contain models with common business objects. In case of XForms created using AppWorks Platform operators, the operators must belong to the same Web service interface.

Restrictions:

- In case of synchronized dialogs for which both the XForms contain tables, the synchronization of data between the tables is not supported.
- The synchronized dialogs feature is not supported for non-transactional models.

Setting properties for controls, regions, and app palettes

After adding a control, region, or App Palette to an XForm, you can modify it to achieve the required run-time behavior. To do so, you need to specify appropriate properties and events in its property sheets.

Each control is associated with a Basic Properties App Palette where you can set basic properties and a property sheet where you can set advanced properties for the control.

To set properties for controls, regions, and app palettes:

1. [Open the XForm](#) in the XForms Designer.
2. Select the control in the Designer Area.
The  (Extend Menu) icon appears on the top right corner of the selected control.
This icon appears only for the current control that you have selected.
3. Click  (Extend Menu).
The Basic Properties App Palette opens.
4. Make appropriate modifications in the Basic Properties App Palette.

ID	Enter the ID of the control.
Class	Enter the name of the style class to be applied to a control.
Height	Enter the height of the control to set its size at run time.

Width	Enter the width of the control to set its size at run time.
Position	Select the position of the control. For details, see Positioning Controls .

- For the Input, Password, Output, Check, Radio, Select, Text, and Frame controls, Height is set to auto and is disabled.
 - In a Vertical layout, the width is set as 100% for all controls, by default.
 - In a Horizontal layout, the width is set as 44% for all controls, by default.
 - You can set the Height and Width in pixels (px), points (pt), em, ex, millimeters (mm), centimeters (cm), inches (in), and percentages (%).
5. Right-click the control and select Properties. Alternatively, double-click the control. The <control name> window opens, displaying various collapsible panes, where you can specify the required properties and events.
 6. Make the appropriate modifications in the <control name> window.
 7. Click .

The properties specified for the control, region, or App Palette are saved.

You can use the Message Map for associating data with the events of a control. For details, see [Using Message Map](#).

You can view and specify properties for controls, toolbar buttons, element bar buttons, and also for the XForm as a whole. For details, see [Property Sheet of Controls, Regions, and App Palettes](#) and [Setting Properties of an XForm](#).

Specifying data values for check controls

A Check control is normally used for Web service operations involving true or false as values. This is because the Check control can hold only one of these values at a time.

However, it is possible to modify the control to represent a value retrieved from the backend. Depending on the value, the Check control is displayed as selected or cleared in an application page.

To specify data values for check controls:

1. Open the XForm in the XForms Designer.
2. Right-click **Check** in the XForm, and select **Properties**.
Alternately, double-click the control. The Check window opens in the XForms Designer.
3. Click to expand the Data Value pane and make appropriate modifications:
 - In **True Value**, specify the value for which the check control will display as selected.
 - In **False Value**, type the value for which the check control will display as cleared.
4. Click .

Splitting controls, app palettes, and XForms

You can divide App Palettes, grouping controls, and XForms into different areas to group controls according to functionality or intended usage. Grouping controls are controls that can hold other controls, such as Groupbox, Group, and TabPage controls.

It is possible to split an App Palette, a grouping control, or an XForm vertically or horizontally. Each split area behaves as a separate, invisible container. By default, each split area retains the layout defined for an App Palette, grouping control, or XForm. You can, however, specify layouts and position and align controls for individual areas.

The procedure to split an App Palette, a grouping control, or an XForm is similar. Consider the procedure to split a grouping control.

To split controls, app palettes, and XForms:

1. [Open the XForm](#) in the XForms Designer.
2. To split a grouping control, right-click it, select **Split** and select the appropriate option from the sub-menu:
 - Select **Vertical** to split vertically.
 - Select **Horizontal** to split horizontally.

The splitter bar appears, and the grouping control is split into areas. On choosing Vertical, the splitter bar is placed to the left of the control, by default. On choosing Horizontal, the splitter bar is placed towards the top of the control. You can drag the splitter bar to resize the areas.

Also, on splitting a control, the controls within are redistributed across the split areas according to their respective positions and the control layout:

- For a vertical layout, all controls are repositioned to the left area when split vertically, and to the top area when split horizontally.
 - For a horizontal or free layout, the existing controls are repositioned and distributed on both sides of the split XForm, depending on their position.
3. Right-click the splitter bar and select Properties to specify its properties. Alternately, double-click the splitter bar. The Splitter window opens.
 4. Make the appropriate modifications to set the splitter properties.
 - Type a unique identifier for the control in the ID field. This is a design-time property.
 - Select Fixed to fix the position of the splitter bar at run time. If not selected, you can alter the position of the splitter bar at run time.
 - Type the text to be displayed as a tool tip in the Tooltip field.
 - Select an option to position the splitter:
 - a. Select **Left** to position the splitter with respect to the left side of the parent control. It divides the XForm in the ratio 1:2, horizontally.

- b. Select **Right** to position the splitter with respect to the right side of the parent control. It divides the XForm in the ratio 2:1, horizontally.
- c. Select **Top** to position the splitter with respect to the top of the parent control. It divides the XForm in the ratio 1:2, vertically.
- d. Select **Bottom** to position the splitter with respect to the bottom side of the parent control. It divides the XForm in the ratio 2:1, vertically.

The positioning options are also available in the context menu of the splitter. You can set or modify the position of the splitter using the context menu.

The splitter properties are set and the splitter bar is positioned as specified.

The distance between the splitter bar and the side with respect to which it is positioned remains constant on resizing the App Palette or grouping control.

Depending on the option you select, the grouping control is divided into areas.

Note:

- Double-clicking the splitter bar at run time repositions the bar such that the bar overlaps the edge with respect to which it is positioned. This hides the controls positioned in the corresponding split area. To restore the collapsed split area, double-click the splitter bar.
- To remove a splitter, right-click the splitter bar and select **Delete Splitter**.
- Using the **Split** context menu option splits the XForm or control into areas that cannot be further split into regions.
- Using the **Split As Regions** context menu option divides the XForm into regions (containing App Palettes) that can be further split into regions or areas.

Using message map

AppWorks Platform XForms provides an easy-to-use interface for mapping data between two data sources without having to resort to manual scripting. Using the Message Map, you can map model data to various controls in an XForm.

In case of composite controls such as Google Map and Chart controls, the data format differs from the standard data format of AppWorks Platform models. Message Map enables you to define data associations between such disparate data sources. The Message Map uses the schema of a composite control to map it to a AppWorks Platform model.

AppWorks Platform supports two-way mapping of data, that is,

- Changes in model data are reflected in the mapped XForms control, and
- Changes made to data in the XForm control are reflected in the mapped model.

A Mapping Object represents the association defined between two data sources.

To launch the Message Map through the Model Properties dialog box:

1. Select **Advanced** in the XForms Designer toolbar.
The toolbar expands to display icons that launch advanced functionality in the XForm.
2. Click  (Manage Model) in the XForm Designer toolbar.
The Manage Model dialog box displays all the models associated with the XForm.
3. Click  (Message Map) for the model that needs to be mapped.
The Message Map dialog box opens. The Message Map dialog box comprises Source and Target panes that are separated by a middle area where data associations are defined. The middle area contains rows of two fields each that display mapped elements. All the available data sources (controls with schema) are listed in the Source and Target panes. A Source element provides data for mapping and a Target accepts data from the source and processes it. Based on the requirements of the application you are creating, you can define a data source as a source or a target in the Message Map.
4. Expand the nodes under the **Source** pane and drag the element to be mapped to a field in a row in the middle pane.
Similarly, drag the corresponding element to be mapped from the **Target** pane to the second field in the row.

While creating a message map using a chart, map the X-Axis label to a string and Y-Axis label to a numeric value.

This maps data from source to the target. The mapped data is represented as follows in the XML Editor.

```
<xml id="EmployeesTable_xforms-onrowselect_assignments">
<Assignments>
<Assignment>
<Source>city/value</Source>
<Target>googlemap1/Map/Location/City</Target>
<Operation/>
</Assignment>
</Assignments>
</xml>
```

Using web components in XForms

Web components are simple, reusable programs that impart specific functionality to controls in an application. Web components offer various functionality that is not generally available in standard controls.

The use of web components makes the process of developing new applications quick and easy. Moreover, it introduces an element of standardization and uniformity into an application, and reduces the chances of errors.

The web components available in AppWorks Platform are listed as Web Component Libraries. The web components also available in AppWorks Platform XForms are Splitter, Toolbar, Table, Tab Page, Calendar, Select, List, and Translator. The Frame web component is available only with AppWorks Platform XForms.

Although you can use any of the listed web component libraries, it is recommended that you use the XForms web components, wherever possible, as these provide additional functionality that are not otherwise available.

To use web components, add them to the XForm or to a control in the XForm, as required. You can also specify their definition in the Code Snippet control to add web components.

AppWorks Platform recommends that you directly add the complex web components (such as a tree component) that appear in the XForm during run time. For simple web components such as upload and progress bar, that do not appear in the XForm during run time, you can specify definitions in the Code Snippet control.

Adding Web Components Directly to an XForm

You can use the following methods to add and remove a web component from a control:

- To add, use `application.addType(<object>, <fullyQualifiedNamespace>);`
- To remove, use `application.removeLibrary(<url>, <object>);`

Where, `<url>` refers to the path of the web component, and `<object>` refers to the ID of the control.

After adding the web component to the control, you can access the properties and methods of the web component through the control.

```
// Adding the "upload" web component to a button with ID "upload" in the
Init Done event. function Form_InitDone(eventObject) { application.addType
(upload, "wcp.library.util.Upload"); } //Removing the "upload" web
component from a button with ID "upload" in the Before Close event.
function Form_BeforeClose(eventObject) { application.removeLibrary
("/cordys/wcp/library/util/upload.htm",upload); } // onClick event handler
for the button "upload" //Accessing the "browse" method of the "upload"
web component through the "upload" button. function upload_Click
(eventObject) { //Using the browse method on the button. upload.browse(1);
}
```

Specifying the `removeLibrary()` method ensures efficient garbage collection.

Adding web components using a code snippet control

You can also add a web component to an XForm using the Code Snippet control.

- To do so, add the following definition of the web component to the Code Snippet control.

```
//Adding the definition for the 'progressbar' web component <div
cordysType="wcp.library.ui.ProgressBar" id='progressbar'></div>
```

After adding the web component to the Code Snippet control, you can access the properties and methods of the web component using its ID.

```
//OnClick event handler for the button "startProgress" //Accessing the  
"start" method through the ID of the progressbar web component. function  
startProgress_Click(eventObject) { //Start filling the progress bar.  
progressbar.start(100); }
```

Viewing markup of controls

AppWorks Platform XForms enables you to easily create user interfaces for your applications, without resorting to manual programming. However, in some cases, you may need to programmatically use the markup of a user interface. AppWorks Platform XForms provides a functionality wherein you can design the required user interface and copy for use, the markup generated for the interface. Using AppWorks Platform XForms, you can view and use the markup of a single control or an XForm.

To view markup of controls:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the required control and select **Get XForms Markup**.
The XForms Markup dialog box opens. It displays the markup generated for the control.

To view the markup of an XForm:

1. Click  (XML) in the XForms toolbar.
The markup of the XForm is displayed in the Source window.
2. You can select and copy the markup generated for the control or XForm and use it as required.

Autoextend

Autoextend is feature in AppWorks Platform that allows you to automatically set the height of the controls in the User Interface. This functionality is supported for the following controls in an XForm.

- Group
- Table
- Choice-box
- Control-box

When you drag and drop the controls, the autoextend behavior will be automatically added to the controls except for the forms in free layout. The height of these controls will be automatically set to auto. Adding other controls to these autoextend-enabled controls (Grouping controls) increases the height of the layout and removing the controls will decrease its size. You can also specify a fixed height for these controls in the basic property sheet or resize the controls and disable the autoextend behavior. Similarly, you can use set default size in the context menu to enable the autoextend behavior. For the layout element (layout div) the autoextent class will be added after the layout class in the class name.

Consider the behavior of autoextend functionality on a table control. The property "Minimum Number of Rows to Display" determines a default value for the height of the table. When autoextend behavior is enabled, the height of the table changes depending upon the value of this property. The number of rows that will be displayed during runtime is independent of the number of records and is always greater than or equal to the value in **Minimum Number of Rows to Display**. If autoextend behavior is added to the table, the size of the table increases with insertion of rows; whereas a scroll bar will be displayed if you specify a fixed height in the table.

For all the controls that support autoextend functionality, the styles of the controls are defined in `xDefault.css`. A prototype method `isAutoExtentEnabled` is available for each control in the designer library. The method returns a true Boolean value if the control is autoextend enabled. To enable autoextend for the custom controls, you need to define the prototype method in the designer library and specify the return value as true. The framework retrieves the correct behavior if a proper autoextent css class is available for that control.

Writing scripts for events

Use the Script Editor to write or edit scripts for events exposed by the XForms object model.

To write or edit the script for a control:

1. Open the XForm in the XForms Designer.
2. Double-click the control, or select **Properties**.
The <control> window for the selected control opens.
3. Click the **Events** pane.
The Events pane expands to display the events available for the control. The list of each event displays available functions.
4. Select a function to attach it with the event, and click  (Script Editor).
The Script tab displays a function name that combines the ID of the control and the name of the event.
The Script tab opens to display the Script Editor, using which you can relate the control to the event.
5. To create a function, click  (Script Editor) without selecting any function.
The Script tab displays a function name that combines the ID of the control and the name of the event.
The Script tab opens to display the Script Editor, using which you can relate the control to the event.
6. Enter the appropriate script and click  (Check Syntax) to validate it.
An alert is displayed if the script contains any syntax or spelling errors.
The Script Editor does not support the overwrite mode. If you press the Insert key to use this mode, an alert is displayed and the Script Editor automatically switches to the default Insert mode.
7. Make the appropriate corrections, and click .

Example

Consider a Button control with 'button1' as ID for which a function needs to be created to display an alert when it is clicked. The Script tab displays a function that combines the control ID with the event name, by default.

```
function button1_Click(eventObject)
{
    alert('Please enter mandatory information to proceed.')
}
```

After you complete this task:

- Click  (Preview) to view the XForm in the Preview window.
In case of the above example, clicking the button (with 'button1' ID) displays the alert specified in the Script Editor.

Customizing XForms

Besides customizing the individual controls on an XForm, you can also perform customizations on the XForm. You can [set the properties](#) of an XForm, [specify its layout](#), or [add a toolbar](#) to it. You can also [set the positioning grid](#) that is displayed in the Designer Area.

Additionally, you can:

- [Create UI using the Field Chooser](#)
- [Generate Input and Output User Interfaces](#)
- [Generate Layouts from Web Service Operations](#)
- [Preview XForms](#)
- [View Scripts](#)
- [Use XLets in XForms](#)

Adding a toolbar to an XForm

You can use the XForms Designer to add toolbars to display common, important functionality applicable in an XForm. It is also possible to group buttons in a toolbar using separator bars. This enables easy identification of similar functionality.

While creating an XForm in the XForms Designer, you can add two types of items (buttons) to the toolbar - standard and custom. Standard items refer to the Save and Print options that are commonly used in most XForms. You can add other items to introduce more functionality to the XForm.

To add a toolbar to an XForm:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the **Designer Area**, select **Toolbar**, and select the appropriate option from the sub-menu:

Add Item	Add the toolbar to the XForm. The toolbar displays a placeholder button, by default.
----------	---

	<ul style="list-style-type: none"> ■ Double-click the placeholder button to open its property sheet and define its properties.
Add Standard Items	<p>Add the Save () and Print () buttons to the toolbar.</p> <p>You can only add one standard item (Save and Print buttons) in a toolbar.</p>
Delete Toolbar	Delete the toolbar.

3. Right-click the toolbar to access any of the context-menu options.
4. Double-click a toolbar (or toolbar item) or right-click it and select **Properties** to open its property sheet and modify its properties.
A toolbar that displays buttons according to the option you select is added to the XForm.
5. You can drag buttons on the toolbar to rearrange them. Placeholders are displayed to help you position the buttons.
6. To group appropriate buttons, right-click the toolbar and select **Add Separator**.
A separator bar (|) is added to the toolbar.
7. Click .

The toolbar is added to the XForm.

Creating UI using the field chooser

You can use the Field Chooser to view the response and request parameters of a model. Using the available parameters, you can create the input and output user interface of an XForm.

To create a user interface using the field chooser:

1. [Open the XForm](#) in the XForms Designer.
2. Click  (Manage Models) from the toolbar.
3. The Manage Models icon () is available only in the advanced toolbar of the XForms Designer.
4. To view the icon, select **Advanced** in the toolbar.
The Manage Models window opens, displaying the models available in the XForm.
3. Click  (Lookup) for the appropriate model.
The Fields - <model> window opens.
4. You can select a model from the **Model** list to view its request and response fields.
4. Drag a field from the Fields pane to the XForm.
The control associated with the field is added to the XForm. The model and reference for the control are updated in its property sheet.
5. For complex data or non-transactional data, specify the complete XPath of the field in

the control property sheet.

6. Click .

Generating input and output user interfaces

While creating an XForm using Web services, you can specify the interface to be generated for each Web service using the options available in the Select Web Service Operation dialog box.

The following options are available in the Select Web Service Operation dialog box for creating UI:

- Generate UI for Input message
- Generate UI for Output message

When you select **Generate UI for Input message** and **Generate UI for Output**, the action:

- Generates the input and output UI for the Web service.
- Creates a model.
- Creates the script for the on-click event handler of the button created in the UI.
- Creates a SOAP request in case of external Web services. You can view the SOAP request using the XML Editor.

To send a request to the service group:

The button added in the UI displays the name of the Web service operation.

- Click the button that was added in the UI display to send a request to the service group along with the parameters provided in the Input UI.
This action fetches the result. The response data, if generated, is displayed in the output UI.

When you select **Generate UI for Input message**, the action:

- Generates the input UI for the Web service.
- Creates a model.
- Creates the script for the on-click event handler of the button created in the input UI.
You can edit this script using the Script Editor.
- Creates a SOAP request in case of external Web services. You can view the SOAP request using the XML Editor.

When you select **Generate UI for Output message**, the action:

- Generates the output UI for the Web service.
- Creates a model.
- Creates a SOAP request in case of external Web services. You can view the SOAP request using the XML Editor.

When you clear both **Generate UI for Input message** and **Generate UI for Output message**, the action:

- Creates a model.
- Creates a SOAP request in case of external Web services. You can view the SOAP request using the XML Editor.

Note:

- No UI is generated if both the Generate UI for Input message and Generate UI for Output message options are cleared in the Select Web Service dialog box.
- If the header of a request contains information that must be sent with the request, the appropriate input UI is automatically created to collect relevant inputs. In such cases, select Generate UI for Input message to enable the generation of UI. Alternatively, you can specify the information required in the header using the request property of the xforms-onrequest event of the Web service.
- Information from the header of a response is not displayed in the UI.

Generating layouts from methods

The XForms Designer provides various controls that can be dragged from the Toolbox tab to generate layouts. The data types, formats, and properties of these controls are then defined and the fields are validated.

However, XForms provides a much quicker way of generating both input and output user interfaces. It is possible to generate a default UI and the model necessary to send and receive data from the backend using a AppWorks Platform Web service (Web service operation) or the WSDL of any external Web service.

When a AppWorks Platform Web service operation is dragged and dropped to the XForms Designer or a WSDL is imported, a model is created automatically and Designer Area displays a default layout of the XForm based on the WSDL. You can retain the default layout or choose another layout for the XForm using the Layout option.

If the Web service operation that is dragged does not have a WSDL, the model is created with only a Groupbox control on the XForm. You need to manually create the controls required on such an XForm.

Generating default layouts

The default layouts generated by the XForms Designer depend on the type of AppWorks Platform Web service operations generated using the [Web Service Generator](#).

The following are the basic methods available in the `_Web Service Generator_` for any table in a database:

Get<TableName>Object	Returns zero or one record from the database.
Get<TableName>Objects	Returns zero or more records from the backend.
GetNext<TableName>Objects	Returns the next set of records, greater than a

	specified key value, from the backend.
GetPrevious<TableName>Objects	Returns the previous set of records, less than a specified key value, from the backend.

The key value is usually the parameter (primary key) used to query for the GetObject and GetObjects methods.

Custom methods can also be generated using the [Query Builder](#) provided by the AppWorks Platform application. Such methods can be interpreted by the XForms Designer to propose a default layout.

Types of layouts

The following layouts are proposed by the XForms Designer using the methods:

- Form layout (Details view)
- Grid layout (List view)
- Aggregated layout (Details-List view)

Form layout

Using the Get<TableName>Object Web service operation, the XForms Designer proposes a Form layout. Since data returned is zero or one, all the fields retrieved from the Web service operation are displayed in a Groupbox control as a detailed view. If the same Web service interface has the GetNext<TableName>Objects and GetPrevious<TableName>Objects methods, the Groupbox control is generated with an element bar to navigate through detailed records.

Because Next and Previous methods always return multiple records from the backend, it is necessary to retrieve a manageable number of records from the backend considering the time taken to retrieve and render the data in client. This is handled in XForms by using a server-side cursor (Iterator Size) that can be specified on a model. By default, Iterator Size is 50, that is 50 records can be retrieved from the backend at a time, and a fresh request needs to be sent to the database if this limit is exhausted.

Any custom Web service operation that returns only a single record will result in a Groupbox control generated in the XForms Designer.

Grid layout

Using the Get<TableName>Objects Web service operation, the XForms Designer proposes a grid (table) layout. The data returned is always zero or more, so all fields retrieved are shown in a table as a list view. Since this Web service operation itself retrieves a range of records, the GetNext or GetPrevious methods are not needed.

AppWorks Platform recommends that you do not use the GetNext or GetPrevious methods with the Get<TableName>Objects Web service operation. If you do set the GetNext or GetPrevious methods in the model property sheet of the Get<TableName>Objects Web service operation, a notification is displayed and the Get<TableName>Objects Web service operation is automatically changed to the Get<TableName>Object Web service operation.

Any custom Web service operation that returns more than one record will result in a Grid generated in the XForms Designer.

Aggregated layout

Aggregated methods for any table can be generated using the Query Builder tool available in AppWorks Platform. These methods have complex responses. For such methods, the XForms Designer proposes an aggregate layout that is a combination of both detail and list view. This layout occurs when the response is a complex XML. All the fields retrieved from the Web service operation display in a Groupbox control as a detailed view enclosing other controls and grids. If the complexity exceeds 1-N level, the response is displayed as XML in a text box.

Consider the following example:

```
<employees> <ID> <name> <orders> <orderID> <orderdate> <products> <ID>
<name> </products> <products> <ID> <name> </products> </orders> <orders>
... </orders> </employees>
```

For the above response, the Groupbox control displays the employee data, such as ID and name, as input fields, and the orders for each employee in a table. The orders table displays the following:

- Input controls for order data such as orderID and orderdate.
- Textarea control for product data. This data is displayed as XML as it exceed 1-N complexity level.

For complex responses, a Groupbox control will be proposed by the XForms Designer and the element bar for the Groupbox control depends on the number of records returned by the Web service operation. If the Web service operation returns more than one record, then the element bar is generated for the Groupbox control. If only a single record is returned, element bar will not be generated.

Note:

- If a Web service interface has all the three methods for a Table that is Get<TableName>Object , GetNext<TableName>Objects , and GetPrevious<TableName>Objects , then dragging any of these methods to the XForms Designer generates a Groupbox control with an element bar. This applies to both default generated methods and Custom methods with the naming convention as above.
- The different layouts generated are based on the methods used and results returned. For complex layouts, it is possible to generate data only for aggregated relations. Association methods (for example, Get<TableName1>With<TableName2>Objects) are not supported. In such scenarios, you can use separate Object or Objects methods for the database tables, and associate them using XForms data model.

Layouts generated from various web service operation combinations

The following table shows the layouts generated when various combinations of methods are dragged to an XForm:

Web service operation combinations	Expected result
Get<TableName>Object	A Groupbox control is generated. You can see one record in a detailed view at a time.
Get<TableName>Object and Web service interface has the GetNext and GetPrevious methods	A Groupbox control with an element bar is generated. Using the pagination bar, you can navigate between records. As the Groupbox control displays only one record at a time, you can convert the UI to a table to view records page by page.
Get<TableName>Objects	A Table control is generated where all records appear in a list view. The number of records viewed depends on the height of the table.
Get<TableName>Objects and Web service interface also has the GetNext and GetPrevious methods	AppWorks Platform recommends that you do not use the GetNext or GetPrevious methods with the Get<TableName>Objects Web service operation. If you do set the GetNext or GetPrevious methods for the Get<TableName>Objects Web service operation, a notification is displayed and the Get<TableName>Objects Web service operation is automatically changed to the Get<TableName>Object Web service operation. On again, programmatically setting the GetNext or GetPrevious methods with the Get<TableName>Objects Web service operation, duplicate records are retrieved from the backend.
GetNext<TableName>Objects or GetPrevious<TableName>Objects	Generates a form layout if the Web service operation has a GetObject Web service operation and grid layout if it has a GetObjects Web service operation, in the above order of preference. If GetObject Web service operation is available, then all three methods are used.

When a default layout is proposed, it is also possible to change the layout to other grouping controls such as Tab Page, Table, and Groupbox using the Change To context-menu option of the XForm.

For an external Web service, XForms designer parses its WSDL and proposes a suitable layout depending on the response of the service. If the service returns only a single record, the XForms Designer proposes the Groupbox control. If the service returns more than one record, then a Grid is proposed. In case of complex responses, a Groupbox control is

proposed and the element bar of the Groupbox control depends on the number of records returned by the service. The element bar is generated by default when more than one record is returned by the service.

Default controls

Depending on the WSDL or XSD of the Web service operation that you select, AppWorks Platform XForms generates various controls inside a Groupbox control or a Table control. The following table provides information about the automatically generated controls:

XSD definition	Controls generated
Input/Output	<p>Elements of all types except Boolean.</p> <p>For elements of type date, the input control will contain the calendar look-up button.</p> <ul style="list-style-type: none"> ■ If the Web service interface to which the selected Web service operation belongs also contains an Update Web service operation, then only input controls are generated, else only output controls are generated. ■ If the element has a fixed attribute defined in the WSDL/XSD, then the output control is generated even if the Web service interface contains an Update Web service operation.
Select	Elements that have enumerated data defined in the WSDL/XSD.
Check	Elements of the type Boolean.
Textarea	<ul style="list-style-type: none"> ■ Element types that are identified as any in the WSDL. ■ Element length greater than 80.
Table	For elements of the array type, AppWorks Platform XForms generates a table and sets the reference to table as "*". This leads to errors when you insert a row in to the table at run time.

Query builder

The [Query Builder](#) can be used to generate associate and aggregate queries. Those queries generally retrieve data from more than one table using the relational constraints between the tables. XForms does not support generating a UI for the association queries, but still association can be achieved in the UI by the following procedure.

Consider two tables - Table1 and Table2 - that are associated to each other. Instead of using the default generated associated methods, association can be achieved by the following steps:

1. Generate Get<Table1>Objects Web service operation for Table1 and Get<Table2>Objects Web service operation for Table2.
2. While drag-dropping the methods, XForms designer asks for associating the models generated. Select*Yes*to associate model for Table1 and Table2.

3. Check if the necessary relation between the constraint fields (Generally the foreign key of Table1 and primary key of Table2) are set correctly in the model properties.

Custom layouts

Though default layouts are proposed when the designer understands the relationship between the methods dragged, the developer also might need to create some composite layouts in order to match user requirements. Some of the custom layouts might demand a developer to show multiple views of the same model, can involve associating models to synchronize the data shown between multiple views of the model. The composite layout consists of at least two views: a main view and an associated view.

The composite views can be:

- From same data source: Since data source is same, developer can use a single model and use it as data source for both the views (Groupbox and Table). Data in such cases will be synchronized between views automatically.
- From different data sources: Models need to be associated. Once a model is dragged into the page, designer asks the developer whether to associate the models. The designer analyzes the relation between models and establishes an automatic relation when the constraints are known. Otherwise, developer has to manually set relations. It is possible to achieve multi-step associated layouts also, where Table1 is related to Table2, Table2 to Table3 and so on.

Remarks

Associating views can also be achieved between pages. This also ensures synchronization of data between models. Refer to table properties for more information.

For composite layouts, there should be at least one parameter that associates the main view with the associated views, else the results could be erroneous.

Navigation

Navigation feature enables the user to browse the records. This feature is functional in the following cases.

1. Get Web service operation returning more than one record.
2. Next and Previous methods are set in the model.

If the UI proposed by the XForms Designer does not have the element bar, you can add an element bar to the Groupbox or the grid and attach the Next and Previous methods to the model to get the navigation functional.

Find feature

All Web service operation-based forms that have parameters to retrieve data have the Find feature enabled by default. This option helps the user to find records from the current data set position. The functionality of the Find feature may vary depending on the type of Web service operation chosen and the layout proposed:

- Form layout: If there is only a Get<TableName>ObjectWeb service operation, Find works if the Web service operation has at least a parameter to retrieve data.
- Grid layout: Same as above.
- Other layouts: If the layout has association between the models, Find can work only if the Web service operation has a reference attribute and one or more additional parameters to retrieve data. For example, if Field1 of Table1 is a foreign key in Table2 and Field2 is primary key of Table2, then Find works when Table1 is associated to Table2.
- Non-AppWorks Platform protocol layouts: Find feature is not enabled.

Previewing XForms

When designing an XForm, you can preview and test your changes. An XForm that is opened using the Preview feature demonstrates run-time behavior. You can use this feature to retrieve and view data, and analyze the XForm functionality.

While previewing an XForm, you cannot view its translated version. Translated versions of XForms are available for viewing only at run time.

Also, you cannot view startup data in the preview of an XForm. Startup data is available only at run time, after you save the XForm and add it to the menu.

1. [Open the XForm](#) in the XForms Designer.

2. Click  (Preview) on the toolbar.

Alternatively, right-click the XForm in the Workspace Documents window and select **Show Preview** from the content menu.

The <XForm name> - Publish to Organization dialog box opens and the XForm is published.

The Preview window opens, displaying a preview of the selected XForm.

The HTML generated for the preview of the XForm is not cached in the application.

Setting properties of an XForm

You can define the properties of an XForm using its property sheet.

To set properties of an XForm:

1. [Open the XForm](#) in the XForms Designer.

2. Double-click the Designer Area or right-click it and select **Properties**.

Alternately, to view the property sheet of an XForm that contains split areas or regions, right-click the [Designer Area Well](#) and select **Properties**.

The Form window opens.

3. Click **Javascript** to expand the pane.

An empty row is displayed in the Javascript pane. You can add JavaScript files containing custom scripts to customize the XForm.

4. Type the URL (relative path) of the JavaScript file to be attached to the XForm in the empty row.
You can add multiple JavaScript files to the XForm and execute them like web page scripts. To do so, click  , and specify the respective URLs in the empty rows added to the pane.
5. Click Stylesheet to expand the pane, and click  (Add).
An empty row is added in the Stylesheet pane. You can specify custom style sheets to apply to the XForm.
6. Click  (Lookup).
The Select stylesheet - CSS dialog box opens.
7. Select a style sheet and click **OK**.
8. You can add multiple style sheets to the Stylesheet pane in the Form window. The style sheets are applied to the XForm in the order in which they are added.
9. Only the stylesheets that are uploaded to a project are available in the Select stylesheet - CSS dialog box. For information about uploading any document to a project, see [Uploading a Document to the Project](#).
10. Click **New** in the Select stylesheet - CSS dialog box to create a style sheet.
11. Define the stylesheet in the Untitled CSS.css window that opens and Save it.
12. If you edit a stylesheet in the project, you must publish it to ensure that the changes apply to the documents where it is used.
13. If the XForm is involved in a workflow, click **Workflow Events** and select an option from the Before CompleteTask list to specify the Web service operation to be executed before a task is executed.
14. Click **Feedback Properties** to alter the display settings of the error and notification messages.
These properties can be applied to input and select controls only:
 - Check **Show Error** to enable the error display for the input/select control.
 - Check **Show Notification** to enable the notification display for the input/select control.By default, both the properties will be checked.
15. Click **Events** and make the appropriate modifications:
 - Select an option from **Init** to specify the Web service operation to execute after the XForm is loaded and before a request is fired to the backend to retrieve data.
 - Select an option from **Init Done** to specify the Web service operation to execute after data is received from the backend and is rendered on the XForm.
 - Select an option from **Before Close** to specify the Web service operation to execute before the XForm is closed.

16. You can also use the Script Editor for associating data with the events of an XForm. To do so, click  (Script Editor) for each event and use the Script Editor to define the script.
17. Click **Form Messages** and specify the messages to be displayed when an XForm is closed.
18. In the Save Confirmation Message field, type the message to be displayed when the XForm is closed without saving its changes.
19. In the Synchronized Dialog Close field, type the message to be displayed when the XForm is closed without saving the changes made in a synchronized dialog.
20. Select **Set Focus at Startup** to specify that the first control on the XForm receives focus when the XForm is loaded.
The Set Focus at Startup check box is selected, by default
21. Check **Transitional** to enable multi-browser support for the XForm.
For more information, see [Enabling Multi-browser Support for Web Content](#).

The required settings for the XForm are defined.

To set properties for the task that is automatically created with an XForm, see [Task Properties Interface](#). You can access the Task Properties Interface from the XForms Designer by clicking  (UI Task Properties) on the toolbar.

Setting the positioning grid

The XForms Designer provides a grid to enable you to position controls in an XForm. The grid is a matrix that divides the Designer Area into smaller areas that act as points of reference while positioning controls. Using the grid, you can align user interface elements with precision.

To set the positioning grid:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the Designer Area and select **Show Grid**.
The grid is displayed in the Designer Area, and a check mark appears for the Show Grid option in the context menu.
The Show Grid context menu option is a toggle option, and selecting it again clears the check mark and hides the grid. On opening an XForm, the grid is enabled, by default.
3. Right-click the Designer Area, select **Grid Size**, and choose a grid size from the sub-menu.
The options available are 2, 5, 8, 10, 15, and 20, and are measured in pixels (px).
Depending on the option you select, the grid is resized. The grid size is set as 10, by default.
4. Right-click the Designer Area and select **Snap to Grid**.
This enables the snap-to-grid feature, and displays a check mark in the context menu.
The controls added to the XForm after enabling this feature are positioned automatically at the nearest reference point in the grid.

- The Snap to Grid context-menu option is a toggle option, and selecting it again clears the check mark and disables the feature.
- On opening an XForm, the snap-to-grid feature is enabled, by default.

The grid settings are saved as user preferences. The XForms that you open display according to the new settings.

Specifying layouts

Layouts define the manner in which UI elements are arranged in an XForm or in a grouping control. The default layout of an XForm and of grouping controls in an XForm is Vertical.

To specify layouts:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the grouping control or the Designer Area of the XForm, and select **Layout > Vertical, Horizontal, or Free**.

Depending on the option selected, the constituent controls are repositioned in the grouping control or the XForm.

Splitting XForms into regions

You can divide XForms into different regions. Regions are placeholders that host sub-applications known as App Palettes in an application. Regions can be further split vertically or horizontally, if required. A region can hold multiple App Palettes, which are displayed as tabs. Selecting a tab displays the respective App Palette.

An App Palette in a region can be further [split into areas](#) using the Split context-menu option. However, it is not possible to split an area into regions. Therefore, if you split an XForm into areas using the Split context-menu option, you cannot further add regions to the XForm.

To split XForms into regions:

1. [Open the XForm](#) in the XForms Designer.
2. To split the XForm, right-click it, select **Split As Regions**, and select the appropriate option from the sub-menu:
 - Select **Vertical** to split vertically.
 - Select **Horizontal** to split horizontally.The splitter bar appears, and the XForm is split into regions.

When you select Vertical, the splitter bar is placed to the left of the XForm, by default. When you select Horizontal, the splitter bar is placed towards the top of the XForm. You can drag the splitter bar to resize the areas. By default, each region holds an App Palette.

Also, on splitting an XForm, the controls are redistributed across the split XForm according to their respective positions and the XForm layout. In case of a vertical and

horizontal layout, all controls are repositioned to the left area when split vertically, and to the top area when split horizontally.

If a region contains multiple App Palettes, each App Palette is displayed as a tab in the tab strip.

3. Right-click the tab strip and select **Properties** to specify the properties of a region. The Region window opens.
4. Type a unique identifier for the region in **ID**. This is a design-time property.
5. Type the style class name to set the style class for the region in **Style Class**.
6. Select **Resizable** to enable the resizing of the region at run time.
7. Select the appropriate options in the **Tab Settings** pane:

Left	Position the tabs to the left side of the region.
Right	Position the tabs to the right side of the region.
Top	Position the splitter at the top of the region.
Bottom	Position the splitter at the bottom of the region.
Vertical	Display the text in the tab vertically.
Horizontal	Display the text in the tab horizontally.

These options are available only in case of multiple App Palettes. The selected options are applied at run time.

5. To customize the text that is displayed in an App Palette tab, select the default text and type your custom text.

The text that you enter in the tab is also displayed as title text for a static App Palette.

To remove a splitter:

- Right-click the splitter bar and select **Delete Splitter**.

Depending on options you select, the XForm is divided into regions.

Viewing scripts

The scripts associated with XForms or controls on an XForm help enhance run-time behavior.

To view scripts:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the control or XForm, and select **View Script**.
The Script tab opens to display the Script Editor that contains the script for the XForm.
In case of a control, the first event handler, if available, is automatically set to focus.

3. Alternatively, you can click  (Script Editor) for each individual event in the property sheet of the control.
The Script Editor opens to display the basic syntax of the script, which you can modify to define the event.
4. Save the XForm to save the edits made to the script.

Using XLets in XForms

Consider a scenario for additional XForm functionality in your Web application, in which you might want to dynamically insert functionality with minimal changes. AppWorks Platform XForms enables this by supporting the use of XLets.

XLets are XForms that contain the required, reusable functionality and are used as plug-ins in other XForms. XLets are plugged into an XForm using its URL.

To be able to use an XForm as an XLet, you must first create and publish the XForm. The URL of the published XLet can then be used in another XForm.

Appending XLets using the xlink parameter

To use an XLet in an XForm, you need to specify the `xlink` parameter in the URL. The syntax for specifying the `xlink` parameter is as follows:

```
xlink=<URL of XLet>,<position of XLet on XForm>
```

Where you can specify the position of the XLet on the XForm.

For example, consider an XLet with `/buttonSnippet.caf` as the URL, and an XForm (`anyForm.caf`) with `/someFolder/anyForm.caf?organization` as the URL.

To append the XLet to the bottom of the XForm, you can use the `xlink` parameter as follows:

```
/someFolder/anyForm.caf?organization='some  
org'&xlink=/buttonSnippet.caf, bottom
```

This will append the button snippet to any `Form.caf`.

You can also attach multiple XLets to an XForm as follows:

```
xlink=<URL of XLet1>,<position of XLet1 on XForm>; <URL of  
XLet2>,<position of XLet2 on XForm>; <URL of XLet3>,<position of XLet3 on  
XForm>
```

This will append the XLets to the main page in the order specified in the `xlink` parameter.

Note: Ensure that the names of all methods, event handlers, and controls are unique for each associated XLet and XForm.

Working with data

While using AppWorks Platform Web service operations and external Web services in an XForm, AppWorks Platform XForms automatically creates the appropriate user interface and data models. These data models send data requests to the backend and bind the retrieved data to appropriate controls in the XForm. However, in some cases you need to set and retrieve data manually:

- When data is already present in the client, such as an XML file, or data that is already received.
- When data is involved in a process workflow, and is a part of the previous form in the process. Here, the data is not from any specific backend source.

For information about the modes in which data is rendered in AppWorks Platform XForms, see [Transactional and Non-transactional Data](#). See [Referencing Transactional and Non-Transactional Data](#) for information about setting data for controls.

The following topics provide information about working with models:

- [Creating Models](#)
- [Adding and Associating Multiple Models](#)
- [Associating Existing Models](#)
- [Retrieving Data For Models](#)
- [Setting Data For Models](#)

Creating models

To create a model:

1. Open the [XForm](#) in the XForms Designer.
2. Click  (Manage Models) on the toolbar.
The Manage Models icon () is available only in the advanced toolbar of the XForms Designer. Check Advanced in the toolbar, to view it.
The Manage Models window opens.
3. Click the toolbar.
An empty row is added in the Manage Models window and the Model Properties dialog box opens.
4. Specify the required details in the various fields and tabs of the [Model Properties dialog box](#).
5. Click **OK**.

A new model is created.

You can also create a model programmatically using the `getModel()` Web service operation.

Adding and associating multiple models

While creating an XForm using AppWorks Platform Web service operations, you can add more than one model to the XForm and associate them using common parameters.

To add and associate multiple models:

1. [Create an XForm using a Web Services](#).

The UI that is generated from the Web service operation is displayed in the designer area.

2. Drag the second Web service operation to the **XForm** from the

WebserviceBindingOperation group in the **Insert** tab.

The Generate Input and Output UI dialog box opens.

3. Select **Generate UI for Input message** to generate an input UI.

4. Select **Generate UI for Output message** to generate an output UI.

For details about generating UI, see [Generating Input and Output User Interfaces](#).

5. Click **OK**.

The Confirm dialog box opens. It lists the models that already exist in the XForm, and prompts you to confirm if you want to associate the new model with any of the existing models.

4. Select the model with which you want to associate the new model.

The Single Transaction check box appears in the Confirm dialog box. Also, the  (Success) or  (Warning) images appear for the selected model to indicate if the association will be successful.

- The Single Transaction option is also available in the Model Properties dialog box.
- Association between models is based on the availability of common parameters between the models.  (Success) indicates that the models contain common parameters and hence their association will be successful.  (Warning) indicates that the models do not contain any common parameters and hence their association will not be successful.

5. Select **Single Transaction**.

- Single transaction is not supported for WS-AppServer models or models associated with WS-AppServer models. The Single Transaction check box is disabled when you select a WS-AppServer model for association.
- The Single Transaction option is also available in the [Model Properties dialog box](#).

This enables the application to save changes made to both the associated models in a single transaction at runtime.

In cases where auto-generated parameters from the parent model are required to be specified in the child model, enabling the Single Transaction option ensures that values are automatically specified and saved for the child model. The required parameter for

the parent model is first auto generated and saved, and then automatically specified for the child model and saved. All this is done in a single transaction with the backend.

6. Click **Yes** to associate the models.
7. Click **No** to avoid the association.

The models are associated as specified and the UI of the new model is displayed in the Designer Area.

- You can create associations between transactional models only. Associations between non-transactional models are not supported.
- You must ensure that the models to be associated comprise common connectors.
- The `updateDocument` event object property of the `onbeforesynchronize` event can be used to modify the data of the corresponding model only. OpenText recommends that you do not modify `updateDocument` for all associated models in case of single transaction.

Associating existing models

The models in an XForm are the components that hold and manage data. A model manages data requests, stores the response data, and synchronizes the changes made to the data with the backend.

If multiple models in an XForm are interdependent for data inputs such that the values from one are used as the input for another, you can associate the models.

In XForms with associated models, when you specify a value as the input for a field, the application will automatically retrieve and display other related information. This is possible as, at the backend, the models in the XForm are associated using common parameters that enable the exchange of data, and thus define the relationship between the models.

Before you begin:

- You can associate the models in an XForm if it contains more than one model.

To associate existing models:

1. Open the [XForm](#) in the XForms Designer.
2. Add a Web service operation to the XForm.
See [Creating XForms using Web Services](#) for details. A Web service operation is added to the XForm and the relevant interface is generated.
3. Click  (Manage Models) on the toolbar.
The Manage Models icon () is available only in the advanced toolbar of the XForms Designer. Select **Advanced** in the toolbar, to view it.
The Manage Models window opens.
4. Click  (Add).
An empty row is added in the Manage Models window and the Model Properties dialog box opens.

5. Select an option from **Parent Model** on the **Associations** tab to specify the model with which the current model is to be associated.

Restriction

- It is not possible to associate a model with itself.
- Models that do not have parameters available for association cannot be associated with other models.

The parameters available for the selected model are automatically displayed in the Request Parameter column.

6. For each parameter, click  (Lookup) in the Field column to display the Model References dialog box, and select an option to specify the field that is available for association in the current model.
7. For each parameter, click  (Lookup) in the Parent Model Field column to display the Model References dialog box, and select an option to specify the field of the parent model with which to associate.

The Parent Model Field column is editable only if a parent model is specified.

8. Click **OK**.

The details of the associated models are saved.

Retrieving data for models

If the data retrieval for a model is set to manual in the [Model Properties dialog box](#), or if no dataset is available, that is the request is not auto-generated, you can use the following API to manually retrieve data from the backend.

```
var request = dataRequest.XMLDocument;
newModel.setMethodRequest(request);
newModel.getDataset();
```

You can also use these API to set the cursor-related methods, such as next and previous, by passing the respective parameters. You can use the `getDataset()` method for a method of the type `get`. If all cursor methods are used, you can use the `reset()` method directly. This method identifies the request type and sends the request accordingly.

Setting data for models

You can bind data to controls using the following:

- The `putData()` method
- Process APIs

Setting data using the `putData()` method

When the `reset()` or `getDataset()` methods are used to set data, the data is retrieved and bound to the respective controls. However, if data is present as XML, use the `putData()` method to set the XML to the data model.

After setting the XML data for the model, you can render the data to controls using the following:

```
newModel.putData(request, true);
```

Setting data using process APIs

Sometimes, as part of a workflow, manual intervention is needed, which necessitates the display of an interface for decision-making. In such a workflow, you can use XForms as screens that display options for decision-making. To use an XForm in a workflow, save it and publish it to the organization. You can drag the published XForm to a business process model to use it in the workflow.

To enable the participation of controls in a workflow, AppWorks Platform XForms provides a set of process APIs for the controls, at run time. It is possible to pass data through a workflow by following certain predefined data formats.

You can use process APIs to pass data as follows.

By passing parameters: Parameters can be passed to an XForm using the `<request>` node. The `<request>` node can be created inside the `<data>` node in the application definition.

The following sample application definition retrieves data from the EmployeesModel model.

```
<data>
<forminputdata xmlns="http://schemas.cordys.com/1.0/xforms/processapi">
<EmployeesModel type="get">
<GetEmployeesObject xmlns="http://schemas.cordys.com/NorthwindMetadata/Employees">
<EmployeeID>5</EmployeeID>
</GetEmployeesObject>
</EmployeesModel>
</forminputdata>
</data>
```

Important: The following way of passing parameters is now DEPRECATED. Please use as given in the example above.

```
<Application>
<url>/Employees.caf</url>
<id>uniqueid</id>
<frame>main</frame>
<description>Employees</description>
```

```

<caption>Employees</caption>
<data>
<request model="EmployeesModel">
<dataset type="get">
<EmployeeID>5</EmployeeID>
</dataset>
</request>
</data>
</Application>

```

Where:

- The model attribute is mandatory and refers to the name of the model from which data is to be retrieved.
- The type property refers to the type of method request. The method request can be of the type - get, next, or previous.

By passing data: You can also pass data directly to a specific model in an XForm. The data will be picked by the model, and rendered into a control, according to set references.

In the following example, the <data> node inside application definition is set for the fifth Employee.

```

<data>
<forminputdata xmlns="http://schemas.cordys.com/1.0/xforms/processapi">
<EmployeesModel type="put">
<GetEmployeesObjectResponse
xmlns="http://schemas.cordys.com/NorthwindMetadata/Employees">
<tuple>
<old>
<Employees>
<EmployeeID>5</EmployeeID>
<FirstName>Steve</FirstName>
<LastName>Buchanan</LastName>
</Employees>
</old>
</tuple>
</GetEmployeesObjectResponse>
</EmployeesModel>
</forminputdata>
</data>

```

Important: The following way of passing data is now deprecated. Please use the example above.

```

<data>
<request model="EmployeesModel">
<EmployeesResponse>
<tuple>

```

```

<old>
<Employees>
<EmployeeID>5</EmployeeID>
</Employees>
</old>
</tuple>
</GetEmployeesResponse>
</request>
</data>

```

In the above example, if the optional attribute `create` is set to true on the `<request>` node, then the specified model, `EmployeesModel`, will be created and data set on it.

The following example demonstrates passing data based on a business object name:

```

<data>
<Request>
<Employees namespace="http://schemas.cordys.com/1.0/demo/northwind">
<PutData>
<GetEmployeesResponse>
<tuple>
<old>
<Employees>
<EmployeeID>2</EmployeeID>
</Employees>
</old>
</tuple>
</GetEmployeesResponse>
</PutData>
</Employees>
</Request>
</data>

```

In the above example, a `<request>` tag containing the business object name is used, by which remove dependency on model name.

It is also possible to send initial data values for free-form controls in an XForm, as displayed in the following example. Here, the value '5' is passed to the `<input1>` input control in the `<data>` node.

```

<data>
<forminputdata xmlns="http://schemas.cordys.com/1.0/xforms/processapi">
<freeformcontrols>
<input1>5</input1>
</freeformcontrols>
</forminputdata>
</data>

```

Important: Passing an action to set data is now deprecated.

By passing an action: Some scenarios in workflow processes demand XForms to open in a specific action mode, for example creating a new registration form requires the XForm to open in the Insert mode.

The following sample shows how to open the Employees data table or group in the insert mode.

```
<data>
<Action>
<Employees namespace="http://schemas.cordys.com/1.0/demo/northwind">
<Insert/>
</Employees>
</Action>
</data>
```

Transactional and non-transactional data

Using AppWorks Platform, you can design XForms for use in an integrated environment, where multiple web applications display live data and that supports integration and communication between various applications. As compared to standard HTML applications, web applications render data from various backend sources and constantly synchronize with the server to update to the latest, live data.

XForms is capable of rendering data in two modes:

- Transactional data
- Non-transactional data

These data can be rendered through Web services (such as AppWorks Platform Web service operations or external Web services) and process API.

Transactional data

Data that conforms to the AppWorks Platform protocol is classified as Transactional Data. Each data object in Transactional data represents an object that can be used for transactions such as creating, editing, and synchronizing with the backend.

The following sample illustrates the transactional data protocol in AppWorks Platform:

```
<tuple>
<old>
<Employees>
<EmployeeID>1</EmployeeID>
</Employees>
</old>
</tuple>
```

For Transactional data, each data object (such as 'Employees' in the above sample) is wrapped in a <tuple> protocol, which is a AppWorks Platform standard, and contains the state of the transactional object. The following are some transaction formats used in AppWorks Platform:

Format	Description
<tuple> <new>...</new> </tuple>	This format is used when data is created in the client. If the format also carries the sync_id attribute, it denotes that the data is yet to be synchronized with the backend.
<tuple> <old>...</old> </tuple>	This format is used when data is retrieved from backend.
<tuple> <old>...</old> <new>...</new> </tuple>	This format is used when data is updated in the server. If the format also carries the sync_id attribute, it denotes that the data is yet to be updated to the backend.

Transactional data provides the following advantages:

- Using transactional data, it is possible do all transactions such as creation, update, deletion, and synchronization of records from the backend. Any change in the data is recorded in the above-mentioned formats.
- While closing a form, the application displays prompts containing various options to save the data.
- In cases where a large number of records (business objects) are available, you can specify a 'cursor' or iteration size to indicate the number of records to be retrieved from the backend. For information, see [Generation of Layouts from Methods](#).
- The pagination feature is supported for transactional data. By specifying a 'cursor', it is possible to retrieve data in batches and paginate the XForm batch-wise.

The pagination feature enables you to access data in a Table control. However, the manner in which rows display in the Table control varies according to the kind of model rendered in it. For a transactional model, the table view contains a fixed set of rows, and empty rows are displayed if number of rows exceeds the available data. For a non-transactional model, the table view does not contain a fixed set of rows, and no empty rows are displayed if data is not available.

Non-transactional data

Data that does not confirm to the AppWorks Platform standard protocol is non-transactional Data. Non-transactional data can be any of the following:

- Data from Java Call services
- Data from an external Web service
- Any custom data (such as XML data)

The following sample illustrates non-transactional data:

```
<items>
<item>
<id/> ...
</item>
<item>
<id/> ...
</item>
</items>
```

This model is rendered in the same manner as Transactional Data. Also, changes made to the data are tracked internally, such that you can insert, delete, and modify data in the client. However, synchronization and saving data to the backend is not possible.

Non-transactional data has the following characteristics:

- While it is possible to modify data in the client, the changes cannot be saved to the backend.
- When an XForm is closed, no prompts are displayed to save the data.
- It is not possible to define a cursor or iteration size to display records in batches.
- The pagination feature is supported for non-transactional data. The pagination bar is enabled if the number of data objects exceeds the height of the table.

The pagination feature enables you to access data in a Table control. However, the manner in which rows display in the Table control varies according to the kind of model rendered in it. For a transactional model, the table view contains a fixed set of rows, and empty rows are displayed if number of rows exceeds the available data. For a non-transactional model, the table view does not contain a fixed set of rows, and no empty rows are displayed if data is not available.

You can display non-transactional data in a Table control by specifying the XPath of the data, for example response/methodName/businessObject. Here, data from each businessObject node will be displayed in a separate row in the table.

Binding data to data models

For AppWorks Platform Web services or external Web services, XForms automatically generates the user interface and binds data to respective controls. Depending on the data model used, the following types of models are created:

- **Transactional Model:** This model is generated in case of transactional data. This model accepts the AppWorks Platform Web service operations (for Next and Previous) optionally.
- **Non-transactional Model:** This model is generated in case of non-transactional data and renders non-transactional data only.

While dragging and dropping a Web service operation to add it to an XForm, XForms automatically assigns the model type, binds the HTML controls to specific references from the data objects, and renders data accordingly. For information about data binding and

setting references, see [Binding Data to Controls](#). For information about referencing data, see [Referencing Transactional and Non-transactional Data](#).

Referencing transactional and non-transactional data

By setting references between controls and models, you can automate the display of a model's data in a control. Setting a reference associates a control and a model such that data is automatically synchronized between the two. This means that you can use the control to add, delete, or modify data in the backend, and that the data displayed in the control is automatically updated with any change in the backend data.

You can use the <control name> window to set references between a control and a model as described in [Binding Data to Controls](#). As mentioned in the topic, to bind data, you must specify a Model and give a Reference to the business object (of the model) from which data is to be displayed in the control. However, the Reference XPath to be specified differs depending on the type of data (Transactional and Non-transactional) used in the model.

Consider the structure of Transactional data (EmployeesModel) as follows:

```
<tuple>
<old>
<Employees>
<EmployeeID>1</EmployeeID>
</Employees>
</old>
</tuple>
```

For Transactional data, you need only to specify the model name (EmployeesModel) in the Model field and the business object name (EmployeeID) in the References field of the <control name> window. The AppWorks Platform automatically picks data from the appropriate nodes of the response. You do not need to define the complete path of the business object as the reference, as it is automatically picked from the tuple/old/Employees structure.

Also, if the same model is associated with a control and its parent control, then the control is not considered as an independent view. It is considered as part of parent control view and renders data as part of the view.

Consider the following Non-transactional data that does not follow AppWorks Platform protocol:

```
<items>
<item>
<id/> ...
</item>
<item>
<id/> ...
</item>
</items>
```

For Non-transactional data, the data starts from the first node, in this case the <items> node. The <item> node is repetitive and contains the <id> node (business object). Values from the <id> node are displayed in a control. The first node, from which the data starts, is not considered while referencing.

For Non-transactional data, you need to specify the model name in the Model field and the complete XPath of the business object in the References field of the <control name> window. This is also true for transactional models for which the Non-transactional option is selected in the Model Properties dialog box. Also, a complex XML, which is a response that contains multiple level of nodes and comprises various data under different nodes, is also considered as Non-transactional data even if it is compliant with the AppWorks Platform protocol.

In the above non-transactional code sample, the XPath for the Reference will be item/id.

As a special case, consider setting references on a Table control such that data is displayed under columns and each set of related data is displayed in a separate row. In this case, you can set the Model and the Reference XPath of the repetitive node (in the above sample codes, Employees and item are the repetitive nodes) on the table (parent control).

Depending on the data type - Transactional and Non-transactional - you need to specify the appropriate Reference XPaths. In order to display data in a table column, specify the respective business object name (in the above sample codes, EmployeeId and id) as a Reference for the column. This displays model data in columns in the table.

Note: Prefilling List and Select Controls also involves specifying a model from which to display data in the controls at run time. Ensure that the model used to bind data to these controls is not the same as the model used to prefill them.

Consider another example of complex data, where the tuple (in case of Transactional data) or first node (in case of Non-transactional data) contains a set of repeating child nodes.

```
//complex nested transactional data
<tuple>
  <old>
    <Employees>
      <EmployeeID>1</EmployeeID>
      <EmployeeName>anna</EmployeeName>
      <Orders>
        <OrderID>1</OrderID>
        <OrderDate>03092009</OrderDate>
      </Orders>
      ...
      <Orders>
        <OrderID>5</OrderID>
        <OrderDate>03102009</OrderDate>
      </Orders>
      ...
    </Employees>
  </old>
</tuple>
```

```
//complex nested non-transactional data
<Employees>
  <EmployeeID>1</EmployeeID>
  <EmployeeName>anna</EmployeeName>
  <Orders>
    <OrderID>1</OrderID>
    <OrderDate>03092009</OrderDate>
  </Orders>
  ...
  <Orders>
    <OrderID>5</OrderID>
    <OrderDate>03102009</OrderDate>
  </Orders>
  ...
</Employees>
```

When represented in an XForm, the interface will include controls for nodes such as EmployeeID and EmployeeName, while data from the repeating child nodes (in the above sample codes, Orders) will be represented in a paginated table. The pagination toolbar of the table can be used to navigate between records. In order to reference such child nodes, you need to provide the parent view (in the above sample codes, Employees) in the property sheet of the Table control.

Even though the data is nested, it is possible to create UI representations that are not nested. To achieve this, you can use the parent view property in the property sheet and design UI such that child views display outside parent views. For details on setting the Parent for a child view, see [Binding Data to Controls](#).

Working with composite controls

AppWorks Platform XForms provides you with a comprehensive list of controls that can be used to create applications. In some cases, while developing an application you may need to use a combination of controls or a specific control that is not available as a standard XForm control.

To facilitate such specific requirements, AppWorks Platform provides you with a Composite Control Framework using which you can build your own controls. The controls built using this framework are called Composite Controls.

Composite controls can be built using existing libraries and can be used to extend the behavior of existing controls or group of controls. Using the Composite Control Framework, you can define and add a run-time behavior to composite controls, enabling you to combine often-used functionality that can be reused in all your applications. You can also create composite controls from REST URLs.

This section caters to three different scenarios of composite control design.

- [Creating a composite control](#): This section gives a step-by-step procedure to create a composite control.

- [Creating a composite control using existing XForms controls](#): This topic guides you through the creation of a composite control that uses XForms controls.
- [Creating a composite control using a REST URL](#): This topic describes the procedure to create a composite control using a REST URL.

The first two procedures are example-based and lead to the creation of usable sample composite controls.

This section also provides information about the basic elements that are required to build any control.

- [Elements of a Design-time Type Library](#)
- [Elements of a Run-time Type Library](#)
- [Templates of a Composite Control](#)

You can refer to these topics while creating your own controls.

Creating a composite control

The steps in this topic guide you to create the design-time library of a basic composite control that can be dragged to an XForm and used as is.

As an example, consider creating a Hyperlink control that, when added to an XForm, adds a hyperlink to it. At run time, clicking the hyperlink opens the specified URL in a new window. You will be able to modify the hyperlink text and URL during design-time.

To create a composite control:

1. [Access the Composite Control Modeler](#) () to create a composite control.
The Untitled Composite Control - Composite Control wizard opens.
2. Type the **Name** of the composite control.
3. Type a **Description** for the composite control.
4. Select the **Control** Control Type option to provide the design-time and run-time libraries based on which the composite control must be created.
The REST URL Control Type option is used to create URL-based composite control.
The title of the wizard changes to <composite control> - Composite Control.
5. In Source XForm, click (Edit) to create an XForm that defines the composite control.
Alternatively click (Lookup).
The Select Source XForm dialog box opens.
6. Specify an existing XForm.
7. Regenerate the Design Time Library if the Source XForm is modified.
8. In the Design Time Library field, click (Edit) to create a design-time library.
The Confirm dialog box opens.
9. Click **Yes**.
The <design-time library>.htm - Htm dialog box opens

- Alternately, click  (Lookup).
The Select Design Time Library dialog box opens.
 - Update the location of `setPublic` and `inherit` in the Design Time Library if it is moved to another location after generation.
 - When working on the Linux OS, ensure that the names of the libraries are in lower case.
10. In the <design-time library>.htm - Htm dialog box, type the name of the design-time library in the Name field.
In this case, you can name the design-time library as hyperlink.
11. Type a description for the design-time library in the Description field.
12. In the Location field, select  (Edit).
The Select Folder dialog box opens,
13. Specify the location in which the library must be saved.
The <design-time library>.htm - Htm dialog box displays the following standard code structure using which you can build your design-time library. This template is automatically generated by the XForms Designer.

```
<!-- CompositeControlDT design time control --> <html
onapplicationready=""><head> <script type="text/javascript"> setPublic
(CompositeControlDT, "<automatically generated folder structure for
designtimeLibrary>"); importType
("cas.xforms.designerlibrary.controls.XFormsControl"); inherit
(CompositeControlDT, XFormsControl); CompositeControlDT.icon =
"/cordys/wcp/theme/default/icon/document/compositecontrol.png"; // This is
the constructor of the composite control // It calls the base class's
constructor function CompositeControlDT(designerView) { this.XFormsControl
(designerView); } </script> </head><body></body></html>
```

In the above code, `designtimeLibrary` refers to the location where the design-time library is stored.

6. Modify the standard code to define the required design-time functionality for your control and click  (Save).
The design-time library for the control is created.
7. Repeat Steps 5 to 7 to similarly create a run-time library for the composite control. The standard code structure that displays for the run-time library is as follows. This template is automatically generated by the XForms Designer.

```
<!-- CompositeControlRT runtime control --> <html
onapplicationready=""><head> <script type="text/javascript"> setPublic
(CompositeControlRT, "<automatically generated folder structure for
runtimeLibrary>"); importType
```

```
("cas.xforms.runtimelibrary.CompositeControl"); inherit  
(CompositeControlRT, CompositeControl); function CompositeControlRT() { }  
function CompositeControlRT.attachType( control ) { //TODO: add attach  
code here } function CompositeControlRT.detachType( control ) { //TODO:  
add detach code here } </script> </head><body></body></html>
```

In the above code, `runtimelibrary` refers to the location where the run-time library is stored.

The run-time library for the control is created.

To create a property sheet for the composite control:

1. In Property Sheet, click  (Edit).
The Confirm dialog box opens.

2. Click **Yes**.

Selecting **Extend Properties** creates a property sheet extending the property sheet of the design time control from which this composite control is created. Selecting or clearing this check box is not persisted.

Alternately, click  (Lookup) to specify an existing property sheet from the Select Property Sheet dialog box that opens. The XForms Designer opens and the <property sheet> - XForm is displayed. In addition to some XForm controls, the Toolbox tab displays the common controls used in a property sheet such as Date, Event, Id, Model, Reference, Web Service, and XML. For information about these, see [Controls Available for Creating Property Sheets](#).

2. Drag the required controls from the Toolbox tab to design a property sheet for the control.

In case of the Hyperlink composite control, the property sheet will contain an **Id** field, a **Text** field to input the text to be displayed as a link, and a **Link** field to input the URL to be used as a link.

- Drag the **ID** control to the XForm and provide a unique **Id**.
- Drag an **Input** control, rename it to **Text**, and provide a unique **Id**.
- Type **@textId** in the **Reference** field.
This reference is used in the run-time library to set text values at runtime.
- Drag another **Input** control, rename it to **Link**, and provide a unique **Id**.
- Type **@linkId** in the **Reference** field.
This reference is used in the run-time library to set URL values at runtime.
- Add the **onChange** event to the **Text** and **Link** **Input** controls, and specify the following in the auto-generated function (in the **Script** tab).

```
htmlElement.control.updateText(htmlElement, eventObject.srcElement.getValue());
```

3. Click .

The property sheet for the control is created and saved in the XML Store.

If you do not create a property sheet for the composite control, a default property sheet is automatically added to the control. This property sheet contains only the ID field.

4. In the Schema tab, enter the XML schema to be used for mapping the control using a message map.

For more information, see [Using Message Map](#).

5. Click  in the <composite control> - Composite Control dialog box.

The composite control is created and displayed in the Workspace Documents window.

6. You can drag the composite control to an XForm to use it.

Alternately, you can right-click the XForm, select **Insert Composite Control**, and drag the required composite control from the Composite Controls dialog box that opens. This right-click menu option is available for a split area, an XForm, and for the Group, Groupbox, and Tab Page grouping controls.

AppWorks Platform is multiple browser compliant. For composite controls created in BOP-4 that are now ported to Cordys BOP 4.1 and need to be made multiple browser compliant, you must ensure multiple browser support for the same.

- If no custom modifications were made to the design-time and runtime libraries of the composite control, you can delete the libraries and generate new ones in Cordys BOP 4.1.
- If you had made custom modifications to the design-time and runtime libraries of the composite control, you must manually provide custom script for multiple browser support. For details, refer to [Enabling Multi-browser Support for Web Content](#).

Before building the package:

- Create Web Library Definition for composite controls in the libraries folder and ensure that the design-time and run-time libraries are packed along with the application.

Creating a composite control using existing XForms controls

The composite control framework enables you to use the controls available in the XForms Designer and create a composite control using them.

If your composite control comprises a group of controls, you must always use a grouping control (the Groupbox, TabGroup, Group, or Table control) as the parent control, while designing the composite control.

Consider creating a Effective Interest Calculator control that combines multiple XForms controls and, at runtime, provides you with an output based on the values you provide. This composite control calculates the effective annual interest rate based on a given nominal interest rate and the number of times it is compounded annually. You will be able to specify the nominal interest rate and number of compounding periods at runtime.

To create a composite control using existing XForms controls:

1. Open a new XForm and design the interface of the control in the XForms Designer. Change the label of each control as given below. Also, modify the ID and Tooltip in the property sheet of each control.

Control	Change label to	Change Id to	Add tooltip text
Groupbox	Effective Interest Calculator	-	-
Input	Annual Interest Rate	interestRate	Enter the annual interest rate
Input	Number of Compounding Periods	compoundingPeriods	Enter the number of compounding periods
Button	Calculate Effective Interest Rate	calculatebtn	Click to calculate effective annual interest rate
Output	Effective Annual Interest Rate	effectiveRate	Effective annual interest rate

The interface of the control is modified as shown below.

2. Right-click the Groupbox control and select **Save as Composite Control**. The UntitledCompositeControl.Htm dialog box opens, where you can provide details regarding the design-time library of the composite control.
3. Make appropriate modifications in the UntitledCompositeControl.Htm dialog box.
 - Type the name of the design-time library in the Name field. In this case, you can name the design-time library as **EffectiveInterestCalculator**.
 - Type a description for the design-time library in the Description field.
 - In the Location field, select and specify the location at which the library must be saved in the Select Folder dialog box that opens. The title of the dialog box changes to <design-time library name>.htm.
4. Click **Save**. The Untitled Composite Control - Composite Control wizard opens.

The design-time library, created above, is displayed in the Design Time Library field.

5. Make appropriate modifications in the Untitled Composite Control - Composite Control wizard.

- Type the name of the composite control in the Name field.
- Type a description for the composite control in the Description field.
- Select the **Control** Control Type option to provide the design-time and run-time libraries based on which the composite control must be created. This option is selected by default.

The REST URL Control Type option is used to create URL-based composite control.

The title of the wizard changes to <composite control> - Composite Control.

6. In the Design Time Library field, click  (Edit) to view the code generated for the design-time library.

Alternately, click  (Lookup) to specify another existing design-time library from the Select Design Time Library dialog box that opens.

The <design-time library>.htm - Htm dialog box displays the following auto-generated code.

```
<!-- EffectiveInterestCalculator design time control --> <html><head> <script
type="text/javascript"> //retain the setPublic() method as shown for your composite
control setPublic(EffectiveInterestCalculator1, "designtimeLibrary"); //provide the
namespace of the group element as defined in the wcpforms:designLibrary attribute in
the markup of your composite control importType
("cas.xforms.designerlibrary.controls.XFormsGroupbox"); //provide the group element
from which this composite control is inherited inherit(EffectiveInterestCalculator1,
XFormsGroupbox); //inherit(); EffectiveInterestCalculator1.icon =
"/cordys/wcp/theme/default/icon/document/compositecontrol.png"; // This is the
constructor of the composite control // It calls the base class's constructor
function EffectiveInterestCalculator1(designerView) { this.XFormsControl
(designerView); } </script> <script xmlns="http://www.w3.org/2002/xforms/cr"
type="cordys/xml" id="_controlDefinition" > <xforms:group
xmlns:xforms="http://www.w3.org/2002/xforms/cr" appearance="box"
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsGroupbox"
xmlns:wcpforms="http://schemas.cordys.com/wcp/xforms" wcpforms:layout="vertical"
wcpforms:class="v_layout" wcpforms:size="null null" wcpforms:labelalign="ontop" >
<xforms:label wcpforms:class="groupheader" >Effective Interest
Calculator</xforms:label> <xforms:input
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsInput"
id="interestRate" doEBIValidate="false" wcpforms:class="v_layout" > <xforms:label
wcpforms:class="v_label" >Annual Interest Rate</xforms:label> <xforms:hint>Enter the
annual interest rate</xforms:hint> </xforms:input> <xforms:input
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsInput"
id="compoundingPeriods" doEBIValidate="false" wcpforms:class="v_layout" >
<xforms:label wcpforms:class="v_label" >Number of compounding periods</xforms:label>
<xforms:hint>Enter the number of compounding periods</xforms:hint> </xforms:input>
```

```

<xforms:group appearance="controlbox"
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsControlbox"
id="controlbox1" controltype="button" wcpforms:layout="horizontal"
wcpforms:class="v_layout" wcpforms:align="center_align" wcpforms:size="null auto" >
<xforms:trigger
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsButton"
id="calculatebtn" wcpforms:class="h_button" wcpforms:size="230px 2em" >
<xforms:label>Calculate Effective Interest Rate</xforms:label> <xforms:hint>Click to
calculate effective annual interest rate</xforms:hint> </xforms:trigger>
</xforms:group> <xforms:output
wcpforms:designLibrary="cas.xforms.designerlibrary.controls.XFormsOutput"
id="effectiveRate" doEBIValidate="false" wcpforms:class="v_layout" > <xforms:label
wcpforms:class="v_label" >Effective Annual Interest Rate</xforms:label>
<xforms:hint>Effective annual interest rate</xforms:hint> </xforms:output>
</xforms:group> </script> </head><body></body></html>

```

The code contains markup of the interface designed for the Effective Interest Calculator control. In the above design-time composite control:

- The `setPublic()` method gives the path where the control is saved (in below example, it's design-time Library). This may vary in case of your composite control. While using the below code to create your own composite control, ensure that the code that is auto-generated for `setPublic()` method is retained.
 - The namespace of the group element is provided in the `importType()` method as defined in the `wcpforms:designLibrary` attribute in the markup of your composite control. For example, in this case, the value of the `wcpforms:designLibrary` attribute is `cas.xforms.designerlibrary.controls.XFormsGroupbox`.
 - In the `inherit()` method, the group element from which your control is inherited is provided, as specified in the `wcpforms:designLibrary` attribute in the markup. For example, in this case, the group element is XForms Group box.
7. In the Run Time Library field, click to create a runtime library and click **Yes** in the Confirm dialog box that is displayed.
Alternately, click  (Lookup) to specify an existing runtime library from the Select Run Time Library dialog box that opens. The `<run-time library>.htm - Htm` dialog box opens.
 8. Make appropriate modifications in the `<run-time library>.htm - Htm` dialog box.
 - Type the **Name** of the run-time library.
In this case, you can name the run-time library as **EffectiveInterestCalculator**.
 - Type a **Description** for the run-time library.
 - In **Location**, select and specify the location at which the library must be saved in the Select Folder dialog box that displays.
The `<run-time library>.htm - Htm` dialog box displays the standard code structure using which you can build your run-time library.

9. Add the following code to create a run-time library for the Effective Interest Calculator control.

In the following example, the setPublic() method gives the path where the control is saved (in below example its runtimeLibrary). This may vary in case of your composite control. While using the below code to create your own composite control, ensure that the code that is auto-generated for setPublic() method is retained.

```
<!-- EffectiveInterestCalculator runtime control -->

<html><head> <script type="text/javascript"> setPublic(EffectiveInterestCalculator,
"runtimeLibrary"); importType("cas.xforms.runtimelibrary.CompositeControl");
importType("cas.xforms.runtimelibrary.ContainerElement"); inherit
(EffectiveInterestCalculator, CompositeControl); inherit
(EffectiveInterestCalculator, ContainerElement); function
EffectiveInterestCalculator() { } function EffectiveInterestCalculator.attachType(
control) { // Mandatory // Call the attachType of superClass. application.addType
(control, "cas.xforms.runtimelibrary.ContainerElement"); // Store the references of
controls inside the Effective Interest Calculator control. // Attach event handlers
as required. control.interestRate = control.getElementById("interestRate");
control.interestRate.onblur = validateFieldValue(control.interestRate, control);
control.compoundingPeriods = control.getElementById("compoundingPeriods");
control.compoundingPeriods.onblur = validateFieldValue(control.compoundingPeriods,
control); control.calculatebtn = control.getElementById("calculatebtn");
control.calculatebtn.onclick = calculateInterestRate(control); control.effectiveRate
= control.getElementById("effectiveRate"); } function
EffectiveInterestCalculator.detachType(control) { // Mandatory // Call the
detachType method of the superClass. application.removeType(control,
"cas.xforms.runtimelibrary.ContainerElement"); // Clear all the instance variables
control.interestRate = null; control.compoundingPeriods = null; control.calculatebtn
= null; control.effectiveRate = null; } function validateFieldValue(htmlElem,
effectiveinterestcalculator) { return function() { var value = parseFloat
(htmlElem.value); // If value of the field is not a number or 0, put a
effectiveRate. if (isNaN(value) || value == 0) {
effectiveinterestcalculator.getApplication().notify("Enter valid value."); } } }
function calculateInterestRate(effectiveinterestcalculator) { return function() { // Parse
the specified values var interestRate = parseFloat
(effectiveinterestcalculator.interestRate.value); var compoundingPeriods =
parseFloat(effectiveinterestcalculator.compoundingPeriods.value); var resultField =
effectiveinterestcalculator.effectiveRate; var errorMsg = ""; if(isNaN(interestRate)
|| isNaN(compoundingPeriods)) errorMsg = "Enter numeric values"; else if
(interestRate <= 0) errorMsg = "Enter valid annual interest rate"; else if
(compoundingPeriods < 1) errorMsg = "Enter valid number for compounding periods"; if
(errorMsg) { effectiveinterestcalculator.getApplication().showError(errorMsg);
return; } // Calculate the effective interest and round it off to 2 decimal places.
var result = 1 + (interestRate/compoundingPeriods); result = new Number(Math.pow
(result, compoundingPeriods) - 1); result = result.toPrecision(4);
effectiveinterestcalculator.effectiveRate.value = result; } } </script>
</head><body></body></html>
```

Note that since the Effective Interest Calculator control is essentially a grouping control (in this case, a Groupbox control), it needs to extend from the Container Element library, which is a run-time representation of all the grouping controls. This is done using the `importType` and `inherit` statements in the above code.

By doing this, the `getElementById()` Web service operation of the container Element library can be used to access individual controls within the Groupbox.

10. Click in the <composite control> - Composite Control dialog box.

The composite control is created. The Effective Interest Calculator composite control is now ready for use. You can enter values in the control at runtime, and calculate the effective interest rate.

Both the composite control and the design-time library of the composite control display in the Workspace Documents window. You can drag the composite control to an XForm to use it. Alternately, you can right-click the XForm, select Insert Composite Control, and drag the composite control from the Composite Controls dialog box that displays. This right-click menu option is available for a split area, an XForm, and for the Group, Groupbox, and Tab Page grouping controls.

You can delete the composite control from the Workspace Documents window. This does not automatically delete the design-time library associated with the control, enabling you to reuse the design-time library with other composite controls.

AppWorks Platform is multiple browser compliant. For composite controls created in BOP-4 GA that are now ported to Cordys BOP 4.1 and need to be made multiple browser compliant, you must ensure multiple browser support for the same.

- If no custom modifications were made to the design-time and runtime libraries of the composite control, you can delete the libraries and generate new ones in Cordys BOP 4.1.
- If you had made custom modifications to the design-time and runtime libraries of the composite control, you must manually provide custom script for multiple browser support. For details, see [Enabling Multi-browser Support for Web Content](#).

Creating a composite control using a REST URL

In addition to manually creating a composite control using design-time and run-time type libraries, you can also create a composite control from a REST URL. Representational State Transfer or REST refers to an architecture in which possible interactions are represented by URLs, the use of which renders resulting information as Web pages on the client computer. An information set rendered as a Web page on the client computer is referred to as a state, while the URL is a representation of the state. Changes in state brought about by changes in representation are called Representational State Transfer.

REST URLs comprise parameters based on which information is rendered. Changing the parameter values leads to changes in the rendered information. In other words, you can retrieve and view information by providing appropriate parameter values in the REST URL itself.

AppWorks Platform XForms provides the functionality to create a user interface for a REST URL. The interface is saved as a composite control and can be reused in XForms as required. In addition, you can use a message map to set or manipulate the REST URL parameters that you specify while creating the control.

To create a composite control using a REST URL:

1. Access the [Composite Control Modeler](#) () to create a composite control using a REST URL.
The Untitled Composite Control - Composite Control wizard opens.
2. Type the **Name** of the composite control.
3. Type a **Description** of the composite control.
4. Select the REST URL Control Type option to create URL-based composite control.
The **Control** Control Type option is used to create composite controls based on design-time and run-time libraries.
The title of the wizard changes to <composite control> - Composite Control. The REST URL tab appears where you can specify details regarding the URL to be used to create the composite control.
5. Enter the URL to be used to create the composite control in the URL field and tab out or click outside the field.
The parameters available in the URL for creating a composite control are displayed in the Configure URL Parameters table. The parameter details are displayed in the Name and Default Value columns.
6. To use a parameter in the composite control, select its corresponding check box in the Use in Map column.
The selected parameter is available for use at runtime. That is, it is possible to set its value at runtime.
7. Provide a **Description** for the parameter.
When you later use the message map, this description is used to represent the selected parameter. To make it easier to identify the parameter, ensure that you provide a meaningful description.
8. To customize the run-time behavior of the composite control, select the **Behavior** tab, and specify the run-time library in Runtime Library.
9. Click  (Edit) to create a run-time library and click Yes in the Confirm dialog box that is displayed.
The <library>.htm - Htm dialog box opens.
You can alternately click  (Lookup) to specify an existing run-time library from the Select Design Time Library dialog box that opens.
10. Make appropriate modifications in the <library>.htm - Htm dialog box.
11. Type the **Name** of the library.
12. Type a **Description** for the design-time library.

11. In Location, select  (Lookup) and specify the location at which the library must be saved in the Select Folder dialog box that displays. The <library>.htm - Htm dialog box displays the standard code structure using which you can build your library.
 12. Add the code to create the run-time control.
It is also possible to specify a design-time library for a REST-URL based composite control. In such cases, the design-time library will be used to define the design-time behavior of the composite control, overriding the REST-URL associated with the control.
7. Click .
- This creates a composite control that has the default representation based on the parameter you selected.

You can now drag the composite control from the Workspace Documents window to an XForm. Alternately, you can right-click the XForm, select Insert Composite Control, and drag the composite control from the Composite Controls dialog box that displays. This right-click menu option is available for a split area, an XForm, and for the Group, Groupbox, and Tab Page grouping controls. Once added to an XForm, you can [use a message map](#) to map data.

When the composite control is used in an XForm, an interface based on the base URL and the specified default parameters is rendered. You can use the message map to pass parameters, based on which the rendered Web page is updated.

Elements of a design-time type library

A composite control comprises a design time library and a run-time library. The design time library is the default library that defines the GUI representation of the control in the XForms Designer and the properties that can be set for the control. These properties control the actions that can be performed with the control at run time.

The run-time library is an optional library that defines the run-time behavior of the control. This library is required only if there is a run-time behavior associated with the control. The run-time options available for a control depend on the properties defined in the design time library of the control.

Basic structure of a design-time type library

A design time type library is similar to a run-time type library in terms of declaration.

Declaring the interface

The interface of a control is defined by making it public. To do so, you must use the `setPublic()` method (function call) as shown below.

```
setPublic(Hyperlink, "mycontrols.designtime");
```

The above code denotes that a library called Hyperlink is present in the mycontrols > designtime folder at <cordys-install-dir>\<your-instance>\web.

Declaring the essential type libraries

All type libraries must be declared in the design time library, including the basic interface or control from which the composite control is extended. The following sample code defines the type libraries essential for the Hyperlink control.

```
importType ("cas.xforms.designerlibrary.controls.XFormsControl"); importType  
("cas.xforms.designerlibrary.XFormsProperty");
```

As a default, every control must use `XFormsProperty`, which is used to manage the properties of a control.

Inheriting from a parent control

Every composite control must extend from a base control. By default, each control extends from `XFormsControl`, as shown.

```
inherit(Hyperlink, XFormsControl);
```

However, it is possible to extend a control from any control. In the following code sample, the `TextareaReadOnly` is extended from the `Textarea` control.

```
inherit(TextareaReadOnly, XFormsTextarea);
```

Define the static properties for a control

The Static properties are the properties necessary for a control. The following code sample defines static properties for the Hyperlink control.

```
Hyperlink.propertiesCaption = "Hyperlink";
```

propertiesCaption	Optional. Refers to the caption that appears on the property sheet of a control.
-------------------	--

In addition to these, you can add or define any other static properties that are required for your control.

Using the constructor

Each constructor of a composite control listens to its parent control and is initialized. The sample code for the Hyperlink control is as follows.

```
function Hyperlink(designerWindow) { this.XFormsControl(designerWindow); }
```

The `designerWindow` property that is used inside the constructor is available for every composite control, by default.

Implementable methods for composite controls

Every composite control follows the Interface > Implementation hierarchy, where some methods need to be implemented for the composite control. Some methods available for implementation are as follows.

Method	Description
getLabelElement()	Optional. It returns the Label (if any) of the composite control.
getRenderProperties()	Optional. It defines the properties that are used to render the HTML control. It is possible to write methods that are automatically called when such a property changes.
isGroup()	Optional. It denotes whether children can be added to the control. Returns true if the control is a Group control that can contain children controls.
isGroupBox()	Optional. It defines a control box (container) for grouping controls, such as Tab Group, Button container, and Image container. Returns true if a control box is defined.
isPrimitive()	Optional. It denotes whether children can be added to the control. Returns true if the control is simple or primitive. For example, the Input control.
isSingleton()	Optional. It denotes whether the control can be added only once to an XForm. Returns true if only a single control is allowed in the XForm.
isAutoExtentEnabled()	Optional. It denotes whether the control can be an autoextent enabled control. Returns true if the control is auto-extendable.
getChildContainer()	Optional. If the HTML template of the composite control is not simple (a simple template has a layout element and a container in it), then this method is used to specify which container is to be defined as the child container.
onCreateContextMenu(contextMenu)	Optional. It is used to create a context menu for the control.
removeChildren()	Optional. It is used to remove the children of a control, if present.
renderChildren()	Optional. It is used if the rendering mechanisms of child elements of various grouping controls are different. For example, the rendering mechanism of a Table control is different from that of a Groupbox control.
toHTML()	Optional. It is used to define an HTML template for the control, to represent it in design time. If not implemented, the default appearance defined in the framework is used.
toXFormMarkup()	Required. It creates and returns the run-time representation for a composite control.

The following sample code implements the `isSingleton()` method for the Hyperlink control such that only one Hyperlink control can be added to an XForm.

```
Hyperlink.prototype.isSingleton = function() { return true; }
```

Specifying render properties

Render properties denote the properties that need to be rendered on the HTML control. These properties are also defined on the run-time representation of the control. The following sample code demonstrates the implementation for the Hyperlink control.

```
Hyperlink.getRenderProperties = function()
{
  if (! this.renderProperties)
  {
    this.renderProperties = [
      new XFormsProperty("id", "ID", "@id"),
      new XFormsProperty("size", "Size", "@wcpforms:size"),
      new XFormsProperty("position", "Position", "@wcpforms:position"),
    ];
  }
  return this.renderProperties;
}
```

Render Property	Description
Id	Required. Refers to the ID of the run-time representation of the control.
size	Optional. Denotes the size of the control.
position	Optional. Denotes the position of the control.

You can also execute or set a specific functionality when this value is set. In this case, a method to set the value is automatically looked for and called.

The following sample code sets the position of the Hyperlink control.

```
Hyperlink.setPosition = function(value)
{
  this.htmlElement.position = value;
}
```

It is also possible to extend or inherit the render properties from a parent control and add more properties to it. The following sample code for an XGrid extends from a Table control and adds its own properties as well.

```
if (! this.renderProperties)
```

```
{
    this.renderProperties = new Array().concat(XFormsTable.getRenderProperties());
    this.renderProperties.push(new XFormsProperty("xpathBusinessObject", "Business Object", "@xpathBusinessObject"));
    this.renderProperties.push(new XFormsProperty("useCheckboxes", "Use Checkboxes", "@useCheckboxes"));
    this.renderProperties.push(new XFormsProperty("shadeClasses", "Shade Classes", "@shadeClasses"));
}
return this.renderProperties;
```

If render properties of the parent control are used as is, the `getRenderProperties()` method need not be written for the composite control.

Specifying the HTML template

An HTML template is defined for representing the composite control in design time. See [Templates for Composite Controls](#) for details.

Adding properties to control

You can specify properties for a control that are used during design time and can be accessed at run time.

You can use the following syntax to define properties for a control.

```
<script type="cordys/xml" id="_propertiesDefinition">
<properties>
<property name="myProperty" description="My Property" type="select"
oncontentchanged="propertyChanged" onbind="handleValueBind">
<default>2</default>
<values>option A:1,option B:2</values>
</property>
<property falsemode="both" type="task"/>
</properties>
</script>
```

When you define the properties node for a control, a property sheet is associated with it. The property sheet appears when you double-click the control at design time. The property sheet displays the properties that you define in the properties node.

The unique name in the above sample code, is the value of the `xfTemplate` attribute in the `toHTML()` method in the design-time library of the composite control. If `toHTML()` method is not used in the design-time library, the library name must be used instead. If library name is used, then the unique name must be defined in lower case only.

The `toHTML()` method is not available for [composite controls created from existing markup](#). The following attributes are available for all properties defined for a control.

- **name:** Denotes the name of the property. The value you enter in the property sheet is set for the control. At runtime, the property is available on the control as

<controlId>.<name>.

- **description:** Denotes the description of the property. The text value provided here, displays as a label for the property in the property sheet.
- **type:** Denotes the type of property. Following is the list of values supported for this attribute.

Property Type	Controls displayed	How it works	Accessed at run time as
Id	Input	Displays an input field with id as its label. When edited, the corresponding property of the control is updated. If a control with same id already exists in the XForm, a notification message is displayed.	<controlId>.id
Boolean	Check box	-	<controlId>.<name>
xml	Output with Zoom button	-	-

- Click **Zoom** to open the XML Editor application. If an XML is specified for the control, it is displayed in the XML Editor. If no request is set for the control, an empty XML is displayed.
- Modify the request and close the XML Editor to update the request on the control.
- The XML provided in the XML Editor must have Id as an attribute. The value of the Id attribute is displayed in the output field.

<controlId>.<name>

select	Select box	The values node is used to populate the control.	<controlId>.<name>
model	Select box	Displays the models available in the XForm.	<controlId>.model
task	Check box	Selecting the check box displays the Task Part Details pane. The falsenode node is used to display options in the pane.	-
webservice	Output with three buttons	-	-

- **Select a Method button:** Click to open up the Select Method dialog page that displays the methods available in the current project. You can select a method and click OK. The name of the selected method displays in the output field. The SOAP Request for the selected method is generated and added to the application. It can also be viewed in the XML Editor in the XForms Designer. The value of the id attribute of the generated XML is

set as <controlId>_request. This ID is set to the requestSchema property of the control, which can be used for accessing the request at run time.

- **Clear Method button:** Click to clear the property set on the control. It also deletes the request from the control and the XML Editor.
- **View/Edit Request XML button:** Click to open the XML Editor. If a request is set on the control, it displays in the XML Editor. If no request is set for the control, an empty XML displays. Modify the request and close the editor to update the request on the control. The SOAP Request can also be viewed in the XML Editor in the XForms Designer. The ID of the XML is set as <controlId>_request. This ID is set to the requestSchema property of the control, which is used for accessing the request at run time.

<controlId>.requestSchema			
region	Select	Displays the IDs of all regions available in the form.	<controlId>.region
reference	Input with zoom button	<p>Clicking the Zoom button opens the Field Chooser dialog box that displays the response schema of the model as a tree list. To reference a node, select it in the tree list and click OK. The name of the selected node displays in the Input field.</p> <p>Note: The Zoom function opens the Field Chooser dialog box only when a model is selected. However, you can also enter your own value in the Input field.</p>	<controlId>.xql
string	Input		<controlId>.<name>

- **falsemode:** Denotes the **Not Available Mode** options to be displayed in the property sheet. The Not Available Mode denotes a scenario where a task involving the control is not available to the user. It applies to task property type.
 - If **falsemode** is set as both, the hide and disable options display in the property sheet of the control, for the Not Available Mode. You can select whether the control must be hidden or disabled when it is not available as a Task.
 - If **falsemode** is set as hide or disable, no options display in the property sheet, for the Not Available Mode. The control is by default hidden or displayed when not available as a Task.
- **oncontentchanged:** Denotes the method that must be called when a property value is edited. The method, when called, is supplied the following parameters.
 - **htmlElement:** The control for which the property is edited.
 - **value:** The edited value of the property.

- **parentWindow:** The window object of the property sheet. The following is a sample property definition with the `oncontentchanged` attribute.

```
<property description="Request" name="request"
  oncontentchanged="requestChanged" type="webservice"/>
```

The following is a sample code for the method.

```
/// When you select a new Web service [method request] in the property sheet, the
requestChanged handler is invoked.
// htmlElement- refers to the control that displays in the designer and for which
the property is edited
// value-refers to the new request that you select
myControl.prototype.requestChanged = function(htmlElement, value, parentWindow)
{
  // Get the runtime object.
  var control = htmlElement.control;
  // Call a prototype method 'setDefaultValuesOnRequest' on myControl and pass the
value.
  control.setDefaultValuesOnRequest(value);
}
```

- **Onbind:** Denotes the method to be called before a value is bound to the control. The method, when called, is supplied the following parameters.
 - **htmlElement:** The control for which the property is edited.
 - **value:** The edited value of the property.
 - **parentWindow:** The window object of the property sheet. This attribute is available for Input and Select controls only. The following is a sample property definition with the `onbind` attribute.

```
<property name="target" description="Target Frame" type="select"
onbind="hideShowFrameProperties">
<values>Frame:frame,Region:region</values>
<property>
```

The following is a sample code for the method.

```
// When a value is bind to the 'target' property in the property sheet, the
hideShowFrameProperties handler is invoked.
// htmlElement- refers to the control that displays in the designer and for which
the property is edited
// value-refers to the new request that you select
myControl.prototype.hideShowFrameProperties = function(htmlElement, value,
```

```

parentWindow)
{
// Hide or show the Frame properties or Region properties in the property sheet.
var propertyWindowDocument = parentWindow.document;
if(value == "frame")
{
propertyWindowDocument.all["frameProperties"].show();
propertyWindowDocument.all["regionProperties"].hide();
}
else
{
propertyWindowDocument.all["frameProperties"].hide();
propertyWindowDocument.all["regionProperties"].show();
}
// Get the runtime object.
Var control = htmlElement.control;
// Call a prototype method 'targetChanged' on myControl and pass the value.
Control.targetChanged(value);
}

```

The following nodes can be defined for a property to add additional features:

- **default:** Denotes the child node that is used to set the default value for a property of the control. The text value of the node displays as the default property in the property sheet. At run time, this value is available for the property of the control. The following sample code demonstrates the use of this attribute for a property of type xml.

```

<default>&lt;xml
id="sampleSchema">&lt;TreeSchema&gt;&lt;searchP
ath&gt;items/&lt;/searchPath&gt;&lt;Root&gt;&lt;description&
gt;Items&lt;/description&gt;&lt;Root&gt;&lt;/TreeSchema&
gt;&lt;/xml&gt;</default>

```

- **values:** Denotes the child node that is used to populate a property of type select. You can specify a list of comma separated description-value pairs to populate the Select control. The following sample code demonstrates the syntax to be used for this attribute.

```

<values>Description A:Value A,Description B:Value B</values>
<property name="chartType" description="Chart Type" type="select">
<values>Three Dimensional Column:Column 3D,Two Dimensional Line:Line 2D</values>
</property>

```

Adding Intellisense options for the control

For each composite control that you create, you can add intellisense options that must display in the Script Editor. To do so, you must specify the API (method or event) in the method and event definitions of the composite control.

Adding methods to the Intellisense

- To add methods to the intellisense, you must add them to the methods definitions in the following format:

```
< script type="cordys/xml" id="_methodsDefinition"> <methods> <method name="setMapControl"> <parameter name="sControlType" /> <parameter name="sControlType" /> </method> </methods> </script>
```

Where the `<methods>` node is used to list the methods supported by the control at run time.

The unique name in the above sample code, is the value of the `xfTemplate` attribute in the `toHTML()` method in the design-time library of the composite control. If `toHTML()` method is not used in the design-time library, the library name must be used instead. If library name is used, then the unique name must be defined in lower case only.

The `toHTML()` method is not available for composite controls created from existing markup. For each method, you can use the `<method>` node to define the method:

- name:** Attribute that defines the name of the method.
- parameter:** Defines the name of the parameter. It is possible to add zero or more parameters for each method or parameter. You can add prefixes to parameter names, such as `s` for string, `I` for integer, and `b` for Boolean. The parameter names defined here will display in the Intellisense at design time.

Adding event definitions

- To add events to the intellisense, you must add the events to the events definition of the composite control. The events can be added as follows.

```
< script type="cordys/xml" id="_eventsDefinition">
<events>
<method name="setMapControl">
<parameter name="sControlType" />
<parameter name="sControlType" />
</method>
</events>
</script>
```

Where list of events supported by the control needs to be mentioned in `<events>` tag.

The unique name in the above sample code is the value of the `xfTemplate` attribute in the `toHTML()` method in the design-time library of the composite control. If `toHTML()` method is not used in the design-time library, the library name must be used instead. If library name is used, then the unique name must be defined in lower case only.

The `toHTML()` method is not available for composite controls created from existing markup. For each event, you can use the `<event>` node to define the event:

- **name:** Attribute that defines the name of the event.
- **property:** Defines the name of the property for the `eventObject` (of the event). It is possible to add zero or more properties for each event. The properties defined here will display in the Intellisense at design time.

Adding schema definitions

You can define the schema of the composite control, which will be used in the Message Map. The schema can be defined as follows.

```
<script type="cordys/xml" id="_schemaDefinition">
<data>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.example.org/MapSchema"
  xmlns:tns="http://www.example.org/MapSchema"
  elementFormDefault="qualified">
  <xsd:element name="BusinessObject">
    <xsd:complexType>
      <xsd:all>
        <xsd:element name="UniqueId" type="string"></xsd:element>
        <xsd:element name="MarkerAttribute" type="string"></xsd:element>
        <xsd:element name="Location">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="Country" type="string"></xsd:element>
              <xsd:element name="AdministrativeArea" type="string"></xsd:element>
              <xsd:element name="SubAdministrativeArea" type="string"></xsd:element>
              <xsd:element name="Locality" type="string"></xsd:element>
              <xsd:element name="Thoroughfare" type="string"></xsd:element>
              <xsd:element name="PostalCode" type="string"></xsd:element>
            </xsd:all>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="InfoWindow">
          <xsd:complexType>
            <xsd:all>
              <xsd:element name="Title" type="string"></xsd:element>
              <xsd:element name="Content" extendable="true">
                <xsd:complexType>
                  <xsd:sequence>
                    </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:all>
            </xsd:complexType>
          </xsd:element>
        </xsd:schema>
      </data>
    </script>
```

Where the unique name is the value of the `xfTemplate` attribute in the `toHTML()` method in the design-time library of the composite control. If `toHTML()` method is not used in the design-time library, the library name must be used instead. If library name is used, then the unique name must be defined in lower case only.

The `toHTML()` method is not available for [composite controls created from existing markup](#).

Providing the run-time representation (XML properties)

A run-time representation of a composite control is required if a run-time library is available for the control. For more information, see [Templates for Composite Controls](#).

Context menus

Each control can have a defined context menu (comprising a set of menu items), which displays when the XForm shows up. The following is a sample context menu definition for the Hyperlink control.

```
Hyperlink.prototype.onCreateContextMenu = function(contextMenu)
{
    contextMenu.addItem("CMenuremoveObjects", "removeSelectedObjects()",
"/cordys/wcp/theme/default/icon/action/delete.png", "Delete", null, null, "Del");
    contextMenu.addSeparator("CSep1");
    contextMenu.addItem("CMenuproperties", "showProperties()", null,
"<b>Properties</b>");
}
```

This code creates an empty context menu and adds items and separators to it. Here, the methods are defined on the XForms Designer. If a context menu's method must be present in a composite control, then the method name must be suffixed with `activeControl()`. This ensures that the method calls the necessary function in the control library.

```
contextMenu.addItem("CMenuproperties", "activeControl().showMyProperties()", null,
"<b>Properties</b>");
```

You can also choose to use the base class context menu and remove the unrequired items from there.

```
XGrid.prototype.onCreateContextMenu = function(contextMenu)
{
    this.inherited(XGrid,"onCreateContextMenu");
    contextMenu.removeItem("CMenulayout");
    contextMenu.removeItem("CMenugridType");
}
```

Accessing parts of a composite control

Various methods are available for composite controls that enable you to access different parts of a control and the designer that hosts the control. The following methods are available in prototype functions of the control.

Method	Description
getControlElement()	Returns the HTML element.
this.defaultView	Returns the window object of the design time composite control.
this.designerView	Returns a handle to the XForms Designer window that hosts this context menu. Using this handle, the following methods can be accessed from XForms Designer.
this.htmlElement	Returns the HTML template defined on the control.
this.getLabel()	Returns the Label of the control, if defined.
this.getEventsDefinition()	Returns the events XML of the composite control.
this.getMethodsDefinition()	Returns the methods XML of the composite control.
this.getPropertiesCaption()	Returns the properties caption of the composite control.
this.getPropertiesDefinition()	Returns the properties XML of the composite control.
this.getSchemaDefinition()	Returns the schema definition of the composite control, if present. Like properties, methods and events, the schema can also be defined with _schemaDefinition as ID.
this.parentControl	Returns the parent of the composite control.
this.setLabel()	Sets a Label for the composite control.

Controlling XForms designer from the composite control

The method `this.designerWindow()` on the composite control prototype returns a handle to the XForms Designer window which hosts this context menu. Using the `this.designerWindow()` method, the following methods can be accessed from XForms Designer.

Method	Description
activeControl()	Returns the handle to the current composite control.
addJavascriptToForm(JavaScriptFile, uniqueId)	Adds the corresponding JavaScript file to the XForm definition. The JavaScript files will be loaded in run time.
addJavascriptToForm(sUrl, sKey)	Used to add a JavaScript (JS) file to the XForm. If key is passed, it is easy to locate the file and remove it if needed.

Method	Description
addScriptToForm(script, functionName, uniqueId)	<p>Adds the specified script to the specified function name, in the XForm Definition. Syntax for use is as given below.</p> <pre data-bbox="752 413 1388 477">addScriptToForm(\$ID\$_openApplication.toString() (), "openApplication", controlId);</pre>
	<p>As an example, let's consider that the above sample code adds the following function to the XForm Definition:</p>
	<pre data-bbox="752 677 1323 794">function \$ID\$_openApplication() { var newApplication = treeItem.data.cloneNode (true); application.loadAppPalette (newApplication); }</pre>
addStyles(arrStyles, bApply, arrKeys)	<p>In this case, <code>\$ID\$</code> will be automatically replaced by the specified <code>controlID</code>. And, the <code>toString()</code> method will convert the function to string.</p>
addStyleSheetToForm(url)	<p>Adds a collection of styles to the XForm definition, saves and remembers them by the keys passed, and applies them when <code>bApply</code> is true.</p>
getPropertiesNode(sNode, ns)	<p>Adds a style sheet (defined using URL) to the XForm definition. The URL of the style sheet is loaded in run time.</p>
getUniqueID(sName)	<p>Creates the XML node <code>sNode</code> and returns it. <code>ns</code> denotes the namespace and is optional.</p>
setChanged(bChanged)	<p>Takes a name and returns unique ID for it. For example, if we pass <code>menuitem</code>, it returns <code>menuitem1</code> if not available for use.</p>
showProperties(oProperties, bUnconditional)	<p>Displays an alert in the XForms designer and prompts the user to save. This method can be called when you need to intimate the designer that something has changed.</p>
removeJavascriptFromForm(sKey)	<p>Used to show property sheet for any object. When passing unconditionally, it closes and opens the existing property sheet.</p>
removeJavascriptFromForm(uniqueId)	<p>Removes the added JavaScript file recognizing the unique key.</p>
	<p>Removes the JavaScript file from the XForm Definition based on the <code>uniqueId</code>.</p>

Method	Description
removeStyle(sKey)	Removes the style specified by the key.
removeStyleSheetFromForm(url)	Removes a style sheet (defined using URL) from the XForm definition.

Controlling the design-time lifecycle

Design-time behavior of a control refers to its behavior in XForms Designer that is its behavior from the time the control is dragged into an XForm to the time it is rendered, including the creation and saving of the control. It is possible to define the behavior of the control in design time.

The following methods can be used to define the design-time behavior of a control.

Method	Description
onAfterCreate()	Called after a new control is created (dragged) in the XForm. It can be used to set default properties.
onAfterDeselect()	Called after a control is deselected in the XForms Designer.
onAfterInsertInParent()	Called after a control is added to a parent control.
onAfterRemove()	Called after a control is removed. It can be used to clean up.
onAfterRemoveChild()	Called after a child control is removed from its parent control.
onAfterRender()	Called after a control is rendered, read from <code>.caf</code> definition and displayed as HTML (save, close and re-open). It can be used to set some extra properties.
onAfterSelect()	Called after a control is selected (highlighted) in the XForms Designer.
onBeforeCopy()	Called before the control is copied.
onBeforeCreate()	Called after the control type is created. If returned false, the control will not be created in the XForm.
onBeforeCut()	Called before the cut action is performed on the control.
onBeforePaste()	Called before the control is pasted.
onBeforeRemove()	Called before a control is removed.
onBeforeSave()	Called before a control is stored into the <code>.caf</code> definition (that is before the XForm is saved). The default behavior is to remove the empty action nodes from the properties, but it can be implemented to do specific tasks while saving the control.
onInitialize()	Called after the design-time HTML element and properties are created.

Elements of a run-time type library

All Composite Controls have a design-time representation (mandatory) that defines how the control is in design time. The run-time library complements the functionality built into the design-time library as it imparts a run-time behavior to the control. It is possible to avoid a run-time library by including the necessary run-time representation in the design-time library. However, in some cases, additional behavior is needed in the controls, which can be achieved only by a run-time library.

This topic describes the content that must be defined for a run-time library while creating a control.

Attaching a run-time library

To associate a run-time library with a composite control, a mechanism is needed to load the library at the run time. This is done by defining a run-time representation for the control in the `toXFormMarkup()` method.

The following sample code associates a control with a run-time library.

```
// this associates the control with a run-time behavior
this.m_xmlProperties.setAttribute("namespace",
"mycontrols.runtimelibrary.Hyperlink");
```

In the example, the complete namespace of the run-time library is specified. The XForms run time understands the syntax and loads the appropriate type library, making it available in run time.

Types of run time

You can implement the run-time type library for a composite control in two ways.

- **Direct:** A run-time type library can be directly loaded and attached to the composite control's run-time representation. The examples considered in the topics related to composite controls, such as Hello World, Hyperlink controls are such cases.
- **Adapter:** In this approach, a type library plugs-in the actual behavior into the composite control run-time representation. The type can be in any form such as JavaScript and HTML. The XForms run time recognizes only the adapter library, while the adapter takes care of communicating with both the XForms run time and the Run-time Library. This approach is especially useful in situations where a third-party library is used, which cannot be modified because of copyright issues.

Creating run-time libraries

Run-time type libraries have a standard interface.

Associating the run-time behavior

To make a run-time library function on an XForm at run time:

1. Load the run-time library. This is done by the XForms run time that recognizes the namespace of the run-time library.
2. When loaded, the libraries have various means of initializing itself and process data. The various possibilities of initialization are as follows.

Attach To: Some run-time libraries do not have a run-time GUI and only impart specific behavior to controls they are attached to. These cannot function independently and must be associated (attached) to certain controls. Once attached, the composite controls initialize on their own by attaching the necessary behavior to other controls.

For example, the Hyperlink control is invisible at run time and does not exist by itself. It needs to be attached to some control. The Attach To context-menu option must be enabled for such controls. The following sample code for theHyperlinkcontrol shows how the attached controls are initialized:

```
Hyperlink.attachType = function(hyperlink)
{
    //make the control's background color orange here
    if ( hyperlink.registeredControls )
    {
        var ctrls = hyperlink.registeredControls.split(";");
        for (var i = 0; i < ctrls.length; i++)
        {
            var html = hyperlink.ownerDocument.getElementById(ctrls[i]);
            html.style.backgroundColor = "orange";
        }
    }
}
```

In the example,

- All initializations are done on the `attachType()` method of the run-time library.
- Each control with the Attach To behavior will have an attribute `registeredControls`, which is a list of (semi-colon separated) control IDs.
- The Hyperlink control will iterate through each control and initialize for each of them.

The above code makes the background color of all attached controls turn orange.

Data model

Some controls (such as Tree and XGrid) require data to operate. For such controls, a mandatory property `Model` needs to be defined, which denotes the data model that supplies data to the control. The following sample code sets the data model for a Tree control.

```

Tree.attachType = function(tree)
{
.....
.....
.....
if (tree.model)
{
  if(tree.treeData) tree.treeData = null;
  var dataModel = tree.ownerDocument.parentWindow[tree.model];
  if (dataModel) dataModel.addListener("xforms-ondatacompleted", bindTreeData(tree,
dataModel));
}
}

function bindTreeData(tree, dataModel)
{
  return function(eventObject)
  {
    tree.setTreeData(dataModel.getData());
  }
}

```

In the above sample code:

- The Tree control looks for a property called model on the control. This property is set during design time from the property sheet.
- The control listens to the xforms-ondata completed event of the data model.
- On the event handler, it picks up the data and uses it for rendering purposes. In this case, it sets the data to the Tree control.

There are innumerable possibilities of making a run-time library apart from the two most practiced approaches given here. You can set the necessary property in the run-time representation, collect it in the run-time library, and do the necessary initializations.

Accessing application object

- To access the window application in which the control is loaded, use `control.ownerDocument.defaultView.application`.

Templates for composite controls

Whether it is creating, using, or extending a composite control, templates are needed to be defined by the developer writing the composite control. Templates are needed for both design time and run time.

This section explores how the design time and run time can be represented and what information is vital for the control's existence.

Design-time representation

The design-time template of the control is defined on the `toHTML()` function which is written for every composite control. The following is the skeleton for such a definition.

```
<composite-control-type-name>.prototype.toHTML= function()
{
    return <template-as-string>;
}
```

It is necessary that this function return the HTML as string, so the framework can load it properly and return it as HTML. The HTML element, once created here, can be accessed from any part of the composite control as follows:

```
var htmlElement = this.htmlElement();
```

A sample definition for a simple HelloWorld control is as follows:

```
HelloWorld.prototype.toHTML= function()
{
    return "<div id='helloworldTemplate' xfTemplate='xhelloworld' movable='true'
resizable='true' \
layout='vertical' class='v_layout' defaultCursor='default' defaultWidth='300' \
style='height:30px;overflow:visible' > \
<img id='moveHandle' style='left:2' ondragstart='return false' \
src='/cordys/cas/xforms/images/designtime/move.gif' /> \
<div class='simplecontainer extensionbox' \
movable='false' resizable='false'> \
Hello World ! \
</div> \
</div>";
}
```

The following table describes every attribute, property, and node in the HTML definition above:

Attribute	Description
<code>Id</code>	Required. Denotes the ID of the design time control template.
<code>xfTemplate</code>	Required. Denotes the XForm template name based on which the control's properties are defined. This is usually "x" followed by the name of the control.
<code>movable</code>	Required. Defines whether this control can be moved in design time or not. Values can be true or false.
<code>resizable</code>	Required. Defines whether this control can be resized in design time or not. Values can be true or false.
<code>visible</code>	Optional. Denotes whether the control is visible at run time or not. Values

Attribute	Description
	can be true or false.
noLayout	Optional. Defines whether the control is constraint to a specific layout on the XForm. If this is true, then the control will not have any anchoring feature, and layout changes on the XForm or on the parent control will not influence this control. In most cases, for controls that are invisible at run time, this property is true. However, this can also be set to true for controls like status bar, so that you can decide and position the control always on the bottom.
layout	Optional. Defines the default layout of the control on the form. It can be vertical, horizontal, or free. Though this is set, the layout preference will be followed based on the parent control or XForm.
class	Optional. A class can be set for composite controls in design time. In most cases it has the layout class (for example v_layout), but it can be of any value.
defaultCursor	Required. Denotes what is the cursor when focused on the control. This can be default, move, or anything definable by the style cursor attribute.
defaultWidth	Optional. Denotes the width of the control.
style	Optional. Any style settings for the control can be defined here. Height is one such attribute which can only be defined here.
img	Optional. An image tag is usually added to every composite control that is movable. This is the image handle with which the control can be moved.
div	Optional. The DIV inside the composite control is called the child container of a composite control and can hold any data inside it. In case of HelloWorld, it carries just the text inside it. In case of the WebNavigator control, it can contain the Tree library.

For more variations of design time HTML representations for a composite control, look into the default composite controls at the following location:

```
<cordys-install-dir>\<your-instance-dir>\web\cas\xforms\designerlibrary\controls
```

and

```
<cordys-install-dir>\<your-instance-dir>\web\cas\xforms\designerlibrary\controls\extensions
```

Run-time representation

The run-time template of every composite control represents how the control will look in run time. This is defined inside the toXFormMarkup() method. The following is the skeleton code for such a definition.

```
<composite-control-type-name>.prototype.toXFormMarkup = function()
{
  if (! this.m_xmlProperties)
  {
    this.m_xmlProperties = this.designerWindow().getPropertiesNode(<tag-name>);

    // mandatory properties
    this.m_xmlProperties.setAttribute("xformextendedname", <composite-control-name>);
    this.m_xmlProperties.setAttribute("id", <id>);
    this.m_xmlProperties.setAttribute("namespace", <library-name>);

    //other properties here
  }
  return this.m_xmlProperties;
}
```

The above code creates the XML properties once, and returns it every time if already created, thereby being performance centric. Once created, the run-time representation can be accessed anywhere inside the composite control as follows:

```
var properties = this.toXFormMarkup();
```

A sample definition for a simple HelloWorld control is as follows.

```
HelloWorld.prototype.toXFormMarkup = function()
{
  if (!this.xformMarkup)
  {
    this.xformMarkup = this.designerView.getPropertiesNode("div");
    this.xformMarkup.setAttribute("xformextendedname", "helloworld");
    this.xformMarkup.setAttribute("id", "");
    this.xformMarkup.text = "Hello World !";

    // custom properties
    this.xformMarkup.setAttribute("caption", this.xformMarkup.text);
    this.xformMarkup.setAttribute("namespace", "mycontrols.runtimelibrary.HelloWorld");
  }
  return this.xformMarkup;
}
```

The following table describes the attributes used in the run-time representation above.

Attribute	Description
xformextendedname	Required. Denotes that this is a composite control. It is the name of the composite control without spaces in between.
id	Required. Denotes the ID of the control in run time, using which the control is accessed.

Attribute	Description
namespace	Required. This denotes the namespace of the run-time library if there is one. Usually every design-time library will have a run-time library, but it is not mandatory. The namespace is synonymous to the java packaging structure that is folder-name.TypeName. For example, for the Statusbar control, it iswcp.library.ui.StatusBar.

All other additional properties set on top of these attributes denote implementation specific to each control.

Integrating an XForm with WS-AppServer

Before you begin:

- To develop applications in AppWorks Platform XForms using the Web services available in the AppWorks Platform WS-AppServer, ensure that the corresponding Web service operations are available in the LDAP.

To integrate an XForm with WS-AppServer:

- Open the **XForm** in the XForms Designer.
- From the Insert tab, select the required Web service operation and drag it to the XForms Designer.
Alternately, right-click the Designer Area, and select **Insert Web Service** to view available Web services and add the required one to the XForm.
The Generate Input and Output UI dialog box opens.
- Select a service from **Services**.
- Select a port from **Ports**.
- Select the action to perform from **Operations**.
- To generate the input UI, select **Generate UI for Input message**.
- To generate the output UI, select **Generate UI for Output message**.
The UI is displayed in the Designer Area.
- Click **OK**.
- From the toolbar, select **Advanced** to view advanced options.
- From the toolbar, click  (Manage Models).
The Manage Models window opens.
- Select the model and click  (Edit).
Alternately, click the model name.
The Model Properties dialog box opens.
- Select **WS-AppServer Integration**.
The WS-AppServer tab opens.
- To enable, disable, hide, or display controls in an XForm based on the business logic available in the WS-AppServer, select **Apply Access Control**.

14. To provide default values for controls in the XForm when a new record is added to it, select **Initialization Required**.
15. To enable server-side validation of data that is displayed in the XForm, select **Constraint Validation**.
16. To specify the Web service operation to be executed before the validate request is sent to the WS-AppServer, select an option from **Before Validation**.
17. To specify the Web service operation to be executed after the validate request is sent to the WS-AppServer, select an option from **After Validation**.
18. Click **OK**.
19. Double click a control on the XForm to define its integration properties.
The <control> window opens.
You can integrate only the Input, Password, Textarea, Check, and Select controls.
20. From the <control> window, select the **Model**.
The WS-AppServer Properties pane opens.
21. Expand the WS-AppServer Properties pane.
22. To enable validation of changes made to specific controls on the XForm, select **Constraint Validation on Change**.
23. To specify how to prefill the Select control with content using the application logic, select **Never**, **Always**, or **Once** from **Initialize Value Sets**.
 - The Initialize Value Sets option is available only for the Select control. For a Select control, the request to retrieve data is sent only when the control is highlighted and only if the data node exists.
 - The `qValues` attribute is used to initialize value sets in WS-AppServer for Select controls.
24. Click .

The changes made to the XForm are saved and the XForm is integrated with the WS-AppServer.

Using the undo and redo features

An undo action reverses the last action that you perform in the XForms Designer, while the redo action undoes the last undo action.

You can use the Undo and Redo features for actions performed in the Designer Area and the Script Editor. It is possible to undo changes in the reverse consecutive order in which they are performed until the first action is undone.

You can use the Undo and Redo features for the following:

- Adding, anchoring, and positioning controls
- Conversion of controls using the Change To feature
- Changes involving grid types and toggle check boxes in Table controls

- Changes involving hiding and displaying headers in Groupbox controls
- Aligning, moving, resizing, and deleting controls, except for moving fields in Table and Tab Page controls
- Changes involving header positions in Tab Page controls
- Typing text in Code Snippet and Rich Text Editor controls
- Resizing and splitting of controls
- Resizing of splitting bar and changes involving splitter types
- Changes involving layouts changes
- Positioning of control labels
- Insertion of methods with default layouts

Using the Undo feature to remove a Web service operation added to the XForm, removes the UI and the model that is generated for the Web service operation.

- Cut, copy, and paste actions in XForms Designer and the Script Editor
- Typing, indenting, and deleting text in the Script Editor

See [Keyboard Shortcuts](#) to view the shortcut options available for AppWorks Platform XForms.

Interceptor support

You can manipulate the generated HTML content from the XForms Engine using the XForm Interceptor mechanism.

XForm interceptor interface

The XForm Interceptor is an interface

`com.eibus.applicationconnector.xforms.Interceptor`, which when implemented can intercept the generated HTML before it goes to the Client (Browser).

The interface `com.eibus.applicationconnector.xforms.Interceptor` contains the following method:

```
public void onGeneratedHTML(int content);
```

Important: Provide the interceptor implementation class name (FQCN) in the XForms tab of the XForms Service Container Properties.

Sample interceptor implementation

The following is the sample implementation of the XForm Interceptor:

```
package com.cordys.test.dynamic;
```

```
import com.eibus.applicationconnector.xforms.Interceptor;
import com.eibus.xml.nom.Node;
public class MyInterceptor implements Interceptor
{
    public void onGeneratedHTML(int content)
    {
        Node.setAttribute(Node.getFirstElement(content), "version", "myversion");
    }
}
```

Managing XForms

In the Workspace Documents (Explorer) window, various options are available to manage the XForms in a project. You can [edit](#), [delete](#), or [rename](#) an XForm. You can also [view and update the XForm's properties](#) such as Name and Description.

Note: For XForms created using Cordys 4.2 C1 or later, internal version support is available. You can continue to use your existing XForms in the current AppWorks Platform version

Reusing XForms

You can use an existing XForm as a template to quickly design other XForms. You can then modify these XForms to specialize them as required.

To reuse XForms:

1. [Open the XForm](#) in the XForms Designer.
2. Drag the XForm that you want to use as the template to the open XForm in the Designer Area.
A message warns that the operation cannot be reverted.
3. Click **OK** to proceed.
The template appears in the XForm and is enclosed by a border that is invisible at run time.

The XForm is added as a template.

When added as a template, the script available in the XForm is also copied with it. Ensure that the names of event handlers for both the XForms are different. If not, you must manually link events to the correct event handlers.

You can add additional controls to the XForm, if required.

To remove the template:

1. Delete its interface and manually remove style sheets and scripts from the property sheet of the XForm.
2. Additionally, you must remove XML definitions and script from the XML Editor and Script Editor, respectively.

Editing or deleting XForms

The Workspace Documents window comprises artifacts, such as XForms and business models, that are associated with various projects. It is possible to edit these artifacts.

To edit or delete XForms:

1. Access the XForm and double-click it. Alternatively, right-click the XForm and select **Open**.
The XForm is displayed in the XForms Designer.
2. Make appropriate modifications to the XForm, and click  (Save) on the toolbar to save the changes.

The changes made to the XForm are saved.

To delete an XForm in the Workspace Documents (Explorer) window:

- Right-click it and select **Delete**.
- Alternatively, to delete an XForm in the Workspace Documents (My Recent Documents) window, select the XForm, click  (Extend Menu) and select Delete.

After you complete this task:

- After creating an XForm, you can translate it into various languages, as required.
- You can also publish it and add it to a Business Process Model for use in an application.

Renaming XForms

You can rename an XForm in the Workspace Documents window.

Before you begin:

- Access the XForm you intend to rename.

To rename an XForm:

1. In the Workspace Documents (Explorer) window, right-click the XForm you want to rename and select **Rename**.
Alternatively, select the XForm and press F2 on the keyboard.
The name of the XForm is displayed in the editable mode.
2. Type the new name of the XForm, and press Enter or click outside the editable area.
3. Alternatively, in the Workspace Documents (My Recent Documents) window, select the XForm, click  (Extend Menu) and select Rename. Type the new name of the XForm in the Rename artifact dialog box that opens and click **OK**.

The XForm is renamed.

Renaming an XForm modifies its description or code, depending on what you specify.

Updating the properties of an XForm

You can modify the properties such as name and description of an existing XForm.

To update the properties of an XForm:

1. Access the XForm you intend to update.
2. In the Workspace Documents (Explorer) window, right-click the XForm and select Properties.
3. Alternately, in the Workspace Documents (My Recent Documents) window, select the XForm, click  (Extend Menu) and select Actions > Properties.
The <XForm name> - User Interface dialog box opens.
The General, Annotations, and History tabs display the properties of the XForm. The History tab displays read-only information regarding the user and the dates on which the XForm was created and last modified.
3. From the General tab, provide the **Name**.
4. Provide the **Description**.
5. For Location, click  (Lookup) to browse to and specify the folder in which the XForm must be placed.
6. Click **here** to view the URL using which you can access the XForm directly, without launching the AppWorks Platform.
7. Copy the URL that displays in the Clipboard dialog box.
4. In the **Annotations** tab, provide applicable comments.
5. Click **OK**.

The changes that you made are saved when you close the <XForm> - User Interface dialog box.

Customizing interface styles

AppWorks Platform uses [Cascading Style Sheets](#) to define the appearance of interfaces in the application. Cascading Style Sheets (CSS) are a set of specifications that define how the application will appear when opened. As all information regarding the application's look and feel is captured in a .css file, it is easy to update or apply new styles to the application.

When you install AppWorks Platform, a collection of styles is provided with the application, by default. You can customize the available styles or create new styles, if required. The topics in this section include a [guideline](#) for the customization of a CSS. For information about creating a CSS and applying it to XForms, see [Customizing and Applying Cascading Style Sheets](#).

Customizing and applying cascading style sheets

Creating custom cascading style sheets

You can use `default.css` to create custom style sheets. The `default.css` is located at `<cordys-instance-folder>/wcp/theme/default/style` in your `<AppWorks Platform_installdir>`.

To do so, save the existing `default.css` under a new name and modify it as required. Ensure that you retain the order of selectors as mentioned in the original cascading style sheet's structure.

Applying cascading style sheets to XForms

To apply a style sheet to an XForm or to a control in an XForm, it must first be uploaded to the project that contains the XForm. For information about how to upload a document to a project, see [Uploading a Document to the Project](#).

You can modify the properties of an XForm to define the stylesheet that must be used with it. For detailed procedure, see [Setting Properties of an XForm](#).

For recommendations on customizing style sheets, see [Guidelines for Customizing Style Sheets](#).

Guidelines for customizing style sheets

AppWorks Platform does not provide an interface to customize cascading style sheets. To edit them, you need to modify the classes of cascading style sheets.

Guidelines for customizing the AppWorks Platform interface

1. To customize the AppWorks Platform interface, you need to edit `default.css`.
2. Ensure that you use the following guidelines while customizing the interface:
 - Retain all class names in the order in which they appear. If required, modify only the class properties.
 - AppWorks Platform recommends that you do not modify `xdefault.css`. If required to do so, test the modifications in the customized and non-customized `default.css` style sheet to ensure that the modified `xdefault.css` works with other style sheets. The changes made to `default.css` are applicable to the entire AppWorks Platform suite.

Guidelines for customizing the XForms interface

1. You can use custom style sheets to modify the appearance of an XForm interface. To do so, specify the URL of the style sheet in the Form - Properties window.
2. Ensure that you use the following guidelines while customizing the interface:
 - Use `default.css` as a template, and retain all class names in the order in which they appear. If required, modify only the class properties.
 - It is not mandatory to carry forward the positional properties of tabs from `default.css`.

Guidelines for customizing the XForms elements

1. You can use custom style sheets with custom class names to modify elements in an XForm.

2. Ensure that you use the following guidelines while using custom class names:
 - Use `default.css` as a template.
 - You need not carry forward the positional properties of tabs from `default.css`.
 - Add the custom class name to the existing selectors.

To modify the appearance of primitive controls such as Input, Password, Output, and Textarea:

- The following custom selectors are needed:
 - `.<custom>.input`
 - `.<custom>.textarea`
 - `.<custom>.output`
 - `.<custom>.h_label`
 - `.<custom>.v_label`
 - `.<custom>.lookup`
 - `.<custom>.lookup.disabled, .<custom>.lookup.output`

To modify the appearance of grouping controls such as the Groupbox control:

- The following custom selectors are needed:
 - `.<custom>.groupheader`
 - `.<custom>.groupcontent`

To modify the appearance of the Button and Frame controls:

1. The following custom selectors are needed:

- `.<custom>.h_button`
- `.<custom>.v_button`
- `iframe.<custom>`

In the above class names, `<custom>` stands for the class that will be specified in the property sheet of the specific control.

Notice the use of descendant selectors for classes of all controls, except the Textarea and Frame controls, which use the tag selector. A similar approach must be followed for other controls such as Check, Select, List, Tab Page, and Table. Also, notice the use of selector without spaces in case of the Buttons and Frame controls.

2. Create the XForm and specify custom class names for the elements whose appearance needs to be customized.
3. Add the custom style sheets to the XForm. The appearance of controls associated with custom classes should change.

Publishing XForms to runtime

Publishing an XForm ensures that you can use it as an application at runtime.

To publish XForms to run time:

1. In the Workspace Documents window, right-click the XForm to be published, and select **Publish to Organization**.
2. Ensure that any changes made to an XForm are saved before publishing it. Unsaved changes in an XForm will not be reflected in the published XForm. The published XForm is saved as a `.caf` file in the XML store at Cordys > WCP > XForms > runtime. While publishing, the design-time folder structure under the project is preserved. However, if you set the qualified name, then it is used to define the folder structure of the saved XForm.
The XForm is published to run time.
3. You can specify the URL of the published XForm to use it.
4. If you republish an XForm after modifications, you must refresh the browser to view the changes. Refreshing the browser removes the cache containing the older published version of the XForm.

Managing translation

AppWorks Platform supports the translation of [artifacts](#) into various languages.

To manage translation:

1. A Translator can right-click a [Project](#), select Translation > Translate to open the Translation Editor and view the list of labels in the Project for translation.
2. It is also possible to translate validation messages and Unified Feedback Objects in an XForm.

The topics in this section provide information about the use of the Translation feature:

- [Working with JavaScript Message Bundles](#)
- [Translating XForms](#)
- [Translating Validation and UFO Messages](#)
- [Reconstructing Missing Translations](#)

Working with JavaScript message bundles

A JavaScript Message Bundle is a collection of messages that are available for referencing from any JavaScript file. The JavaScript Message Bundles contain both the message texts and their translations. It provides a single central location to manage the messages and message translations of a JavaScript file.

To work with JavaScript message bundles:

1. Select a starting point and click  (JavaScript Bundle) to open the JavaScript Bundle editor.
The JavaScript Bundle editor opens.
2. Click  (Add) on the toolbar.
An editable row is added to the JavaScript Bundle editor.
3. In the Text column, enter the **Message**.
4. In the Identifier column, enter a unique identifier for the message.
5. Click .

The JavaScript Message Bundle is created and visible in Workspace Documents either in the Explorer or My Recent Documents view.

6. Right-click the **JavaScript Message Bundle** and select **Translate** from the context-menu.
The Translation Editor opens.
7. Specify the translations of the messages in the appropriate target languages.
For information, see [Translating Validation Messages](#).
8. Click .
9. In the Workspace Documents, Right-click the **JavaScript Message Bundle** and select **Publish** from the context-menu.
This creates a MLM file for the JavaScript Message Bundle.

The JavaScript Message Bundle is ready for use in any JavaScript file.

You can use the APIs provided in the MessageBundle and Localization libraries to reference the JavaScript Message Bundle. Use the static API MessageBundle.get(path) to retrieve the localization object of a JavaScript Message Bundle, and use the get(textIdentifier) API to retrieve the TranslatableText object. For more information, see Localization and MessageBundle.

Translating XForms

To be able to cater to a varied set of users, you may need to translate the XForms that you create in AppWorks Platform into various languages.

To translate XForms:

1. In the My Applications App Palette, select  (Workspace Document Explorer) to display the Workspace Documents window. You can initiate the translation process at the document level or project level:
 - In the Workspace Documents, right-click the document, and select **Translate** to open the Translation Editor.
 - In the Workspace Documents (Explorer) window, select the **XForm**, right-click and select **Translate**.
The Translation Editor <XForm> window opens displaying the existing labels and

messages in the XForm. For information about adding messages to an XForm, see [XForms Designer](#).

2. For enabling the descriptions of controls, such as Status bar, Slider, and Select box translatable, add those descriptions to the Messages tab.
For information about adding messages to an XForm, see [XForms Designer](#).
3. To add translation languages, click the Language Settings icon.
The Language Settings dialog box opens where you can add or delete languages. After adding the languages, you can set the language for a project. To do so, select the appropriate language from the Project Language list, which displays all added languages.
4. To edit the translation of a label, type the new translated label under the corresponding language column.
Ensure that the labels or messages that you define do not contain a mix of single quotation marks and double quotation marks.
5. Delete the identifiers that are not used in your project and click .
The identifiers and translations are saved as a `.mlm` file in the XML Store.

If you do not delete unused identifiers from the Translation Editor, all the identifiers will be saved to the `.mlm` file. You can delete identifiers only on the project level.

The XForm is translated.

After you complete this task:

- After translating an XForm, you must publish the project to view the translated XForm in run time.

Translating validation and UFO messages

The messages that are displayed in any XForm are attributed to one of the following sources:

- Default messages and tooltips of an XForm
- Default labels in the user interface
- Custom messages added to an XForm

Default validation messages and tooltips for the XForms

While formatting a control in AppWorks Platform, you can define a data type for the control, specify constraints, and provide validation messages for the constraints. For example, for a control of the data type integer, the default validation message will be Enter a non-empty value for {0}; where the insertion {0} is replaced with the respective values at run time.

The default messages and tooltips of an XForm are bundled together in the `cordys.Xforms.messages.xml` message bundle. To translate the messages and tooltips of the message bundle, you need to create an XML file for each translation language, and

specify the translations in it. For example, to translate into the Dutch language, create cordys.Xforms.messages_nl-NL.xml and provide the translations.

Default error messages and UFO labels in any user interface

The messages used in the Unified Feedback Objects (UFOs) are bundled as Cordys.Ufo.messages.xml as part of the installation. To provide the translation of these messages, you need to create a message bundle and add the language code as a suffix to the file name. For example, create cordys.Ufo.messages_nl-NL.xml for the Dutch language translation.

Custom messages used in the XForms

You can add custom messages to an XForm using the Messages tab available in the XForms Designer. The custom messages that you add will be available in the Translation Editor for translation.

To add custom messages to an XForm:

1. In the My Applications App Palette, select  (Workspace Document Explorer).
The Workspace Documents window opens.
You can initiate the translation process from the project context or document context.
2. Edit the translation of a message by typing the new translated message under the corresponding language column.
Ensure that the labels or messages that you define do not contain a mix of single quotation marks and double quotation marks.
3. Delete the messages that are not used in your document and click .
The messages and translations are saved as a .mlm file in the XML Store.
If you do not delete unused messages from the Messages tab, all the messages will be saved to the .mlm file.

The messages for your document are translated.

You can also provide custom messages programmatically in your applications using the localization_get() method.

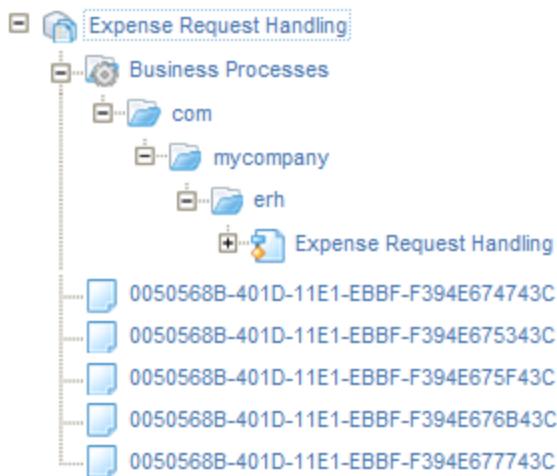
Reconstructing missing translations

The translation feature of AppWorks Platform enables you to make specific application components translatable. This includes User Interfaces, Business Process Models, Business Identifiers, and JavaScript Message Bundles.

When developing translatable application artifacts (such as User Interfaces, Business Process Models, and Business Identifiers), the translations are stored at project level. While moving such application components to a different project, within the same workspace, the relation with their translations is still intact. However, when you decide to delete the original project, a message appears saying that some artifact from the project is still in use. On verifying, you may not find any files, as the translation files are not visible to the end user.

If you continue with deleting that project, the translatable components that originate from that project are disconnected from their translations.

In BOP 4.1 CU5, the system automatically tries to reconstruct those translations whenever you open such a document. This causes those documents to be modified. In team development workspaces, this causes such documents to lock, thereby blocking others from editing the same document. Although you can save the changes to those documents, the regenerated translation files are not stored yet. This you only find out when performing other actions that require all documents to be stored, like synchronizing the workspace with the file system, validating or publishing a document, or making changes available to others. Eventually you end up with many translation related files being stored in the project hierarchy whereas they should have been part of projects themselves without being visible to end users.



This behavior has been corrected in BOP 4.1 CU6 (and later releases) by not automatically reconstructing the translation files any more. The developer is in control of creating translations and reconstructing missing translations. The downside is that the descriptions of activities in such Business Process Models cannot be shown before those lost translations are reconstructed. These problems can be solved by reconstructing the missing translations as described below.

To reconstruct the translations, you can use the Reconstruct Translations option available in the context menu of all the translatable application components.

To reconstruct translations:

1. In the My Applications App Palette, select (Workspace Document Explorer). The Workspace Documents window opens.
2. You can initiate the reconstruction process at document level or at project level in one of the following ways:
 - In the Workspace Documents, right-click the required document, and select Reconstruct Translations. This will reconstruct all the missing translations in that document.

- In the Workspace Documents (Explorer) window, right-click the required project, and select Reconstruct Translations. This will reconstruct all the missing translations for all documents contained in that project.
3. Based on your requirement, open the Translation Editor in one of the following ways:
- Right-click the relevant document and select Translate to open the Translation Editor.
 - Right-click the relevant project and select Translate to open the Translation Editor.

The Translation editor opens displaying all the non-translated items.

For JavaScript Message Bundles, the original texts are not available anymore; all reconstructed messages will show up as "Untitled Message."

4. Provide the translations for all translatable texts in the documents.
5. Click .

The translations are reconstructed.

Translating text to a target language

AppWorks Platform users spread across different geographies need to cater to the demands of the local and regional language preferences so that business is smoothly conducted. To serve the needs of its users, AppWorks Platform provides translation support. The text used within a User Interface and Business Process Model can now be translated into a preferred language using the Translation Editor.

To translate text to a target language:

1. In the Workspace Documents, right-click the required <Document> and select **Translate**.
For information on the translatable artifacts and their associated documents, see [Artifacts with Translation Support](#).
2. To provide translation at the project-level, right-click the <Project> and select **Translation > Translate**.
The Translation Editor - <document name> appears. It contains a table that lists all translatable labels available in the document or project. The labels are listed in the language specified for the project, and must be translated into the specified target languages. The table also lists the identifiers for each label.
3. To add or delete target languages, click  (Language Settings).
Alternatively, you can use the Column Chooser to hide or display the target language columns in the table.
4. To change the Language, right-click a column and select **Change Language**.
The feature that enables changing the translation languages in columns of the Translation Editor is useful in cases where the correct Project language was not set earlier or the Project language is unknown.

5. Select an identifier and click  (Lookup) in the toolbar to view the documents that use the identifier.
6. In the Translation Editor - <document name> window, specify the appropriate translations for the labels.
7. Click .
8. [Publish](#) the document.

The Translation Editor is available for translating the labels at the project level and at the document level.

If the Translation Editor is opened at a project level, the delete button () is enabled using which you can remove unused identifiers.

Artifacts with translation support

The following table contains the comprehensive list of document types and their specific artifacts that have translation support. Auto suggest is also an inbuilt feature for the same.

Document Type	Translatable Artifacts
User Interface	<ul style="list-style-type: none"> ■ User Interface description ■ Control labels ■ Custom messages
JavaScript Message Bundle	Message texts
Business Process Model	<ul style="list-style-type: none"> ■ Business Process Model description ■ Activity description ■ Notification Subject of Human Task Activity ■ Transparent text, text annotation, and text descriptions. ■ Vertical and horizontal lane descriptions. ■ Decision and case model descriptions.
Business Identifier	Business Identifier description
Entities	<ul style="list-style-type: none"> ■ Display Names of Entities ■ Display Names of Properties ■ Display Names of Relationships ■ Display Names of Lists ■ Display Names of Actions
	See the <i>AppWorks Platform Low-Code Design Guide</i> .

Setting language preference at project level

To set a language preference at the project level:

1. In Workspace Documents, do any one of the following:
 - Right-click the <User Interface> and select **Translate**.
 - Right-click the <Business Process Model> and select **Translate**.
 - Right-click the <Project> and select **Translation > Translate**.
The Translation Editor - <User Interface/Business Process Model> opens.
 2. In the toolbar, click  (Language Settings).
The Language Settings dialog box opens.
 3. Click  (Add) to add a preferred language.
A new row appears.
 4. Type the language name under the Language column.
AppWorks Platform allows the usage of **language codes** specified by the International Standards Organization (ISO) only. You can also view the language codes at [MSDN](#).
 5. Click .
- The preferred language is added to the translation repository and appears in the Translation Editor - <User Interface/Business Process Model>.

To delete a language:

- In the Language Settings dialog box, select the check box corresponding to the language and click  (Delete).

To edit a language preference:

- Click the relevant language and modify as required.

To change the project language:

Project Language denotes the language in which the project is developed. This is English by default.

1. To change the project language, right-click <Project> and select **Language Settings**.
2. Select a language from **Project Language**.
3. You can create more than one preferred language.

Enabling multi-browser support for web content

AppWorks Platform provides you with the functionality to create Web applications that are accessed and used through browsers. However, until recently, the Web applications were supported only on a single browser, the Microsoft Internet Explorer. With other browsers gaining popularity, it has now become essential for Web applications to be able to support and perform on multiple browsers. In order to support multiple browsers, the Web

applications must be compliant with W3C standards, as this ensures that most browsers render the Web content of the Web application properly.

AppWorks Platform now provides an enhanced functionality to create XForms that are standards compliant and hence supported by multiple browsers. You can also convert an existing XForm to a standards-compliant XForm that can be rendered in multiple browsers. However, a Web application may comprise various other content types that must also be made standards compliant. This section provides information on making the following content types standards compliant.

- XForms
- HTML
- Library or Type Library
- JavaScript (.js) files
- Style sheets (.css files)
- Dynamic XForms

A Web application that is standards compliant is also referred to as a Transitional mode application. Web applications that do not comply to standards are known as Quirks mode applications.

AppWorks Platform provides multi-browser support for the following modes of the listed browsers.

Browser	Supported Mode
Internet Explorer 6.0 SP2	Quirks and Transitional
Internet Explorer 7.0	Quirks and Transitional
Internet Explorer 8.0	Quirks and Transitional
Firefox 3.5 and later	Transitional
Chrome 1 and later	Transitional
Safari 4.0 and later	Transitional

Note: The Safari 4.0 browser version supports only the Basic Authentication mode. The Safari 4.0 browser does not support the Integrated and NTLM authentication modes. Additionally, AppWorks Platform supports HTML 4.0.1 and JavaScript 1.3 and later for rendering Web content in all the above browsers.

As shown in the table, standards-compliant Web content is supported in all the above browsers. While, other Web content that is not standards-compliant can be viewed only using the Internet Explorer browsers.

In order to create Web content for a transitional mode application, some changes are necessary in the way the various content types that constitute the application are defined and used. Standardizing the way Web content is used while creating an application, enables

it to be rendered on various browsers. This section provides information about the changes that are required while standardizing the following major content types.

- [Application or Form changes](#)
- [Script, Style, and XML data island changes](#)
- [Type Library changes](#)
- [HTML and JavaScript changes](#)
- [XML changes](#)
- [Event Model changes](#)
- [Regular Expressions changes](#)
- [Style changes](#)

Application or form changes for multi-browser support

An application or an XForm must be standards-compliant for it to be supported in multiple browsers.

XForm level changes

To make an XForm Standards compliant, open the XForm in the XForms Designer, double-click it, and select Transitional in the property sheet that opens. This ensures that all content in the XForm is modified to support multiple browsers.

Application level changes

To make an application Standards compliant:

- Ensure that DOCTYPE is declared in the beginning of the application. This indicates that the content can be rendered in the Standards mode on most browsers.
The namespace URL, `eibus` with `xmlns` as its prefix, which is usually set on a Web application, is no longer required.

As an example, consider the following sample code.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode
<html xmlns:eibus>
  <head>
    <script type="text/javascript" src="/cordys/wcp/application.js"></script>
  </head>
  <body>
  </body>
</html>
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<script type="text/javascript" src="/cordys/wcp/application.js"></script>
</head>
<body>
</body>
</html>
```

Script, style, and XML data island changes for multi-browser support

For Web content in an application to render and work properly in multiple browsers, the scripts and style tags, and the XML data islands in the Web content must be modified to be standards-compliant.

To do so, proper type attributes must be set for all script, style, and XML tags, as shown:

- <script type="text/javascript" src="...">
- <style type="text/css">
- <link rel="stylesheet" type="text/css" href="..."/>

For XForms, type attributes for the script, style, and XML tags are automatically provided if the Transitional property is selected in the XForms property sheet.

Type library changes for multi-browser support

To ensure multi-browser support for HTML content, ensure that:

- The <eibus> tag is replaced with <div> tag.
- The <div> tag is placed in the BODY (<body> tag) of the HTML content.
- The <div> tag contains the cordysType attribute that refers to the fully-qualified name of the type library.

As an example consider the following sample codes.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode <head>
<eibus:dataisland id="busDataIsland1" automatic="true"/> </head> <body> </body>
```

New Usage

```
\New Usage: Standards compliant sample code in Transitional Mode <head> </head>
<body> <div id="busDataIsland1" cordysType="wcp.library.data.BusDataIsland"
automatic="true"/> </body>
```

Additionally, to ensure standards compliance:

- All Web Libraries must be converted to Type Libraries.
- You must use the fully qualified name to refer to each AJAX Toolkit type library.
- Libraries must now be identified in the following manner. Earlier, the `getElementsByTagName()` method was used to identify libraries by their tag names.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode var splitters =
document.getElementsByTagName("splitter"); for (var I = 0; I < splitters.length;
I++) { var splitter = splitters[i]; }
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode var splitters =
document.getElementsByTagName("div"); for (var I = 0; I < splitters.length; I++)
{ var splitter = application.hasType(splitters[i], "wcp.library.ui.Splitter") ?
splitters[i]: null; }
```

- Prototypes declared on a Type Library must be standards compliant as shown below.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode function
SampleObject { } function SampleObject.prototype.TestMethod(parameter) { }
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode function
SampleObject { } SampleObject.prototype.TestMethod = function(parameter) { }
```

HTML and JavaScript changes for multi-browser support

The HTML and JavaScript content types need to be modified to enable multi-browser support. This involves change in the way API are used in applications. The various changes required are as mentioned below.

Accessing HTML elements in JavaScript

The following properties apply to a HTML element accessed through JavaScript, with the exception of the window JavaScript object.

Standards non-compliant properties (Used in Quirks Mode)	Standards compliant properties (Used in Transitional Mode)
document <i>Syntax</i> <pre>var documentObject = htmlElement.document;</pre>	ownerDocument <i>Syntax</i> <pre>var documentObject = htmlElement.ownerDocument;</pre>
parentWindow <i>Syntax</i> <pre>var documentObject = htmlElement.document.parentWind ow;</pre>	defaultView <i>Syntax</i> <pre>var documentObject = htmlElement.ownerDocument.defaultView;</pre>
parentElement <i>Syntax</i> <pre>while (element != null) { element = element.parentElement; }</pre>	parentNode <i>Syntax</i> <pre>while (element != null && element.nodeType != 9) { element = element.parentElement; }</pre> The additional check on nodeType ensures document root is not included while searching
children <i>Syntax</i> <pre>var childNodes = htmlElement.children;</pre>	childNodes <i>Syntax</i> <pre>var childNodes = htmlElement.childNodes;</pre>
outerHTML <i>Syntax</i> <pre>var outerHTML = htmlElement.outerHTML;</pre>	This has no equivalents, although one possibility could be to use innerHTML of the parent, though the solution may not return same results. For example, <pre>Var outerHTML = htmlElement.parentNode.innerHTML;</pre>

The following events apply to HTML elements accessed through JavaScript.

Standards non-compliant events (Used in Quirks Mode)	Standards compliant events (Used in Transitional Mode)
onselectevent is used on the application. <i>Syntax</i>	onapplicationselect event is now used on the application. <i>Syntax</i>

Standards non-compliant events (Used in Quirks Mode)	Standards compliant events (Used in Transitional Mode)
<pre><html onselect="onSelectApplication () " ... Or application.addListener ("onselect", handler);</pre>	<pre><html onapplicationselect="onSelectApplication () " ... Or application.addListener ("onapplicationselect", handler);</pre>

Accessing custom HTML properties in JavaScript

In HTM content, it is possible to set custom properties for an HTML object and access them in JavaScript. While the properties can be set as before, for enabling multi-browser support, they must be read using the `getAttribute()` method.

Consider the following example,

```
<input1 id="input1" customproperty="customvalue" ...
```

Earlier, in Quirks Mode, the properties were viewed as,

```
var customvalue = input1.customproperty;
```

In the Transitional Mode, the properties must be viewed as below to ensure multi-browser support,

```
var customvalue = document.getElementById("input1").getAttribute("customproperty");
```

This can be set when the properties are read from the HTML control for the first time, as follows.

```
var input1 = document.getElementById("input1"); if (undefined ===
input1.customproperty) input1.customproperty == input1.getAttribute
("customproperty");
```

Using the "===" operator ensures the data checked is NULL instead of FALSE.

Using standard JavaScript methods

The usage of the following standard JavaScript methods must be modified as mentioned below.

Standards non-compliant methods (Used in Quirks Mode)	Standards compliant methods (Used in Transitional Mode)
<p>create() method for an HTML element: The parameter can be an HTML markup.</p> <p>Syntax</p> <pre>var html = window.document.createElement ("<div style='display:inline'>");</pre>	<p>create() method for an HTML element: The parameter can only be the tag name, following which the other properties must be set programmatically.</p> <p>Syntax</p> <pre>var html = window.document.createElement ("div"); html.style.display = "inline";</pre>
<p>create() method for the IFrame control: The width and height need not be specified.</p> <p>Syntax</p> <pre>var iframe = window.document.createElement ("<iframe>");</pre>	<p>create() method for the IFrame control: The width and height must be specified.</p> <p>Syntax</p> <pre>var iframe = window.document.createElement ("iframe"); iframe.style.width = "100%"; iframe.style.height = "100%";</pre>
<p>insertBefore() method: The second parameter is optional. If not set, the behavior of the control is similar to the appendChild() method.</p> <p>Syntax</p> <pre>document.body.insertBefore (newChild);</pre>	<p>insertBefore() method: The second parameter is not optional. Set to null, if not used.</p> <p>Syntax</p> <pre>document.body.insertBefore(newChild, null);</pre>
<p>HTML controls: It was possible to access all HTML controls in a Web application using the window["id"] collection.</p> <p>Syntax</p> <pre>htmlControl1.style.display = "none";</pre>	<p>HTML controls: Controls on a Web application can only be accessed using the getElementById ("id") method.</p> <p>Syntax</p> <pre>var htmlControl1 = document.getElementById ("htmlControl1"); htmlControl1.style.display = "none";</pre>
<p>allcollection: it returns an array of HTML controls referred by ID or name.</p> <p>Syntax</p> <pre>var html1collection = document.all ["html1"]; var html2collection = htmlControl1.all["html2"];</pre>	<p>No equivalent available for this method for browsers. It is possible to use the getElementById("") method to return a single HTML control.</p> <p>Syntax</p> <pre>var html1 = document.getElementById ("html1");</pre> <p>AppWorks Platform provides an API that is an equivalent of the element.all["id"] method and is used as shown below:</p> <pre>var html2 = cordys.getElementById</pre>

Standards non-compliant methods (Used in Quirks Mode)	Standards compliant methods (Used in Transitional Mode)
	("html1");
<p>Retrieving data from a collection: Standard methods that use parentheses - "(" and ")" - are used.</p> <p><i>Syntax</i></p> <pre>var row = sampleTable.rows("row1"); var child = xmlObject.childNodes(2);</pre>	<p>Retrieving data from a collection: Square brackets - [] - must be used.</p> <p><i>Syntax</i></p> <pre>var row = sampleTable.rows["row1"]; var child = xmlObject.childNodes[2];</pre>
<p>eval() method: Is used as below to evaluate specified expression.</p> <p><i>Syntax</i></p> <pre>var newObject = oldObject.cloneNode(true);</pre>	<p>cloneNode() method: Once used, copies all function pointers and custom attributes on the cloned object.</p> <p><i>Syntax</i></p> <pre>var newObject = oldObject.cloneNode(true); newObject.property = oldObject.property; ...</pre>

XML changes for multi-browser support

The XML content in an application must be made standards compliant to enable multi-browser support.

Requesting XML object through XML HTTP

Different browsers require different implementations of XMLHTTP for requesting an XML object. To make it Standards compliant, AppWorks Platform provides a standard API to call the `ActiveXObject()` method, so that the application need not be hard coded by users for supporting various browsers.

As an example, consider the following sample codes.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode var myConnection =
new ActiveXObject("Microsoft.XMLHTTP");
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode var myConnection =
cordys.getConnection();
```

XML content in the <head> tag

XML Web content (data islands) in the `<head>` sections of an application do not render properly with all browsers. To resolve this, a new type namespace is introduced that wraps XML islands with a `<script>` tag.

As an example, consider the following sample code.

Old usage

```
\\"Old Usage: Standards non-compliant sample code in Quirks Mode <xml
id="applicationmanager"> <Application toolbar="true">
<icon>images/applicationmanager.gif</icon>
<url>debugger/applicationmanager.htm</url> <id>ApplicationManager</id>
<description>ApplicationManager</description> <caption>ApplicationManager</caption>
<frame>headerToolbar</frame> </Application> </xml>
```

New usage

```
\\"New Usage: Standards compliant sample code in Transitional Mode <script
type="cordys/xml" id="applicationmanager"> <Application toolbar="true">
<icon>images/applicationmanager.gif</icon>
<url>debugger/applicationmanager.htm</url> <id>ApplicationManager</id>
<description>ApplicationManager</description> <caption>ApplicationManager</caption>
<frame>headerToolbar</frame> </Application> </script>
```

In the previous example, the wrapper object represented by "id" is displayed only after the page has been loaded. Due to this, it is not possible to use the XML document in the global JavaScript code that is executed when a Web page is loaded.

The Standards-compliant code is converted to a valid XML document by all browsers that can be accessed by `XMLDocument`, and is available for use as required.

Consider the following sample code for accessing an XML document in an application or form.

Old usage

```
\\"Old Usage: Standards non-compliant sample code in Quirks Mode var sampleXMLObject
= applicationmanager;
```

New usage

```
\\"New Usage: Standards compliant sample code in Transitional Mode var
sampleXMLObject =
applicationmanager.XMLDocument;
```

No support for `src` attribute

The `src` attribute is not supported by all browsers. Alternatively, use the `load()` method of XML Document to load XML from a URL.

AppWorks Platform XForms automatically generates XML in this format when HTML is rendered at runtime.

Usage of `.documentElement` and `.xml` properties

The `.xml` property that is used to serialize an XML node is supported by the Internet Explorer browser only. To make it Standards compliant, a new wrapper, `cordys.getXML()`, is introduced to serialize XML into a string. The `.documentElement` property must be used only on the `XMLDocument` and not on the wrapper object itself.

Consider the following sample code.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode var documentElement  
= myXML.documentElement; var xml  
= myXML.xml;
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode var  
documentElement = myXML.XMLDocument.documentElement; var xml =  
cordys.getXML(myXML.XMLDocument);
```

Comments in XML data island

Comments on an XML data island must be placed inside the XML and not between the `<script>` and the first `<xml>` tags, otherwise the comment will be considered as part of the `XMLDocument`. Also, the `<script type="cordys/xml">` tag must contain a single child tag.

Consider the following sample code.

Old usage

```
\Old Usage: Standards non-compliant sample code in Quirks Mode <script  
type="cordys/xml" id="someXML">  
<!-- This is some comment which should not be placed here --> <someXML> </someXML>  
</script>
```

New usage

```
\New Usage: Standards compliant sample code in Transitional Mode <script  
type="cordys/xml" id="someXML"> <someXML>  
<!-- comment is allowed here--> </someXML> </script>
```

XML document manipulations

For creating, loading, cloning, appending, and selecting XML documents, AppWorks Platform provides wrapper APIs that must be used on top of the XML.

The attributes of an XML document can be accessed only as an array on multi-browsers, and cannot be used to iterate through the collection (only supported Internet Explorer).

Consider the following sample code.

Old usage

```
\\"Old Usage: Standards non-compliant sample code in Quirks Mode var attributes =
xmlNode.attributes; for ( var I = 0, length = attributes.length;
I < length; I++ ) { var attribute = attributes[i]; ... }
```

New usage

```
\\"New Usage: Standards compliant sample code in Transitional Mode var attributes =
xmlNode.attributes; for (var attribute =
attributes.nextSibling(); attribute ;attribute = attributes.nextSibling()) { ... }
```

XPATH expressions

When using the `selectSingleNode()` and `selectNodes()` methods to search an XML document, the use of XSLPattern is not supported by multiple browsers. To ensure multi-browser support, use XPath instead.

Consider the following sample code. Here, `<key>` is associated with the `http://schemas.cordys.com/1.1/ldap` namespace. While using XPath, the selection criteria must be set as `././ldap:key`, but the Internet Explorer browser ignores the namespace and the `./key` expression.

Old usage

```
\\"Old Usage: Standards non-compliant sample code in Quirks Mode <xml
id="requestGetUserDetails"> <SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Body> <GetUserDetails
xmlns="http://schemas.cordys.com/1.1/ldap"> <key>userfound</key> </GetUserDetails>
</SOAP:Body> </SOAP:Envelope> </xml> var request =
requestGetUserDetails.XMLDocument.documentElement; var key = request.selectNodes
("././key"); var envelope = request.selectNodes("././ancestor(SOAP:Body)");
```

New usage

```
\\"New Usage: Standards compliant sample code in Transitional Mode <script
type="cordys/xml"> <xml id="requestGetUserDetails"> <SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Body> <GetUserDetails
```

```

xmlns="http://schemas.cordys.com/1.1/ldap"> <key>userfound</key> </GetUserDetails>
</SOAP:Body> </SOAP:Envelope> </xml> </script> var request =
request GetUserDetails.XMLDocument.documentElement; var ownerDocument =
request.ownerDocument; cordys.setXMLNamespaces(ownerDocument,
{"ldap": "http://schemas.cordys.com/1.1/ldap",
"SOAP": "http://schemas.xmlsoap.org/soap/envelope/" } ); var key =
cordys.selectXMLNodes(request,"./ldap:key"); var envelope= cordys.selectXMLNodes
(request,"./ancestor::SOAP:Body");

```

In some cases, it may be required to select a node with a specific tag Name, without knowing if a (default) namespace is defined in the XMLDocument. For example, an the Application Definition can be of a local XML Data island on the Web page without a namespace, or a document from the XML store with xmlstore as its default namespace. As an alternative to ignoring namespaces in the XPath query language, you can use the `local-name()` method in your XPath query.

As an example, consider the following sample code. The sample code returns the `frame` element regardless of the namespace to which it is associated.

Old usage

```

\\Old Usage: Standards non-compliant sample code in Quirks Mode
cordys.setXMLNamespaces(dataNode.ownerDocument, {
"xs": "http://schemas.cordys.com/1.0/xmlstore" } ); var frameNode =
cordys.selectXMLNode(applicationDefinition,"xs:frame");

```

New usage

```

\\New Usage: Standards compliant sample code in Transitional Mode var frameNode =
cordys.selectXMLNode(applicationDefinition,"*[local-name()='frame']");

```

You can learn more about the XPath syntax at <http://www.w3schools.com/xpath>

Event model changes for multi-browser support

So far, `window.event` has been used for standard HTML events in accordance with the Internet Explorer browser. In other browsers, `eventObject` is the first parameter passed when an method is executed. To support multi-browsers, all methods must pass the event object or must use the global object if no event object is present.

To ensure multi-browser support, you can use one of the following options to attach methods to events.

- Methods can be attached to events using only a function pointer, as shown.

```

myDiv.onclick = eventHandler;

```

You can also attach events to methods through HTML, as shown.

```
<div onclick="eventHandler(event)"/>
```

The `onclick` event, when used with the `Contextmenu` library, is not supported by browsers other than the Internet Explorer. For multi-browser support, ensure that you use the `onmenuitemclick` event instead.

- The earlier way of attaching methods to events through HTML may not be supported on all browsers. AppWorks Platform provides the following generic method that is supported by all browsers and must be used programmatically to attach the events.

```
cordys.addDOMListener(myDiv, "onclick", eventHandler);
```

After an event is attached to a method, it can be accessed in a method as given below.

```
function eventHandler(eventObject) { if (! eventObject ) eventObject = window.event;
}
```

In cases where the `srcElement` of an event is not the same as the element that raised the event, you can pass the pointer to the inline methods as shown below.

```
<div onclick="inlineEventHandler(event,this)">
```

If the `onmenuitemclick` event is used with the `Contextmenu` library:

- The `this` property does not refer to the menu item, but to the `contextmenu` definition object.
- The `event` property refers to the custom `eventObject`. The properties available for use are as follows:
 - `event.activeElement`: Refers to the HTML element to which the `contextmenu` is attached.
 - `event.srcEvent`: Refers to the event associated with the `onmouseup` action on the visible HTML of the `contextmenu`.

Handling events on DIV

Browsers, other than the Internet Explorer browser, cannot properly handle events such as `onfocus` and `onblur` when present in a `<div>` tag. To enable browsers to handle these events, add `tabIndex=0` (or any higher number other than 0) to the `<div>` tag.

Event properties

Different browsers handle events in different ways. Therefore, a variety of event properties are available and their usage differs from browser to browser. For example, the event used

to determine the position of a mousedown or identify the mouse button clicked, differs from browser to browser.

The following table provides information about the events available on various browsers.

- v denotes that the event property is available for the respective browser.
- n/a denotes that the event property is not applicable to the respective browser.

Event Property	Description	Description	Firefox	Chrome	Safari
button	Invokes a mouse button. For example, when used in the mousedown event.	0=default (=n/a) 1=Left 2=right 4=middle	0=Left(default) 1=middle 2=right	0=Left (default) 1=middle 2=right	0=Left (default) 1=middle 2=right
offsetX	Refers to the coordinates relative to object.	v	n/a	v	v
wheelDelta	Retrieves the distance and the direction in which the wheel button has rolled.	v	n/a	v	v
detail	Retrieves the distance and the direction in which the wheel button has rolled.	n/a	v	n/a	n/a
layerX	Refers to the coordinates relative to the object in the current layer.	n/a	v	v	n/a
clientX	Refers to the coordinates relative to the client area.	v	v	v	v
pageX	Refers to the coordinates relative to the page as visible in the application window, disregarding the scrolled area.	n/a	v	v	v
x	Refers to the coordinates relative to the specified parent element.	v - calc. diff. compared chrome/safari	n/a	v	v
screenX	Refers to the coordinates relative to the screen (main screen, in case of dual screen support).	v	v	v	v
type	Refers to the type of event, for example, mousedown.	v	v	v	v
srcElement	Refers to the object that fired the event.	v	n/a	v	v
target	Returns the reference of the target to which an event is originally dispatched.	n/a	v	v	v
currentTarget	Refers to the current object that is the target of the event.	n/a	v	v	v

Event Property	Description	Description	Firefox	Chrome	Safari
explicitOriginalTarget	Refers to the original target of the event.	n/a	v	n/a	n/a
originalTarget	Refers to the original target of the event, prior to any changes that may have been made to the target.	n/a	v	n/a	n/a

Regular expression changes for multi-browser support

To be able to work with applications in multiple browsers, all content types in the application must be Standards compliant.

Regular expressions in an application are constructed in the following ways:

- Using a regular expression literal, such as `var re = /ab+c/;`.
- Calling the constructor function of the `RegExp` object, such as `var re = new RegExp ("ab+c");`.

Different browsers behave differently when using regular expressions. In the Internet Explorer browser, a new `RegExp` object is created each time the regular expression is used, while in other browsers a global `RegExp` object is created that is used each time. This can result in issues in the search functionality of the browsers.

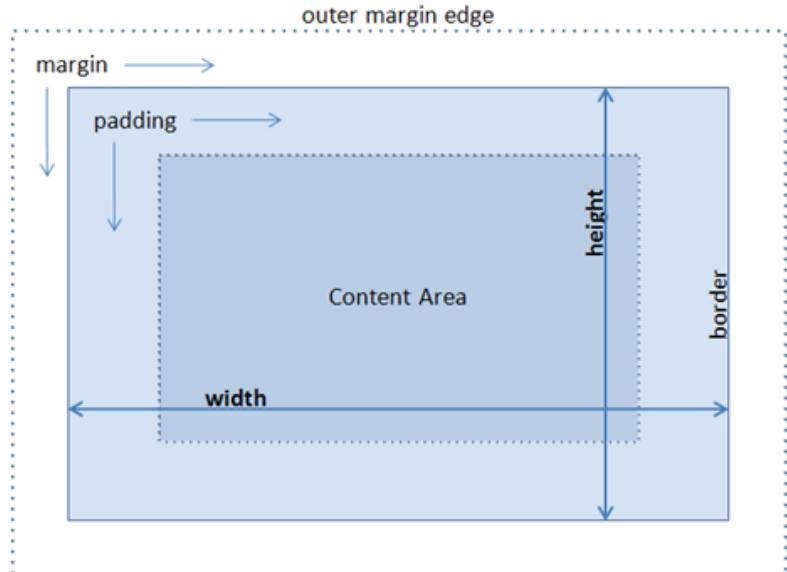
To avoid such issues:

- Always create a new `RegExp` object.
- Use a global `RegExp`, and reset the property `regexp.lastIndex = 0` every time before calling `exec()`.
- Avoid using the `/g` flag (global search) unless really needed.

Style changes for multi-browser support

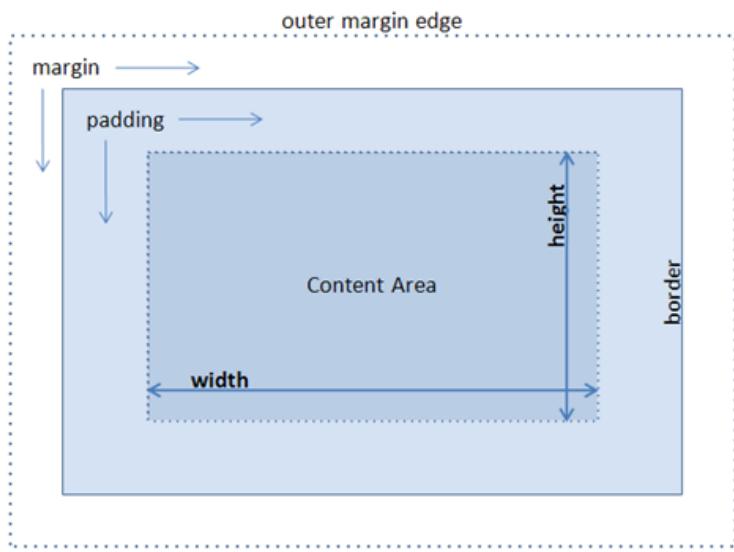
Styles are interpreted differently in different browsers. While it is possible to handle a few style aspects properly for all browsers, for most aspects this can be managed through clever application design. This topic provides information about differences in the rendering of styles (box models) in different browsers.

In case of a Quirks Mode application, which is not standards compliant, a Web page layout is rendered based on the following box-model representation.



Quirks Mode

In Quirks mode, the width and height that you specify, includes padding and border. However, in case of the standards-compliant Transitional Mode, the width and height refer only to the content area and do not include padding and border, as shown in the following box model.



Transitional Mode

The box model used affects the rendering of elements, according to properties like width, height, margin, or padding that are assigned to it. To avoid variances in the rendering of elements in different browsers, it is important that the box models for all elements

associated with width, height, margin, or padding be corrected to reflect the Transitional mode. Some common changes are described in the sections below.

Total rendered extent of an element

In the Quirks mode, if no margin is specified, the rendered extent of an element is the exact height and width specified. Typically, padding is used to create space, and the border to give a visual boundary to the element. The padding and the border must be calculated such that the total rendered extent of element is equal to the width and height values.

However, in the Transitional mode, the rendered element does not include the padding and border and these are added to form the element's rendered extent.

Hiding of elements by setting width and height to '0'

In cases where width is set as 0 (zero), transitional-mode applications continue to display the padding and border, since they fall outside the element. As negative values for width are not supported, you can use `element.style.display = 'none'` to completely hide the element.

Specifying 100% height and width

In the Transitional mode, when you specify an element to be 100% in width, and the element has a border, padding or margin, the rendered extent of the element will be more than 100%.

For example, consider an iframe inside a parent DIV that is 500px wide and 200px high. If you specify the iframe dimensions to be 100% wide and 100% high, and the iframe has a 1px border, you will see a scroll bar on the DIV.

This is because the rendered width of the iframe itself is 500px, that is 100% of its parent DIV, and the borders on the left and right-hand side of the iframe, making the effective rendered width 502px, take up an additional two pixels. By the same logic, the effective rendered height is 202px.

For such a case, it is advisable to use a negative-margin on the iframe to compensate for the border. Use of padding is not feasible, as padding does not support negative values properly. Setting `style="margin:-1px"` on the iframe will compensate the 1px borders and make the iframe fit the parent DIV, without displaying scroll bars.

Another approach that can be used in the above scenario is to use "edge-positioning," that is to set `position: absolute` and define the four offsets for the iframe as follows.

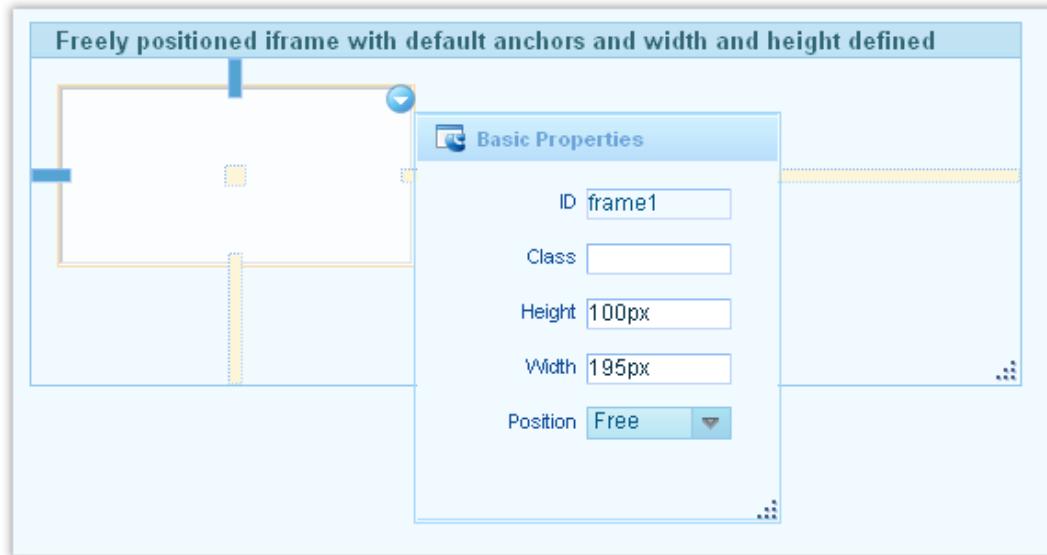
```
\\"New Usage: Standards compliant sample code in Transitional Mode
iframe.customstyle
{ position:absolute; top:1px; left:1px; right:1px; bottom:1px; }
```

If you are using the XForms Designer, you need not specify custom CSS properties, as these can be defined through the property sheet of the XForm. For details, see the next section about the use of absolute positioning and anchoring.

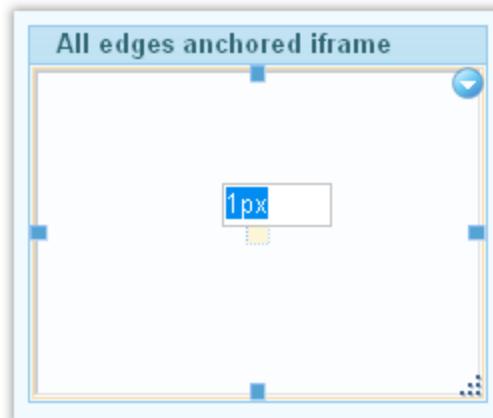
Use of absolute position and anchoring in XForms Designer

A control is said to have the absolute CSS position if its position is set as Free and the layout of its parent control or XForm is also set as Free in the XForms Designer. In such a case, it is possible to apply anchors to the control and also specify the top, right, bottom, and left offsets, as desired. The width and height of a freely-positioned control is picked automatically based on the extent of control as seen in the XForms Designer.

If a freely-positioned control is not being displayed with the proper width and height, you may check and specify the correct width and height values in the Basic Properties of the control in XForms Designer. For a freely-positioned control, a default anchor appears on two adjacent edges as seen from the anchoring guides, and must have positive values of width and height as seen from the Basic Properties.



Now, if all four sides are anchored, the XForms designer considers all four offsets and does the required adjustments to make the control fit the parent, an exception being the case where '0' is defined as the offset for a control that has 1px border, as is the case with the above iframe example. In such cases, an offset of '1' must be used.



Scroll bars appearing in pages migrated to transitional mode

When a page is migrated from C3 and is updated according to the Transitional mode, it may happen that certain areas of the page start displaying scroll bars. This may be due to the changes made to the box model; refer to the sections above to correct the problem.

If the issue persists, it is recommended to check the widths and heights of respective controls. You may also choose to reset the widths to 'default' from the context menu of control, or may assign explicit width values that do not result in scroll bars.

For transitional-mode pages, it is imperative that the controls have heights explicitly specified; the same must be verified in the XForms Designer.

Using the quirks box model

The CSS3 property box-sizing, enables you to choose the box model to employ for a particular element. You can use the box-sizing property as follows.

```
\\"New Usage: Standards compliant sample code in Transitional Mode -moz-box-sizing : content-box; -webkit-box-sizing : content-box; /* sets the box model to CSS2.1 or transitional box model */ -moz-box-sizing : border-box; -webkit-box-sizing : border-box; /* sets the box model to quirks box model */
```

This is a rather recent property and the AppWorks Platform XForms pages do not make use of this so far. All AppWorks Platform pages follow CSS2.1 Transitional box model only. This property is recorded here as a helpful tip but not as recommendation or example of usage.

Properties and syntax

Unitless values in quirks mode

In the Quirks mode, the values without units are automatically assumed to have units. In transitional mode, the units need to be specified to have the correct value picked up.

```
\\"Old Usage: Standards non-compliant sample code in Quirks Mode \\"Quirks model accepts (but bad practice) h1 {color: FF0000;} /* missing the octothorpe!, but interpreted as #FF0000 */ h2 {font-size: 18 px;} /* one too many spaces, but interpreted as 18px */ div {width: 500;} /* where's the unit? interpreted as 500px */
```

In Transitional mode, this is not acceptable, and the units have to be specified explicitly.

```
\\"New Usage: Standards compliant sample code in Transitional Mode \\"Transitional mode needs correct units h1 {color: #FF0000;} h2 {font-size: 18px;} div {width: 500px;}
```

Width of inline elements

In Quirks mode, an inline element can have a width, as follows:

```
span with  
width:  
100px;
```

In Transitional mode, the width of an inline element is ignored:

```
span with width: 100px;
```

This difference applies to all inline elements, whether they are inline by semantics (for example, span) or a block-level element that is specified as inline through CSS (for example, `<div style="display:inline">`).

Overflow:visible property

In an element with a fixed height and associated with the `overflow:visible` property (which is default), if the content is too long then:

- In Quirks mode, it stretches up the elements to accommodate all the content.

```
This paragraph has height: 100px;, and a lot of content. Does the browser rigidly adhere to this height, or does it stretch the element to contain all the content?  
Content  
Content  
Content  
Content  
Content  
Content
```

- In Transitional mode, the content should flow out of the element.

```
This paragraph has height: 100px;, and a lot of content. Does the browser rigidly adhere to this height, or does it stretch the element to contain all the content?  
Content  
Content  
Content  
Content  
Content
```

Cursor:pointer property

Firefox does not recognize the `cursor:pointer` style property. Use the `cursor:hand` style property instead.

Filters

Filter is an Internet Explorer specific CSS property that does not work on any other browser. Internet Explorer Filters may be used for achieving common effects like setting opacity, making gradient, and drawing shadows on elements.

Opacity

Modern browsers accept the CSS 3 property 'opacity', but the Internet Explorer still uses the Alpha filter for opacity. In order to be multi-browser complaint, it is advisable to use all declarations, as shown in below example where an opacity of 60% is achieved.

```
\New Usage: Standards compliant sample code in Transitional Mode
filter:progid:DXImageTransform.Microsoft.Alpha(opacity=60); opacity:0.6;
```

Gradient

Gradient filter is not supported in any browser except Internet Explorer. To support gradient, create a PNG image that can be set as background image and repeat it in x or y direction as per the effect desired.

For example, you can create an image that looks like:



Add the following style properties to the DIV or element that should have the gradient effect.

```
\New Usage: Standards compliant sample code in Transitional Mode width:100px;
height:40px; border:1px solid; background-image:url('gradient.png'); background-
repeat:repeat-x;
```

The PNG images are rendered with a blue box around the transparent areas in Internet Explorer 6. See last section for details.

A GIF image can also be used in place of the PNG image. However, GIF images,

- Do not support alpha transparency that is important for the background color to show through.
- Cannot be used on different background color, and requires different images for different background colors.
- Show flakes if placed on an area other than it is designed for.

The PNG images are more generic and can be reused on any range of backgrounds.

Spaces in filter

In Transitional mode, you must not use spaces in a filter.

```
\Old Usage: Standards non-compliant sample code in Quirks Mode \ unacceptible
usage with a space after ':' filter: progid:DXImageTransform.Microsoft.Gradient
(GradientType=0,StartColorStr= '#20ffcd00',EndColorStr= '#30ffcd00');
```

The CSS properties following such a filter, including the filter itself, are ignored. Ensure there is no space after ":" in style property, or after "=" of the inline attribute for filter.

```
\New Usage: Standards compliant sample code in Transitional Mode \\ acceptable  
usage without any spaces filter:progid:DXImageTransform.Microsoft.Gradient  
(GradientType=0,StartColorStr='#20ffcd00',EndColorStr='#30ffcd00');
```

Text properties

CSS properties such as word-wrap and text-overflow are supported by Internet Explorer, but not by other browsers. As a result, browsers other than the Internet Explorer may display overflowing text and no ellipsis. You can prevent the overflow of text by using the correct width/height and overflow combination. Word-wrap is supported in Firefox version 3.5 and later.

Browsers other than the Internet Explorer do not support the writing-mode: tb-rlCSS property that is used for writing vertical text. There is a possibility to use proprietary Firefox and webkit properties as shown below.

```
-webkit-transform: rotate(90deg); -moz-transform: rotate(90deg);
```

These are available in Firefox 3.5+, Apple Safari 4+, and Google Chrome2+.

Rounded corners using border

The CSS 3 specification recommends a new property border-radius. It is not supported in most browsers yet. However, Firefox and webkit based browsers like Chrome and Safari have implemented their own proprietary properties. So the following code can be used to achieve rounded corners in Firefox and webkit based browsers.

```
-moz-border-radius: 5px; -webkit-border-radius: 5px
```

This is not supported in Internet Explorer.

There is a slight difference between Firefox and webkit based browsers when it comes to create rounded corners selectively. The syntax for creating specific corners is as follows.

```
-moz-border-radius-topleft : 5px; -webkit-border-top-left-radius : 5px; -moz-border-radius-topright : 5px; -webkit-border-top-right-radius : 5px; -moz-border-radius-bottomleft : 5px; -webkit-border-bottom-left-radius : 5px; -moz-border-radius-bottomright : 5px; -webkit-border-bottom-right-radius : 5px;
```

Changing scroll bar colors

In Internet Explorer, you can change the style of the scroll bars. To do so, you can use the CSS properties as shown below.

```
\New Usage: Standards compliant sample code in Transitional Mode scrollbar-3dlight-  
color: #f7f9f5; scrollbar-arrow-color: #406593; scrollbar-darkshadow-color: #f7f9f5;  
scrollbar-face-color: #dde8f9; scrollbar-highlight-color: #c2d8f2; scrollbar-shadow-  
color: #c2d8f2; scrollbar-track-color: #f7f9f5;
```

All the above mentioned styles are Internet Explorer specific and are not supported by Firefox. There seems to be no alternative for Firefox. Apple Safari 4 supports styling of scrollbars, other webkit based browsers like Chrome do not support it in current release, but may support it in future releases.

PNG images

The PNG images are not rendered correctly in Internet Explorer 6 because the Internet Explorer 6 does not support Alpha transparency in PNG images.

Alpha transparency adds an additional channel in the PNG image to handle semi-transparent areas. This helps in smoothing the edges and achieving transparent gradients in images.

The only downside of using PNG images in Internet Explorer 6 is a 'blue' box that is rendered. This does not affect the functionality in any way, but mars the overall appearance of page.

The modern browsers, Internet Explorer 7, and later versions of the Internet Explorer browser support PNG images properly.

Accessing XForms and translations using URLs

This topic describes accessing XForms and translations using URLs.

Accessing XForms

At runtime, an XForm may be used as a standalone XForm, or as an application in the AppWorks Platform application. In both the cases, you can access XForms from a browser by specifying its Uniform Resource Locator (URL).

To access an XForm within the AppWorks Platform application, use the following URL format:

```
<url>/home/<organization name>/<folder name>/.../<folder name>/<XForm name>.caf
```

For example, to access file1.caf in folder1 in organization acme, in the AppWorks Platform application, the URL will be /home/acme/folder1/file1.caf.

To open a standalone XForm, use the following URL format:

```
http://<server name:port number>/home/<organization name>/<folder name>/.../<folder name>/<XForm name>.caf
```

Where, port number refers to the port specified while installing AppWorks Platform. If no port number is specified, mention only the server name.

For example, to access a standalone form `OrderDetails.caf` from an external application `example.com`, in organization `acme`, the URL will be
`http://example.com/home/acme/OrderDetails.caf``http://example.com/cordys/OrderDetails.caf`.

You must specify the folder path in the same way in which it was structured during design time under the Application Models repository.

The organization attribute in the URL is optional. You can use this attribute to access XForms from organizations other than yours. To access an organization, specify the organization name or its distinguished name (DN).

Accessing translations

You can access translations of a single form or multiple forms.

Viewing translations of a single XForm

After an XForm is translated, you can view it in any of the translation languages specified in the XForm. You can use the query string to specify the language in which to view a single, translated XForm.

To view a translated XForm in the AppWorks Platform application:

- Specify the language code in the URL of the XForm as follows:

```
<url>/home/<organization name>/<folder name>/.../<folder name>/<XForm name>.caf?language=nl</url>
```

To view a translated, standalone XForm:

- Specify the URL of the XForm in the following format:

```
http://<server name:port number>/home/<organization name>/<folder name>/.../<folder name>/<XForm name>.caf[?language=<language code>]
```

Where, port number refers to the port specified at the time of installing AppWorks Platform. If no port number is specified, mention only the server name.

You must specify the folder path as it is displayed under the Application Models repository.

Both the organization and language attributes in the URL are optional. You must specify both the attributes to access a translated XForm from organizations other than yours.

To access a translated XForm in the current organization, use the language parameter only.

- If the language settings on your computer are set to a language for which translation is available, the XForm opens in the set language, by default.
- If the language set for your computer varies from the translation language that you specify in the URL, the XForm opens in the language specified in the URL.

Viewing translations of multiple XForms

You can modify the system parameters such that XForms automatically appear in a translated language. To do so, you need to set the language code as an extensible parameter of `.caf` in the system parameter collection as follows:

```
if (! system.parameters['.caf']) system.parameters['.caf'] = new Array();
system.parameters['.caf']['language'] = '<language code>';
```

This causes all XForms to appear in the specified translation language.

While translating multiple XForms, the XForm on which the language code is set is not translated; you must translate it as any other standalone XForm. A combination of both the above approaches enables you to view all XForms in the required language. Also, the language set on the query string (as for translating standalone XForms) takes precedence over the language set on the parameter (as for translating multiple XForms).

Viewing translations of XForms based on browser settings

In the previous sections, it is assumed that the translated language is specified in the URL. However, in some cases the URL may not contain information about the language in which the XForm must be displayed at runtime.

In these cases, the languages specified in the browser and translation languages specified in the XForm (saved in the `.mlm` file) are considered while displaying a translated XForm. The selection of the translation language is based on the criteria given below.

- The languages set in the browser are given higher priority if no language is specified in the URL of the XForm.
- If any of the browser languages exactly match the translation language specified in the XForm, the XForm is displayed in that language at runtime. For example, if the browser languages are specified in the order - en-AU, hi-IN, te-IN; and the translation languages specified in the XForm are hi-IN, sr-Cyrl, es-CL, the XForm will be displayed in hi-IN, which is the first exact match.
- If none of the browser languages exactly match the translation language specified in the XForm, then language codes mentioned in the browser are truncated and again used to find a match. This is done for consecutive browser languages until a match is found. For example, if the browser languages are specified in the order - en-AU, sr-Cyrl-BA, te-IN; and the translation languages specified in the XForm are sr-Cyrl, ta-IN, es-CL, there is no exact language match possible. In this case, the browser language en-AU will be truncated to en and searched for a match. Since, no match is available for en, the consecutive language sr-Cyrl-BA is truncated to sr-Cyrl and searched for a match. Since a match is available for sr-Cyrl, the XForm will be translated and displayed in sr-Cyrl.
- If no match is found in the browser languages and translation languages specified in the XForm, the XForm is displayed as is.

Accessing AppWorks Platform with a specific locale

To access the AppWorks Platform instance in a particular locale, you can set it in the Welcome page user preferences, add the language parameter with a value in the URL or set the browser specific language, so that the locale formats can reflect in all the AppWorks Platform UIs.

Format:

```
http://<server_name:port_number>/home/<organization_name>?language='<locale_name>'
```

A complete list of supported locales and corresponding codes in AppWorks Platform is listed in the [locales](#) topic.

Precedence in determining the locale

AppWorks Platform uses several approaches to determine the locale:

- From the URL (language Query String Parameter)
- From the message (for example, i18n SOAP header)
- From User Preferences
- From Browser Accept-Language header
- From AppWorks Platform default system Locale

Working with bi-directional support in AppWorks Platform user interfaces

Bi-directional text is the text that contains text in both the directions that is right-to-left (RTL) and left-to-right (LTR). Unlike several languages that are written from left to right (LTR), scripts such as Arabic and Hebrew and derived scripts such as Urdu, Persian, Yiddish, Jawi, and Ladino are written from right to left (RTL). If LTR is combined with RTL in the same document, each word in the text is written in a different direction, which is known as bi-directional text.

Base direction

Base direction refers to the overall direction of the text in a document or a block of text. The default direction for language in HTML is left to right. Depending upon the language, you can change the direction of the script in the `dir` attribute as:

- LTR: Left-to-right text direction.
- RTL: Right-to-left text direction.

The visual ordering and the logical ordering of the elements in the user interface also affect the directionality of a web document.

The order of precedence in which the language setting is applied to the user interfaces of AppWorks Platform is as follows:

1. The language parameter that is defined in the URL such as
<http://machineName/cordys/Forms/sample.caf?language=ja-JP>.
 See [Language Codes](#) for further details.
2. The language setting that is set in the Preferences through the Welcome page.
3. The setting of the language in the browser.

The following topics describe how to develop and implement RTL support in the User Interfaces of AppWorks Platform:

- [Developing User Interfaces](#)
- [Customizing HTML for RTL support](#)
- [Customizing CSS for RTL support](#)

Customizing CSS for RTL support

The platform and applications are supported through `/cordys/wcp/theme/default/style/template/rtl.css`. However, few form specific customizations and small scale or composite control customizations require CSS support in addition to the default `rtl.css`.

In such cases, consider the following options:

1. Create a `custom-rtl.css` and include it via script (assuming that `custom.css` is attached to the form).

Example:

```
if(application.rtlDirection) linkCSSFile(fileURL); //fileURL is the custom.css url
function linkCSSFile(fileUrl)
{
  <!-- Dynamically link the default rtl css file to the current application -->
  var fileref=document.createElement("link");
  fileref.setAttribute("rel", "stylesheet")
  fileref.setAttribute("type", "text/css")
  fileref.setAttribute("href", fileUrl);
  document.getElementsByTagName("head")[0].appendChild(fileref);
}
```

2. Add the directional attribute on the HTML tag, and directional class on body for RTL to contextualize the required selectors by usage of classnames.

```
function applicationReady()
{
  <!-- In onApplicationready event handler of the application add the directional
```

```
attribute on html tag, and directional class on body for RTL -->
if(application.rtlDirection)
{
  document.getElementsByTagName("html") [0].setAttribute("dir","rtl");
  document.getElementsByTagName("body") [0].className += " d-rtl";
}
}
```

3. Include only one CSS, but contextualize the required selectors by usage of classnames.

```
.foo{
  <!-- normal way, and the default properties for LTR orientation goes here -->
  border:1px solid #404040;
  border-width: 1px 2px 3px 4px;
  margin:1em 2em 3em 4em;
  padding:1ex 2ex 3ex 4ex;
  text-align:left;
  width:500px;
  left:10px;
  right:auto;
}
body.d-rtl .foo {
  <!-- the properties for the foo selector for RTL orientation goes here -->
  border-width: 1px 4px 3px 2px;
  margin:1em 4em 3em 2em;
  padding:1ex 4ex 3ex 2ex;
  text-align:right;
  left:auto;
  right:10px;
}
```

Customizing HTML for RTL support

To make a custom HTML page RTL compliant:

1. Set listeners for language detection.
2. Import translationUtil Library and detect language code. Check if RTL is required.
3. Import the default rtl.css.
4. Dynamically link the default rtl.css file to the application.
5. Add the directional attribute on the html tag and the directional class on body for RTL to contextualize the required selectors by use of classnames.

```
<!-- Set listeners for language detection -->

<!-- Import translationUtil Library and detect language code, and check whether RTL
```

```

is necessary -->
application.importType("translation.library.util.TranslationUtil");
application.rtlDirection = TranslationUtil.getLanguageDirection(application) ==
"RTL" ? true : false;
if(application.rtlDirection) addRTLFeatures();

<!-- import default rtl.css -->
function addRTLFeatures()
{
<!-- Dynamically link the default rtl css file to the current application -->
var fileref=document.createElement("link")
fileref.setAttribute("rel", "stylesheet")
fileref.setAttribute("type", "text/css")
fileref.setAttribute("href", "/cordys/wcp/theme/default/style/template/rtl.css");
document.getElementsByTagName("head") [0].appendChild(fileref);
}

<!-- Add the directional attribute on html tag, and directional class on body for
RTL to contextualize
the required selectors by usage of classnames -->
function applicationReady()
{
<!-- In onApplicationready event handler of the application add the directional
attribute
on html tag, and directional class on body for RTL -->
if(application.rtlDirection)
{
document.getElementsByTagName("html") [0].setAttribute("dir","rtl");
document.getElementsByTagName("body") [0].className += " d-rtl";
}
}
}

```

Developing user interfaces in AppWorks Platform

The design of the user interface differs for LTR and RTL languages. For LTR languages, the text on the upper-left of the page is given the highest priority; whereas in the RTL languages, the elements in the upper-right are scanned first. Depending upon the direction of the text, the HTML elements are rendered to load a page. The user interfaces and the layout of the reports that were originally designed for LTR languages are localized for RTL languages. The controls that support the RTL functionality are listed in [Supporting Controls](#).

For the user interfaces that are developed using AppWorks Platform, the RTL support is provided by default.

To ensure RTL orientation support:

1. Always use vertical or horizontal layouts for container elements such as form, tabs, group boxes, and splitters.
2. Always use vertical or horizontal positions for the controls.
3. Avoid free layout and free positioned elements.

4. Allow sufficient space for the labels:
 - Top labels are best used for text of varying sizes.
 - Side-right aligned labels in LTR mode (the text that aligns with the edge of the control) are best used when the horizontal form labels vary greatly in width.
 - When using horizontal labels, it is best to leave 30% additional space than usual for the labels that are more than 100 characters, 50% additional space for labels up to 100 characters, and 40% additional space for labels less than 10 characters.

Language codes

The following list contains codes for some of the languages that require RTL orientation.

Algeria	(ar-DZ)
Bahrain	(ar-BH)
Egypt	(ar-EG)
Iraq	(ar-IQ)
Israel	(he-IL)
Jordan	(ar-Jo)
Kuwait	(ar-KW)
Lebanon	(ar-LB)
Libya	(ar-LY)
Morocco	(ar-MA)
Oman	(ar-OM)
Qatar	(ar-QA)
Saudi Arabia	(ar-SA)
Syria	(syr-SY)
Tunisia	(ar-TN)
United Arab Emirates	(ar-AE)
Yemen	(ar-YE)

Supporting controls

The following controls and libraries support RTL:

- Input
- Output
- List

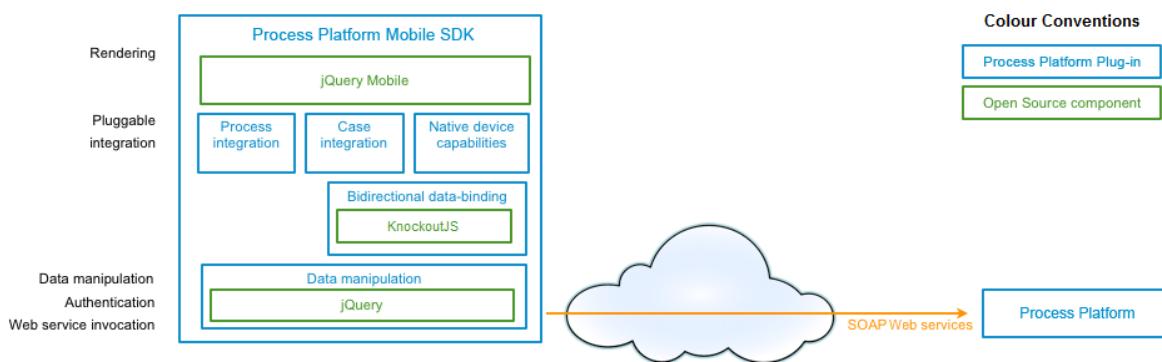
- Password
- Button
- Check
- Image
- Groupbox
- Textarea
- Autosuggest
- Radio
- Upload
- Slider
- Artifact Viewer
- Tabs
- Calendar
- Select
- Grid
- Toolbar, Titlebar, and Elementbar
- UFOs
- Statusbar

Working with HTML5 SDK

This section provides an [overview](#) of the HTML5 Software Development Kit (SDK) and explains how to [start](#) using it.

Overview

AppWorks Platform HTML5 SDK enables easy development of Web apps over AppWorks Platform for the mobile and tablet devices. The apps built using this SDK work as Web apps when launched directly from the browser.



AppWorks Platform HTML5 SDK consists of a set of plug-ins that are useful for building the mobile (Web Apps) Web applications. The plug-ins are used to interact with various AppWorks Platform artifacts such as Web services, Business Processes, Workflows, and Cases.

AppWorks Platform HTML5 SDK uses various familiar open source technologies to include:

- jQuery and jQuery Mobile: Optimized for browser independence
- KnockoutJS: Cross-browser JavaScript library designed to simplify dynamic JavaScript UIs

The following three approaches are available for developing the mobile apps:

- **Native apps** are developed using a specific programming language supported by the device. These apps run on the device and can access the native features of mobile. Native apps are fast, reliable, and have a look and feel closest to the ones available on the device. The disadvantage is that development is complex and cumbersome, as these require different skill sets and code-bases to support various mobile devices.
- **Web apps** are developed using Web technologies (HTML5, CSS, and JavaScript) and they run on a browser on the mobile device. These apps cannot access the native features of the mobile, are slower, and are not as reliable as a Native App. The advantage is that you can run the same code as all the devices that support the Web technologies. Only one skill-set and code base are required.
- **Hybrid apps** are similar to Native apps and run on the device, but they are developed using Web technologies (HTML5, CSS, and JavaScript). These apps run inside a native container. They leverage the browser engine of the device, not the browser, to render the HTML and process the JavaScript locally. They are capable of accessing the native features of the mobile. Hybrid Apps have an advantage over the Native Apps and the Web Apps.

AppWorks Platform provides a [AppWorks Platform HTML5 SDK](#) to enable easy development of Web Apps on AppWorks Platform for mobile and tablet devices. For more information on the layers used for mobile development, see [Framework](#).

Overview of the HTML5 SDK

AppWorks Platform HTML5 SDK enables easy development of Web apps over AppWorks Platform for the mobile and tablet devices. The apps built using this SDK will work as Web apps when launched directly from the browser.

AppWorks Platform HTML5 SDK consists of a set of plug-ins that are useful for building the mobile (Web Apps) or Web applications. These plug-ins are used to interact with various AppWorks Platform artifacts such as Web services, Business Processes, Workflows, and Cases

It uses various familiar open source technologies, such as the following:

- jQuery and jQuery Mobile: Optimized for browser independence
- KnockoutJS: Cross-browser JavaScript library designed to simplify dynamic JavaScript UIs

The following three different approaches are available for developing the mobile apps:

- Native apps are developed using a specific programming language supported by the device. These apps run on the device and can access the native features of the mobile. These are fast, reliable, and have a look and feel closest to the ones available on the device. The disadvantage is that the development is complex and cumbersome as these require different skill sets and code-bases to support different mobile devices.
- Web apps are developed using Web technologies (HTML5, CSS, and JavaScript) and they run on a browser in the mobile. These apps cannot access the native features of the mobile, are slower, and not as reliable as a Native App. The advantage is that you can run the same code as all the devices support the Web technologies used. One skill-set and one code base only is required.
- Hybrid apps are similar to Native apps and run on the device, but they are developed using Web technologies (HTML5, CSS, and JavaScript). These apps run inside a native container. They leverage the browser engine of the device, not the browser, to render the HTML and process the JavaScript locally. They are capable of accessing the native features of the mobile.

Hybrid Apps have an advantage over the Native Apps and the Web Apps. AppWorks Platform provides a [HTML5 SDK](#) to enable easy development of Web Apps on AppWorks Platform for the mobile and tablet devices. For more information on the layers used for the mobile development, refer to [Framework](#).

Framework

The following sections describe the reason for choosing HTML5, JQuery, jQuery Mobile, and Knockout.js as the layers for mobile development.

HTML5

Features:

- Works on all the smart phones and hence, a single code base
- Easy to develop and maintain
- Supports more capabilities such as Offline DB, Push Notifications, and Access to Native Features
- Availability of developer eco-system and support in the developer community
- Availability of libraries and toolkit

JQuery

JQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML.

Features:

- Easy to extend with plug-ins
- Acts as a base for other libraries such as jQuery-UI and jQuery Mobile
- Familiar to a lot of Web developers
- Less maintenance costs
- Simplifies DOM element selection, DOM traversing and manipulation, Event handling, CSS manipulation, Effects and animations, Ajax interactions, and other utility methods
- Compatible with all the widely used browsers
- Used by over 40% of the top million [most visited websites](#)

For more information about jQuery, see the [official jQuery website](#).

jQuery Mobile

New tools and frameworks are emerging with HTML5 and mobile Web development to provide more consistent and comprehensive HTML5 support across mobile browsers. A few of the popular HTML5 mobile Web frameworks are [jQuery Mobile](#) and [Sencha Touch](#).

Features:

- Built on jQuery code
- Compatible with all the major mobile platforms including iOS, Android, Blackberry, WebOS, Symbian, Windows Phone 7, and more; tablets; e-reader; and all the major desktop browsers
- Lightweight and limited dependencies to optimize the speed
- HTML5 markup-driven configuration
- Touch and mouse event support
- Unified UI widgets with powerful framework
- Ajax-powered navigation with animated page transitions
- Underlying code base automatically scales to any screen
- UI widgets that are platform independent

Sencha Touch

Features:

- Easy to set up
- Built-in transition effects
- Supports iOS, Android, and Blackberry
- Resolution independent
- Touch event support
- Ajax and JSONP support

jQuery Mobile and Sencha Touch - A comparison

- jQuery Mobile is widely used by the developer community compared to Sencha Touch.
- jQuery Mobile supports more number of mobile platforms compared to Sencha Touch.
- The documentation of Sencha Touch is not comprehensive, but jQuery Mobile documentation is good.
- jQuery mobile is lightweight compared to Sencha touch.
- Sencha Touch supports MVC (Model-View-Controller) style application design, whereas jQuery mobile is simply a load of markup and a load of jQuery script converting your HTML elements into touch-friendly interface components.
- jQuery Mobile framework is easy to integrate with other technologies.
- jQuery Mobile is free, whereas Sencha Touch is available free of charge for commercial and open source application development. However, embedding Sencha Touch in a Web application builder or Software Development Kit (SDK) requires a paid commercial OEM license agreement.

jQuery Mobile has many other advantages over Sencha Touch and therefore, jQuery Mobile is used in the AppWorks Platform HTML5 SDK. The users can also use Sencha Touch instead of jQuery Mobile, if required.

KnockoutJS

KnockoutJS is a cross-browser JavaScript library designed to simplify dynamic JavaScript UIs by applying the Model View View Model (MVVM) pattern. For more information, refer to the [KnockoutJS official Website](#).

Features:

- Declarative bindings in HTML, which reduces scripting
- Automatic UI refresh on model changes
- [Dependency tracking](#)
- HTML templating
- Small and lightweight
- Works with jQuery smoothly
- Highly customizable and pluggable

Other similar libraries include Ember.js and backbone.js, which follow MVC and Model-View-Presenter (MVP) respectively.

Benefits of KnockoutJS over these libraries are:

- Automatic observables and dependent observables. For example, compute fullName as firstName + lastName.
- Nested templates
- Easy to set up

Getting started with the SDK

You can get started with developing mobile pages using AppWorks Platform HTML5 SDK.

Even though the examples and descriptions provided in this topic are for creating the mobile pages, you can also use them to create other Web pages using the SDK.

Prerequisites

- AppWorks Platform must be installed.
- Northwind related Web services must be created for some demo pages.

Demo apps

You can refer to the sample HTML files in the demo folder shipped with the HTML5 SDK.

The files in the demo folder use the standard methods created using the [Northwind Database](#) with the Namespace `http://schemas.cordys.com/NW``http://schemas.cordys.com/NW`.

For more information, see [Generating Standard Web service Operations on a Database](#).

Creating an app

To create an app:

1. You can add mobile pages to any of your projects by adding the HTML documents to the web folder. For more information on creating a document, see [Creating a Document](#).
2. You can paste your mobile page code (HTML) in the new HTML document.
Only plain text editing is available.
3. To start with a template of the page, you can use any third party HTML designer, such as [Codiqa](#) for jQuery Mobile. You can use any of the examples in the htm15/demo folder or use the following code snippet, in which jsRender is used as the rendering library:

[expand source](#)

For more information on the HTML used in this example, see [jQuery Mobile](#).

You can use a binding library of your choice for the data binding. This SDK supports jsRender and knockoutJS out of the box.

You can also use other libraries such as knockoutJS, which is bundled with the AppWorks Platform HTML5 SDK, by replacing the **include for jsrender** with the following:

```
<script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"/>
```

The jsRender supports only unidirectional binding and knockoutJS is used in the SDK for its model implementation.

- For more examples on the usage of jsRender as a rendering library, see [Retrieving the Data](#).
- The examples provided in [Building Transactional Mobile Applications](#) and [Building Read-only Mobile Applications](#) use knockoutJS.

An app has been created.

For examples on using the HTML5 SDK, see [Sample Pages](#).

Using the App

Publish the Web page you developed using CWS (Collaborative Work Space). You can view the Web pages using a browser on your PC or mobile by accessing the following URL:

```
http://<your domain>/cordys/html5/demo/employees.htm
```

Adding the Styles

You can include the default stylesheet provided by [jQuery Mobile](#) or create custom themes using [ThemeRoller](#). It allows you to build a theme and download a custom CSS file, ready to be added into your project and inserted into your HTML page.

You can create a general theme for an organization and a specific theme for a project, but both of them must be included in the header of your pages before you specify the standard jQuery Mobile CSS.

```
<head>
  <title>jQuery Mobile page</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/cordys/com/myOrg/css/themes/my-organization-theme.css" />
  <link rel="stylesheet" href="/cordys/com/myOrg/my project/css/themes/my-project-theme.css" />
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
  <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"></script>
</head>
```

Sample pages

This topic describes the demo pages shipped with the AppWorks Platform HTML5 SDK and their limitations.

- All these are simple Web pages built using [jQuery](#) and [jQuery Mobile](#). The [jQuery](#) library is mandatory, but [jQuery Mobile](#) can be replaced by any other UI library.

- A few pages display the use of [KnockoutJS](#) for rendering and managing data. This can be replaced by any other rendering engine.
- Authentication is handled internally in the authentication plug-in.

Most of the demo pages are standalone pages and can be directly viewed in the browsers. However, few pages that are provided to showcase the use of the methods from workflow plug-in, process plug-in, and case plug-in have dependencies on the BPM and the case instances or Inbox tasks. As the dependent components are not shipped along with the SDK, these are just examples and may not render the expected data.

The demo pages are classified according to the following criteria:

- The task details or task approval pages
- The standalone Web pages
- The dependent Web pages
- The Web pages for reference

Task details or task approval pages

approvetask.htm

This Web page:

- Can be used by any business process that involves sending a document or an item for approval.
- Displays the business identifiers of the process and the options specified in the message map.
- Can be used for approval or rejection and for any other action by specifying different options. See [Working with the Generic Approval Page](#) for more information.
- Depends upon the inputs from the AppWorks Platform Process and cannot be viewed independently on the browser.
- Is added as an External User Interface in the process and does not require the mobile tagging.

The following image displays the Generic Approval Task page used in the Order Management Process:

Customer	Cordys
Product	Web Cams
Quantity	250
Cost	600
Discount	25

Comment

Approve Reject

Here, the Customer, Product, Cost, Quantity, and Discount are added as Business Identifiers; Approve and Reject are added as the custom Actions.

customapproval.htm

This Web page:

- Is added as a reference for the user to create their own task details page or task approval page.
- Depends upon the inputs from the AppWorks Platform Process and cannot be viewed independently on the browser.
- Takes in OrderID as the input from the Process and displays the Order Details. The user can select the Shipment Details and confirm Shipment.
- Is added as an External User Interface in the process and does not require the mobile tagging.

For more information, see [Building a Custom Approval Page](#).

The following image displays the Custom Task Details page:

The screenshot shows a mobile application interface with the following sections:

- Order Details** section:
 - OrderID: 10248
 - Order Date: 1996-07-04T00:00:00.0
 - Required Date: 1996-08-01T00:00:00.0
 - Freight: 32.38
 - Shipping Address: Vins et alcools Chevalier, 59 rue de l'Abbaye, Reims - 51100
- Select Shipment Details** section:
 - Shipping Date: Month Day Year dropdowns
 - Shipping Method: Select dropdown
- Buttons**: Confirm Shipment

Standalone web pages

Google Visualization and Fusion Charts are supported only by the Android browser in Android 3.2 or later. Other browsers for the Android mobiles, such as Mozilla Firefox and Google Chrome support the charts irrespective of the Android version. This is applicable to the following Web pages:

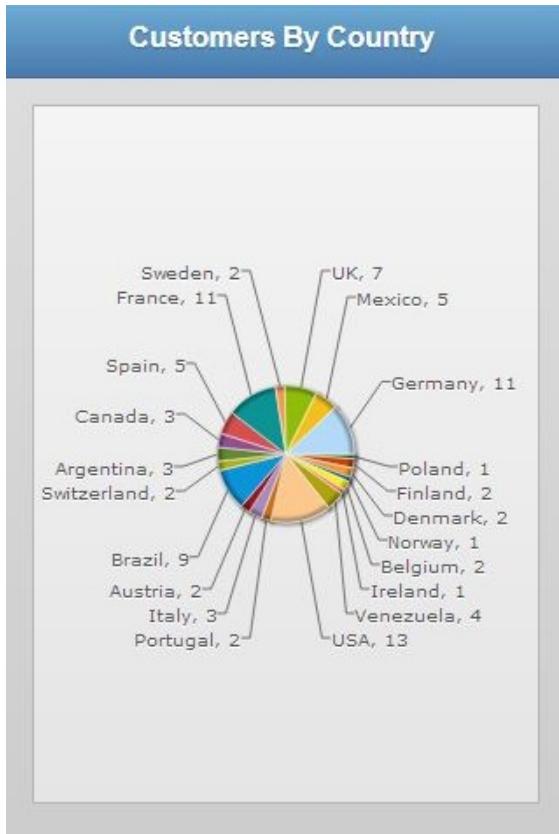
- [customersbycountry.htm](#)
- [productsinstock.htm](#)

The Web pages mentioned in this section can be viewed on the browser using the following URL:

<http://<server-name>/home/<organization-name>/html5/demo/<demo-page-name>.htm>

[customersbycountry.htm](#)

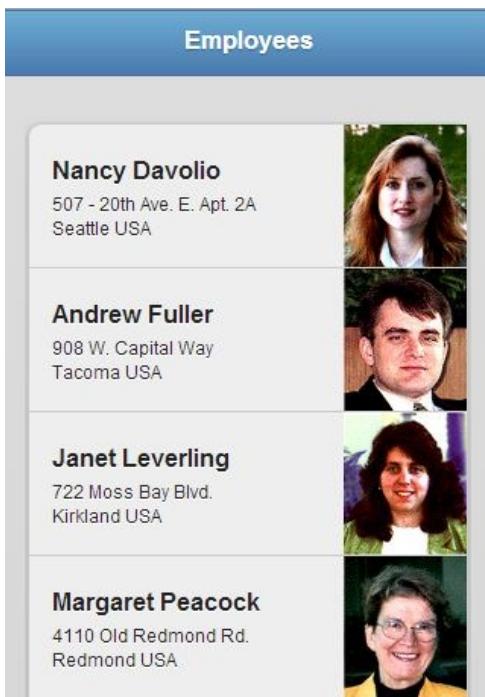
This demo page describes an example to display the graphs using the AppWorks Platform HTML5 SDK by integrating it with the Fusion Charts. In this example, all the customers of the Northwind Table are retrieved and a pie chart of the number of customers by the country is displayed as shown in the following image:



You can click any country to view the customers from that country in a list. The `$cordys.model` plug-in along with KnockoutJS are used to bind the data and the graph. For more information on this Web page, see Example 1 in [Displaying Graphs on a Mobile Device](#).

employees.htm

In this demo page, all the employees from the Northwind Database are rendered using `jsRender`. Alternatively, KnockoutJS can also be used for rendering the data. For more information on using KnockoutJS, see the `employeeslist.htm` section. The `cordys.ajax` plug-in is invoked to retrieve the employee information by specifying the method with its namespace and parameters. For more information on this Web page, see [Retrieving the Data](#). The list of employees is displayed as shown in the following image:



[employeeslist.htm](#)

This demo page lists all the employees from the Northwind database and uses KnockoutJS to render the data. For more information on using jsRender, see the `employees.htm` section above. The method `read()` from the model plug-in is used to retrieve the employee information by specifying the method with its namespace and parameters. For more information on this Web page, see Example 1 in [Building Read-only Mobile Applications](#).

[employeeslistwithnavigation.htm](#)

This demo page describes how to implement pagination over the Web services implementing the Cordys cursor mechanism using the model plug-in. This is based on the example in [Building Read-only Mobile Applications](#). The demo page is built on the Employees table from the Northwind Database. For more information on this Web page, see [Implementing Pagination](#).

The following image displays a list of four employees per page in the App:

The screenshot shows a web page titled "Employees". It displays four employee records in a vertical list:

- Nancy Davolio**
507 - 20th Ave. E.
Apt. 2A
Seattle USA
- Andrew Fuller**
908 W. Capital Way
Tacoma USA
- Janet Leverling**
722 Moss Bay Blvd.
Kirkland USA
- Margaret Peacock**
4110 Old Redmond Rd.
Redmond USA

[employeeswithdetails.htm](#)

This Web page is added as an enhancement to the page `employeeslist.htm`. You can tap the required Employee in the list to view the corresponding details as shown in the following image:

The screenshot shows a web page titled "Employee Details" with a back button. It displays the following information for employee number 1:

1
Ms. Nancy Davolio
Sales Representative

Form fields for Nancy Davolio:

- Title of Courtesy: **Ms.** (selected)
- First Name: **Nancy**
- Last Name: **Davolio**
- Title: **Sales Representative**
- Phone: **(206) 555-9857**

For more information on this Web page, see Example 2 in [Building Read-only Mobile Applications](#).

employeeswithlocation.htm

This Web page is added as an example to show the integration with Google Maps using the AppWorks Platform HTML5 SDK. It lists all the employees from the Northwind database and on selecting a particular employee the details page is displayed with the address of the employee on Google Map. The \$.cordys.model plug-in and KnockoutJS are used to bind the data. For more information on this Web page, see [Integrating with Google Maps using HTML5 SDK](#). The following image displays an Employee and their Address and the corresponding Location is shown using Google Maps.

The screenshot shows a mobile-style web page titled "Employee Location". At the top, there is a back button and the title. Below the title, there is a profile picture of Ms. Nancy Davolio, her name, and her title "Sales Representative". A section labeled "Address" contains the address "507 - 20th Ave. E Apt. 2A, Seattle, USA". Below this, a section labeled "Location" displays a map of a neighborhood. The map shows several streets including 11th St, St Joseph Church, E Roy St, Alona St, Holy Names Academy, 21st Ave E, 22nd Ave E, 23rd Ave, 24th Ave E, and Prentis I. Frazier Park. A red marker indicates the location of the address listed above. There are also icons for a magnifying glass and a plus/minus sign for zooming.

employeeswithtranslation.htm

This Web page is added as an example to display the features of translation plug-in that is used to translate messages and labels in the HTML pages. To view the page translated in Dutch directly from the browser, you can suffix ?language=nl-NL to the URL as shown in the following example:

`http://<server-name>/home/<organization-name>/html5/demo/employeeswithtranslation.htm?language=nl-NL`

For more information on this Web page, see [Translating a Page into a Specific Language](#). The translatable content will be translated and displayed as shown in the following image:

The screenshot shows a mobile application interface titled "Medewerker Details". At the top left is a "Terug" button with a back arrow icon. The title "Medewerker Details" is centered above a list of employee information. The first item in the list is "1" followed by the name "Ms. Nancy Davolio" and the title "Sales Representative". Below this, there is a section for "Aanspreektitel" with a dropdown menu showing "Mejuffrouw" with a checked checkmark. The form fields for "Voornaam" (Nancy), "Achternaam" (Davolio), "Titel" (Sales Representative), and "Telefoon" ((206) 555-9857) are displayed below. To the left of each field is its label.

[employeeswithupdate.htm](#)

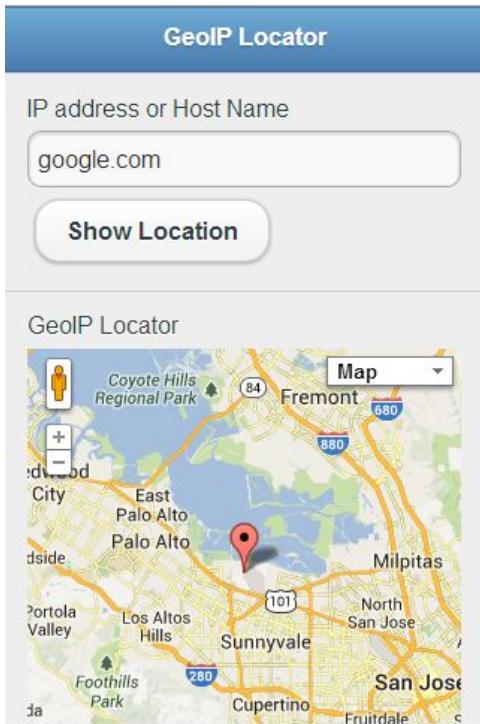
This example describes the procedure to build transactional mobile applications over the AppWorks Platform Web services using the cordys.model plug-in. It uses the KnockoutJS library to handle the updates. This page uses methods read() and update() from the AppWorks Platform Model plug-in. For more information on how to use AppWorks Platform Model Plug-in for CRUD operations, see [Building Transactional Mobile Applications](#).

[jqueryvalidation.htm](#)

This example describes the procedure to make a field mandatory such as input box or select box, and specify the validation rule such as minimum or maximum range. It is built on the Employees table from the Northwind Database and displays the employees form with the field validations. If all the mandatory fields are provided and the specified rules are met for the field, the employee data will be saved in the Employees table. For more information on this Web page, see [Validating a Page Using jQuery Validation Plug-in](#).

[restsample.htm](#)

This page describes retrieving the data from a REST Web service using cordys.ajax plug-in. The process is explained using an example Geo IP Locator. This example uses the [REST Web service](#) for retrieving the details of the provided IP address or host name. It uses Google Map for displaying the location of the IP address. You can provide the IP address or Host Name to view its location in the Google Map.



productsinstock.htm

This demo page describes the procedure to display the graphs using the AppWorks Platform HTML5 SDK by integrating it with Google Visualization API. It retrieves all the products of the Northwind Table and displays a graph showing the number of available units of each product in a bar graph. The \$.cordys.model plug-in and KnockoutJS are used to bind the data and the graph. For more information on this Web page, see Example 2 in [Displaying Graphs on a Mobile Device](#). The following image displays the products and their respective units in stock:



orderslistwithswipe.htm

This Web page is added as an example to display pagination with endless list. In mobile interfaces, pagination is usually done using an endless list instead of the pagination buttons. In an endless list, the specified number of records are retrieved first and when the scroll reaches the end of the page, the next set of records are retrieved and the result is appended to the existing list. This behavior continues until all the records are retrieved. This example retrieves all the Orders from the Northwind Database and the orders are listed using the endless list. For more information on this Web page, see Example 2 - Pagination With Endless List in [Implementing Pagination](#).

Web pages for reference

Important: These pages will not render as expected as they have dependencies on the instances of other AppWorks Platform components such as, Process and Inbox, which are not included as a part of the SDK.

The following Web pages are shipped with the SDK for the reference and they describe the implementation of few methods from workflow plug-in and process plug-in.

Report damage model

These Web pages were designed as a part of a sample scenario Report Damage, where a photo of the accident is captured and uploaded for reporting the damage.

attachphoto.htm

This Web page is used for attaching the photo to the model taken from the device camera and uploading it to the Process. It lists the use cases for `getTaskDetails()` and `addAttachment()` methods from the plug-ins workflow plug-in and `getBusinessIdentifiers()` from process plug-in.

reportdamage.htm

This Web page is added as an example to display the use of `navigator.geolocation.getCurrentPosition()` to get the current location of the user.

Script loader using require.js

Some third party libraries such as jQuery, Knockout, and so on, are shipped with AppWorks Platform. When you develop an application using the AppWorks Platform SDK, some APIs from the third party libraries are directly consumed. When these APIs are changed or deprecated in the next version of the library, for example, if you upgrade to the next version of OpenText AppWorks Platform and the third party library's version has changed, the application functionality can break.

Requirement

You must be able to load the third party script files for every release without worrying about backward compatibility. Script loader `require.js` is implemented in the HTML5 SDK to support backward compatibility.

Using script loader

Prerequisite: You must have basic knowledge of using `require.js`.

A script loader includes a single script tag in the HTML page instead of including script tags in every JavaScript file. A simple script loader tag is as follows:

```
<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/latest.js"></script>
```

A script tag loads `require.js` specified in the `src` attribute. AppWorks Platform provides the `latest.js` file, which is a default bundle pointing to the latest third party libraries. The script tag then checks the `data-main` attribute and loads the `latest.js` file, which is shown below.

```
require({
  paths: {
    'jquery': '/cordys/thirdparty/jquery/jquery',
    'html5sdk': '/cordys/html5/cordys.html5sdk',
```

```

'html5sdk.util': '/cordys/html5/cordys.html5sdk.util',
'cordys.mobile' : '/cordys/html5/plugins/cordys.mobile',
'jquery.mobile': '/cordys/thirdparty/jquery/jquery.mobile.min',
'jsrender': '/cordys/thirdparty/jsrender/jsrender',
'knockout.src': '/cordys/thirdparty/knockout/knockout',
'knockout.mapping.src': '/cordys/thirdparty/knockout/knockout.mapping'
},
shim: {
'jquery' :{},
'cordys.mobile' : {deps:['jquery']},
'html5sdk' : {deps:['jquery']},
'html5sdk.util' : {deps:['html5sdk','knockout','knockout.mapping']},
'jquery.mobile' : {deps:['jquery']},
'jsrender' : {deps:['jquery']},
'knockout.src' : {deps:['jquery']},
'knockout.mapping.src' : {deps:['knockout', 'jquery']}
}
},['html5sdk']);

```

The `latest.js` file contains the paths for the third party libraries that are provided by AppWorks Platform. The dependency between the libraries is maintained by the shim configuration. On loading the `latest.js` file, the modules (each script file is a module) and their corresponding paths are loaded in `require.js`.

The script loader `require.js` checks the `require` function of the `latest.js` file to check the dependency for the second argument '`html5sdk`'. It loads `jquery` before `html5sdk` because in the shim configuration `html5sdk` depends on `jquery`.

Loading additional script files

In the above mentioned package only `html5sdk` is loaded by default. The remaining files are loaded using the `data-include` attribute. All the modules that you want to load must be specified as comma-separated values as shown below.

```

<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/latest.js" data-
include="jquery.mobile,knockout,knockout.mapping"></script>

```

On Load event

We must call a callback function when all the scripts are loaded, which is provided by the `data-ready` attribute, because `require.js` loads the files in an asynchronous way. Any code that must be executed after loading the scripts must be wrapped inside this function. This function is a replacement for the `jquery onload` and normal document `onload` function. The function name to be called must be set as the value for the `data-ready` attribute as shown below.

```
<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/latest.js" data-
include="jquery.mobile,knockout,knockout.mapping"
data-ready="requireLoad"></script>

<script type="text/javascript">

function requireLoad(args) {

-- Code that needs to be executed after load of script files--
}
```

Additional configuration

If you want to load script files that are not mentioned in the included package, you must use the data-include-config attribute. The object name that defines the configuration must be set as value for the attribute as shown below.

```
<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/latest.js" data-
include="jquery.mobile,jquery.validate,knockout,knockout.mapping"
data-ready="requireLoad" data-include-config="dataConfig"></script>

<script type="text/javascript">

var dataConfig = {
paths: {
'jquery.validate' :
'http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.1/jquery.validate'
},
shim: {
'jquery.validate' :{deps:['jquery']}
}
}

function requireLoad(args) {

-- Code that needs to be executed after load of script files--
}
```

The object mentioned in the data-include-config attribute is added to the existing configuraton object mentioned in the latest.js file and it can be loaded using the data-include attribute.

Modules provided by platform

The following modules are provided by AppWorks Platform packages:

- jquery
- html5sdk

- html5sdk.util
- cordys.mobile
- jquery.mobile
- jsrender
- knockout
- knockout.mapping

The module names are case-sensitive and should be carefully mentioned in the data-include attribute. The above mentioned module names may not match with the name in the packages, for example, knockout is mentioned as knockout.src in latest.js for implementation purpose. In the application, only the module names mentioned above must be used, which will be mapped to the appropriate name by the SDK.

- data-include, data-include-config and data-ready are html5sdk custom attributes. They are not like data-main, which is from require.js.
- Move all the code that depends on script files to the function mentioned in the data-ready attribute.
- If your application involves model binding, include knockout and knockout.mapping.
- Use module name mentioned in the documentation. Do not use the names mentioned in the package.
- Avoid loading script files directly through script tags. Define your custom config through data-include-config attribute and load it properly.
- Example pages using script loader have been given under Script_loader folder of html5 demo.

Version support

If the current version of AppWorks Platform is 10.8, config10.6.js will point to the third party library versions that are supported in AppWorks Platform 10.6 and similarly config10.7.js will point to the libraries supported in 10.7.

Case 1: If you always want to use the latest version, you must include latest.js in your application.

```
<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/latest.js" data-
include="jquery.mobile,knockout,knockout.mapping"></script>
```

Case 2: If you are using the older version of AppWorks Platform, you must use config<OPENTEXT_CORDYS_VERSION>.js, for example, config10.6.js.

```
<script src="/cordys/thirdparty/require/require.js" data-
main="/cordys/html5/config10.6.js" data-
include="jquery.mobile,knockout,knockout.mapping"></script>
```

Organization level styling and BASE relative URLs

AppWorks Platform Web front end is multi-tenant aware; that is, the Web content can be deployed in an organization that allows, for example, a different style for that organization. Each tenant is provided with its own corporate styling even though the same application is used by all the tenants. With multi-tenant aware, the developers now can develop, publish, and debug their own version of, for example, a JavaScript without other developers undoing their changes while publishing their content.

Organization-specific web content

The Web content deployed to a specific Organization is stored in a different folder from the Web content deployed to the Shared level. In the earlier versions of AppWorks Platform, all the Web content was deployed to a folder <cordys-Instance>/Web, which is changed to a folder named <cordys-Instance>/webroot. The webroot folder has a Shared folder and an Organization folder. The Organization folder contains a folder per organization along with its name. When the Web content is deployed or published to the Organization level, it is copied to the organization-specific sub-folder.

The Web server is made multi-tenant aware so that when it serves the Web content, the Organization level is verified for the content first and the Shared level later. The Web content in the organization folder is served whenever it is available. Therefore, when an image in a certain URL is available in both the Shared and Organization levels, the Organization level image is served to the front end.

From absolute to BASE relative URLs

In the earlier version, AppWorks Platform used absolute URLs, where the URL is the complete folder structure from the hostname to the actual file name. The following is the example of a script and a style sheet loaded through an absolute URL:

```
<head> <script src="/cordys/wcp/application.js"></script> <link rel="stylesheet" href="/cordys/wcp/style/default.css"></link>
```

BASE URL

To support a multi-tenant Web content, the organization name is made a part of the URL; for example, <https://bopserver.cordys.com/home/acme>, where acme is the organization name. The organization is a part of the URL, which means that the URLs cannot be used as absolute; otherwise, the URL will be as specified above with the /home/acme part in it. This is solved by introducing the use of base relative URLs. The base relative URLs are the URLs that are relative to the BASE URL. The BASE URL of a AppWorks Platform page is the URL including the organization name; for example, <https://bopserver.cordys.com/home/acme> is referred as the < organization URL>. The URLs used in forms, HTML pages, and Javascript are relative to this organization name including the BASE URL. This behavior is available by

using the BASE element. Refer to [BASE element](#) for more information. It is an HTML tag that can be added to the HEAD tag as specified in the following example from W3C:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <HTML> <HEAD> <TITLE>Our Products</TITLE>
<BASE href="http://www.aviary.com/products/intro.html"> </HEAD>
```

BASE element and application.js

The BASE element is set when a form is generated in the run time. Thereon, all the URLs are relative to the new BASE URL. For the HTML files, the BASE element is not added as these are not generated. It is set by the application.js script that is loaded in every AppWorks Platform HTML application; that is, until the application.js script is loaded, the URLs are handled as relative URLs. This has implications for the URL used to load the application.js script. The script must have a URL relative to the page loading it. The following example displays the usage of relative URL in the application.js from a page with the URL:

<organizationURL>/com/cordys/mypage.htm:

```
<head> <script src="../../wcp/application.js"></script>
```

Referring to other Web content

When using the XForms designer, the management of the above URL is handled by the XForms engine. Also, while adding a Javascript, image, or stylesheet to the form, the developer need not worry about the BASE relative URL.

In case the developer uses Javascript to refer to an image or another form, then the above action is needed as the scripts are not changed by the XForms engine. When developing an application in the pure HTML, the developer must handle the URLs. The following rules determine which URL must be used:

- URLs from Javascript are required to be relative to the BASE URL; they can be URLs to Javascript, images, other forms, stylesheets, and so on. There is no difference between the Javascript included in a Form or HTML and the Javascript included in a separate JS file. The BASE URL is the path including the organization name, for example, <https://bopserver.cordys.com/home/acme>, called the < organization URL>.
- URLs within an HTML element are required to be relative to the BASE URL. For example, the URL in the src attribute of an image element is required to be relative to the < organization URL>.
- URLs within a Cascading Style Sheets (CSS) are required to be handled differently. The CSS does not support the BASE relative mechanism. Therefore, the URLs within a CSS must be relative to the CSS file itself.

Examples

The following sections provide the examples for the URLs referenced from a page accessed through the URL: < organization URL >/com/cordys/mypage.htm

Including a JS file

The following example provides the URL to a JS file in the folder <Web Content Folder >/com/cordys/js/utils.js:

```
<script src="com/cordys/js/utils.js" type="text/javascript"/>
```

The URL provided is relative to the BASE URL.

Linking a CSS file

The following example provides the URL to a CSS file in the folder <Web Content Folder >/com/cordys/css/style.css:

```
<link rel="stylesheet" href="com/cordys/css/style.css" >
```

Note: The URL is relative to the BASE URL.

Reference to an image file

The following example provides the URL to an image in the folder <Web Content Folder >/com/cordys/img/logo.png:

```

```

The URL is relative to the BASE URL.

Special care with URLs from CSS

As mentioned earlier, the CSS does not support the base relative URLs. The URLs are formed to be relative to the CSS file. The following example contains the usage of a URL with a CSS file accessed through the <organization URL>/com/cordys/css/style.css

Reference to an image file

The following example provides the URL to an image in the folder <Web Content Folder >/com/cordys/img/logo.png:

```
background-image: url( "../img/logo.png" )
```

The URL is relative to the CSS itself.

Reference to another CSS file

The following example provides the URL to import another CSS file in the folder <Web Content Folder >/com/cordys/css/acme/customer.css

```
@import url( "acme/logo.png" )
```

The URL is relative to the CSS itself.

Upgrading an existing Web content

The Web content with multitenant support provides new capabilities. Also, the existing content still works with the absolute path. To make an existing Web content multitenant aware, it must be updated. See [Correcting Absolute and Relative Paths in Web Artifacts](#) on how to update the existing Web content.

Managing AppWorks Platform using web task manager

The Web Task Manager enables you to manage the applications and libraries that run in the AppWorks Platform. Each instance of an application or library running in the AppWorks Platform application is known as a Web task.

You can start a Web task, stop a running Web task, or create a new Web Task using the Web Task Manager window.

Before you begin:

Ensure that you have the Developer role to work with the Web Task Manager.

To use the web task manager:

1. In the My Applications App Palette, click  (Web Task Manager). The Web Task Manager window opens, displaying a table that contains details about the various Web tasks running in the AppWorks Platform application. The details that display for each Web task are as follows.

Id	Displays the ID of the application under the column.
State	Displays the status of the application (whether loaded, loading, or unloaded).
URL	Displays the URL of the application under the column.
Instances	Displays the number of instances of an application, running in the AppWorks Platform.

Visible	Displays a check box under the column to specify whether the application is visible.
Debug	Displays a check box under the column to specify whether to run the SOAP Debugger for the Web task. If selected, the SOAP Debugger window displays the SOAP messages being sent and received for the Web task.
Spy	Displays a check box under the column to specify whether to run the Spy window for the Web Task. If selected, the Spy window displays the transactions that take place in the system.
Current Style	Displays the style applied to the Web Task

2. To work with existing Web tasks, select a Web task and make appropriate modifications.
 - Click  (Start) to start the selected Web task.
 - Click  (Stop) to stop the selected Web task.
 - Click  (Refresh) to refresh the list of Web tasks.
3. To create a new Web Task, click  (New) in the toolbar.
An empty row appears in the table.
4. Enter the URL of the application in the URL column of the empty row.
Details related to the application display in the other columns of the table.

This adds a Web task to the Web Task Manager window, which can be managed using the options available in the toolbar.

Overview of task

AppWorks Platform has many predefined roles such as System Administrator, Organizational Administrator, and Developer. These roles are associated with a prescribed set of rights or activities, which a role is authorized to perform. An application contains many activities to perform, but a role need not be allowed to perform all the activities or access all the features. This calls for simpler roles with lesser number of activities or fine-grained tasks or applications. Based on the authorization or access level, the application will enable or disable activities for that user. For example, a developer must be allowed only to view the Application details and not to install an Application.

AppWorks Platform provides a mechanism to define a set of activities for a given user depending on the role(s) associated with that user. This facilitates the user to work only on those predefined activities or features and disable other rights.

Using Task, you can be aware of the following aspects in an application:

- Who is accessing the application
- What is the user's role(s)

- Which task(s) the user can perform

Typically, multiple users with multiple roles and authorization levels use AppWorks Platform with different goals. AppWorks Platform allows you to define custom roles depending on the user requirement and assign task(s) or activities to that role in the deployed environment. Using task, you can manage and maintain the tasks depending on the roles or user.

Task

A Task is an activity that can be performed independently. See [Task definition](#) for information on the associated properties. See [example](#) for a better understanding.

Configured task

A configured tasks is a group of task parts, which can be enabled or disabled for user access. You can configure task parts while assigning tasks to users or roles. See [Configured Task definition](#) for information on the associated properties.

Tasks and Configured Tasks are stored in a Task Repository that is a part of AppWorks Platform Repository. Task framework provides Web services and Java API to retrieve task Information.

A task is automatically generated when you [create a user interface](#).

Chapter 18

Working with application connectors

An Application Connector provides a SOAP interface to an application. Application Connectors act as translators that convert SOAP messages, which are XML messages in AppWorks Platform, to appropriate application/service APIs. The Application Connectors also convert information received from the APIs to incoming SOAP messages. Thus, in AppWorks Platform, an Application Connector is used to communicate with a back end. A connector is loaded in a Service Container.

A Service Container can connect to multiple applications or backends through multiple application connectors.

The generic life cycle of creating and deploying an Application Connector is as follows:

1. Creation: AppWorks Platform provides a set of default Application Connectors that provide connectivity to various back ends. Apart from these, the developer can build his own custom connector to connect to any specific back end (that is not supported by the available connectors).
An application connector can be used in a development environment also, after it is defined or created.
2. Transit: Once an application connector is created or defined in a development environment, it is packaged into an ISV along with the other essential data, crucial for running applications on AppWorks Platform. Such data can include BPMs, Method sets, XForms, Schemas etc.
The completely built Application is dispatched and loaded in a deployment environment. Upon loading, the contents within it are transited and mapped to their respective system resources.
3. Deployment: All those connectors that are loaded along with the Application are listed during the Service Group creation. You can select the necessary connector to establish a connection between the Service Container and the back-end application.

Application connectors are placed at the following location:

The Welcome page > My Applications > XMLStore Explorer () > Collection > Cordys > WCP > Application Connector.

Type of connectors

Application connectors are broadly classified as the following types

- Built-in connectors - The in-house connectors that are available by default with in AppWorks Platform. Example: XMLStore Connector, Monitor Connector, LDAP Connector.
- Customized connectors - The tailor-made connectors that are created and adapted as per the user requirement. These connectors are **created and configured** before they are deployed in runtime.

Creating and configuring a custom connector

AppWorks Platform1 provides a set of default Application Connectors that process the business logic and provide connectivity to various back ends. Apart from these, you can build your own custom connector to connect to any specific back end. When the default set of connectors do not serve the purpose for your AppWorks Platform1 based custom applications, you can choose to create a custom connector.

After creating a custom connector, it is essential to configure it to the AppWorks Platform1 framework to enable connectivity. AppWorks Platform1 provides a feature to provide a user-friendly interface to configure the custom connectors. After configuration, you can find the schema of this interface at Welcome page > My Applications > XMLStore Explorer () > Collection > Cordys > WCP > Application Connector.

Before you begin:

- Install AppWorks Platform1.
- You must have the role Developer.

To create and configure a custom connector:

1. Create two java files by extending ApplicationConnector class and implementing ApplicationTransaction class. Ensure to override the following Web service operations:

Java Class	Web service operations to be overridden
class com.eibus.soap.Application Connector	<ul style="list-style-type: none"> ■ open() - This is invoked by the service container after establishing the connection to the AppWorks Platform1 ESB. ■ close() - This is invoked by the service container after closing the connection to the AppWorks Platform1 ESB. ■ reset() - This is invoked by the service container to clear the cache. ■ createTransaction() - This is invoked by the service container to indicate the arrival of a new message and creation of a SOAPTransaction for it. ■ commit() - This is invoked by the SOAPTransaction, when the entire SOAPTransaction succeeds and can
com.eibus.soap.Application Transaction	

Java Class	Web service operations to be overridden
	<p>be committed.</p> <ul style="list-style-type: none"> ■ abort() - This is invoked by the SOAPTransaction, when one of the ApplicationTransactions has returned false during the process() method. ■ canProcess() - This method returnsTrue, if the XML mapper can process Web service operations defined of the specified type such as SQL, Java, LDAP and so on. ■ process() - This is invoked to process the relevant body blocks within this transaction.

For more information about these Web service operations, refer to Java SDK documentation.

2. Generate a JAR file for the above indicated Java files.
This JAR file will be used while registering the custom connector.
3. Create a HTML file, include /cordys/wcp/admin/behavior/applicationconnector.js and add the method initialize() as an event-handler to the application's onapplicationready event.
4. [Creating a project](#)In this HTML file, create the implementation for the following Web service operations: createConnectorConfiguration() and fillInPropertyScreen().
5. [Create a custom Web service](#) with the necessary Web service operations.
6. Configure your connector and associate with the framework using the HTML page, implementation class (java class) and publish the connector. Follow the below steps to configure the connector:
 - a. [Create a project](#) in CWS.
 - b. [Select a starting point](#) and click  (Application Connector) to open the Application Connector Editor.
The Untitled Application Connector - Application Connector window appears.
 - c. Provide Configuration information in the <Application Connector Name> - Application Connector window.

Ensure to provide the Implementation Class and Property Page details during configuration. If not, the custom connector will not be available for deployment.

- Click .
- The Save Document dialog box opens.
- Provide the Name and Description, and click  (Lookup) to browse and select the location to save the connector. The folder path where the connector will be created appears in the Save in Folder field.
- Close the Application Connector window.

The custom Application Connector is created and configured with the provided name and is stored at the selected location.

After you complete this task:

- [Publish](#) the custom Application Connector.
- Create a Service Group and during the process ensure to select the custom connector and custom Web service interface that you have created in the previous steps. The configuration page for the Application Connector would be taken from the Property Page URL mentioned while configuring the connector.
- [Deploy the custom Web service interface in the XForms](#).

Application connectors

The application connectors enable integration of various applications and technologies with AppWorks Platform through a set of open standards. These connectors act as a means to establish technical interoperability between the components of AppWorks Platform and other components such as Java, C++, Visual Basic, .Net and so on.

The application connectors of AppWorks Platform support reuse of the program logic and provide transparency of location in a distributed environment. The detailed information related to the application connectors of AppWorks Platform follows.

Auditing

Auditing assists you to record the audit information such as when the Artifact is configured in SOAP Mode. Also, it helps you to retrieve the audit information whenever you wish to view.

BAM connector

The BAM Connector filters business data, monitors the critical events of business and non-process data for further analysis. Also, it performs the administrative tasks (publish, unpublish, execution) of AppWorks Platform BAM artifacts like Process Monitoring Object, Business Measure, KPI and Business Event Response.

Business Process Management

The Business Process Management connector provides runtime environment for deployment, execution, debugging, administration, monitoring of business processes and archival of historical process data.

CoBOC service

CoBOC Service Connector provides a robust mechanism for object state management, configurable business rules, and real-time state monitoring.

AppWorks Platform (<instance name>) service

AppWorks Platform (<instance name>) Service may be a service running on a Windows machine, or a daemon running on a Linux machine. It is a program responsible for the lifecycle (start, stop, reset, and re-start) of the Service Containers that are configured on the machine.

Other - Custom application connector

AppWorks Platform enables building custom application connectors to connect to any specific back-end that is not supported by the default connectors.

Collaborative workspace connector

Collaborative Workspace (CWS) connector handles all the development activity in a workspace, including creation of solutions, projects, and documents. It acts as a bridge between the CWS repository and the file system and keeps them in sync. It is also responsible for generating run-time application content by building, publishing, and packaging the application.

A CWS Service is available with the System organization and comes along with the AppWorks Platform installation. You need not create another Collaborative Workspace Service. However, to address fail-over situations, you may create another CWS Service within the CWS Service Group (add a new CWS Service to the existing CWS Service Group) in the System organization. When you do this, ensure not to modify the default settings of the CWS Service Group.

Data transformation service

Data Transformation Service Connector provides a translation service to other components and applications and is responsible for implementing the XML transformation.

Document store

AppWorks Platform can be configured to work with any content repository. After the configuration is done, AppWorks Platform behaves as the client for the content repository for storing documents. For more information on Document Store, refer Document Store. Data Transformation Service Connector provides a translation service to other components and applications and is responsible for implementing the XML transformation.

E-mail connector

E-mail connector enables sending and receiving mails through the SMTP or POP3 or IMAP protocol. You can receive a single mail or multiple mails. You can also send and delete mails. For information on E-mail connector APIs, refer to E-mail Connector APIs.

Event service

Event Service enables the subscription of events and the subscribed events are then published to the subscriber. For more information on Event Service, refer to Event Service.

File transfer protocol (FTP) connector

File Transfer Protocol (FTP) connector is used to upload and download files from an FTP server to a system on which the FTP Service Container is running.

The FTP connector APIs handle several types of FTP commands. For information on this, refer to FTP Connector APIs.

LDAP connector

LDAP connector reads and writes the User details, Roles, ACLs, Web service Interfaces, Service Groups, and Service Containers into OpenText CARS. Do not delete this connector because all AppWorks Platform components use it to connect to OpenText CARS.

MDM publisher

While installing AppWorks Platform, you must provide database details for placing MDM content. If this step is skipped, the MDM service containers are not created. Later when you want to work with MDM, you must create MDM publisher and MDM service. MDM model contains hub and spokes for managing data. MDM Publisher propagates (publishes) the changes at Hub to all the subscribing spoke systems. It also handles the Publish To Run-time process (PTR) of MDM Models.

MDM service

If MDM is not configured during the installation of AppWorks Platform then MDM Service must be created later to work with MDM. The MDM Service receives the changes from all the spoke systems and updates the Hub data store. It also performs bulk uploads of data. It handles the requests for log viewer, to view the upload and the synchronization logs.

Notification service

The Notification service of AppWorks Platform deals with notifying users about business events that occur. A notification can be delivered to the AppWorks Platform Inbox of a user, to an external email address, or to an application that uses the Inbox. It facilitates the deployment of various Delivery Models, Worklists, and Dispatch Algorithms.

Rule repository service connector

Rule Repository connector is used to define and evaluate Rules and Decision Cases in AppWorks Platform. The Rule Repository connector facilitates modeling of complex Rules and Decision Cases and enables you to invoke Java program constructs from rule expressions.

Scheduler service connector

Scheduler Service Connector enables you to deploy and execute various schedules that are used for managing time-driven applications.

Security administration

The functions of security administration such as creating trust stores and so on are available in the 'System' organization. The security administration connector is created when you want to implement the functionality of security administration in other organizations within AppWorks Platform. You can also create it for load balancing.

Single Sign-On

Single Sign-On (SSO) is a mechanism for managing authentication. The SSO connector is used for enabling auditing for authentication. You may create multiple SSO connectors for configuring fail over. For information on configuring SSO for failover, refer to Configuring SSO for Fail Over.

Universal description, discovery, and integration (UDDI) connector

UDDI connector enables:

- Locating and consuming external Web services in a UDDI registry.
- Viewing WSDL for a selected Web service.
- Publishing AppWorks Platform Web service operations to a UDDI Registry.
- Setting specific entities and services as preferences so that AppWorks Platform Web service operations can be published to them.
- Creating new entities and services and publish AppWorks Platform Web service operations into them.
- Modifying business entity and service names, description. Add and delete category and identifier bags. Update, add, and delete contact information.

User management

The User Management Connector facilitates the centralized management of users and roles. It provides an easy way of populating user information as batch operations into AppWorks Platform.

WS-AppServer connector

The WS-AppServer connector provides connectivity and data management capabilities while working with relational databases and Java classes. It performs the following functions:

- Establishes a bi-directional link with data in a relational database and objects in code
- Supports all leading DB Vendors
- Supports creation of Custom objects representing data from multiple relational tables
- Supports executing SQL queries on the database
- Handles all types of relations among data objects (1-1, 1-n, n-n)
- Manages data persistence in statefull and stateless conditions
- Supports generation of Java code for both standard and custom data models
- Facilitates adding custom logic to the generated Java code
- Supports adding event listeners to the application logic that react and log information on events before, during, and after a transaction takes place
- Supports generation of Web service operations for both standard and custom data models

XForms connector

The XForms connector handles all requests and responses related to content in an XForm. During design time, this connector is used to retrieve property sheets, and to preview and publish XForms. At runtime, this connector is used to retrieve content for an XForm, and configure the compression and content-expiry settings.

HTTP connector

The OpenText HTTP connector provides connectivity to web servers and services exposed through HTTP.

HTTP connector accesses the following services:

- Web services with XML payload as request and response
- REST services with XML and JSON

HTTP connector supports the following HTTP operations:

- GET
- POST
- PUT
- DELETE

Any service that has to be accessed through HTTP connector must be available as a SOAP web service in the platform since AppWorks Platform interprets only SOAP services. The HTTP connector, with pluggable request and response handlers, maps the SOAP request from AppWorks Platform to the corresponding request expected by a service. For more information on the features of HTTP connector, see [Features of HTTP connector](#).

JMS connector

The OpenText JMS connector provides reliable connectivity between service containers or applications through JMS. The JMS connector connects to JMS middleware such as ActiveMQ, Websphere MQ, Sun J2EE, and others with the AppWorks Platform service container.

The JMS connector supports:

- JMS Specification Version 1.1
- Messaging using Queues and Topics
- Distributed transactions
- Different message formats, such as clear text, XML, and base64
- Request/Reply and a provision to generate a correlation ID
- Different types of message pollers, such as Interval Poller, Compatibility Poller, Request-Reply Poller, and Custom Poller
- Works as a transporter of messages between AppWorks Platform and Queues or Topics
- Exposes Web services to consumers to work with the JMS connector

Repository

Repository Service acts as a repository for AppWorks Platform objects. This service container contains the following:

- Tag Service - This service manages tags
- Task Service - This service retrieves and stores tasks to the XDS
- XML Store - This service reads and write XML objects to a persistent storage

For basic applications involving Application-connector development, you must include `cordyscp.jar` located in the `<AppWorks Platform_installdir>`, for building the java applications.

Part III

HOW TOs

Chapter 19

Business process modeling (How Tos)

This section provides a list of topics that help you easily perform specific tasks while modeling business processes using AppWorks Platform. The examples used in the below topics help understand how a task can be performed.

- [Invoking a web service in rules](#)
- [Invoking a web service through schedules](#)
- [Triggering a business process from rules](#)
- [Triggering a business process from schedules](#)
- [Triggering a business process model from a WS-AppServer application](#)
- [Triggering a notification request from an XForm](#)
- [Triggering rules from a business process model](#)
- [Triggering a business process model from an XForm](#)
- [Using data transformation web service in a business process model](#)
- [Using a web service in a data transformation model](#)
- [Using a web service in a business process model](#)
- [Using an external web service in a business process model](#)
- [Using an Inbox model in a business process model](#)
- [Using an Email model in a business process model](#)
- [Using human interaction in a business process model](#)
- [Using human interaction in a case model](#)
- [Using a business rule in an XForm](#)
- [Using WS-AppServer business logic in XForms](#)
- [Using an automatic follow-up connector in a case model](#)
- [Using free follow-up in a case model](#)
- [Using a document received event in a case model](#)
- [Using an intermediate follow-up connector in a case model](#)
- [Using the applicability service in a case model](#)
- [Triggering a case model from an XForm](#)
- [Configuring attachments in BPM](#)

- Publishing business identifier values
- Using a manual follow-up connector in a case model
- Enabling debugging on previous business process models

Invoking a web service in rules

AppWorks Platform provides a set of Web service interfaces, each of which can be used for various transactions on the back-end. You can view and manage the existing Web service interfaces and Web service operations, and can also generate default Web service operations and build custom Web service operations for databases. You can invoke these Web service operations while defining rule actions on a database template.

Before you begin:

- You must have already generated a [Web service](#).
- You must have a data template to define rules on it.

Depending upon how you access a document, creating a rule and accessing the business object can be done in various ways. See Accessing the Rule Modeler in the *AppWorks Platform Administrator's Guide* for more information.

To invoke a web service in rules:

1. On the **Properties** tab, set the properties of the rule.
For more information on the various fields on the rule properties interface, refer to properties of a rule.
2. To define the rule behavior, click the **Behavior** tab.
3. To invoke a Web service, do the following:
 - a. In the Rule Definition pane, right-click and select > **Actions** > **Invoke Web service**.
 - b. Type the **Action Name**.
For more information on naming the action, refer to Guidelines for Naming Rule Actions.
 - c. Click  on the rule modeler tool bar.
A Quick Access Menu is displayed.
 - d. Click  Insert on the sidebar.
All the related artifacts are displayed on the Quick Access Menu.
 - e. Drag the required Web service from the Quick Access Menu to the Action - Invoke WebService section.
The Action - Invoke WebService pane is populated with the Method Name. The request based on the WSDL is displayed in the Request text box below.
 - f. Click **Add**.
The type of action along with the name is displayed next to then in the Rule Definition pane.

4. Click **Save**.

A rule is defined on the template that invokes a Web service. Repeat the task to add more Web service actions to the rule.

Invoking a web service through schedules

AppWorks Platform provides a set of method sets, each of which can be used for various transactions on the back-end. You can view and manage the existing method sets and methods, and can also generate default methods and build custom methods for databases. These methods (Web services) can be invoked automatically at scheduled intervals.

Before you begin:

- You must have already generated a Web service.

To invoke a web service through schedules:

1. Select a starting point and click  to open the schedule modeler.
2. Select the **Schedule Type**.
Depending upon the kind of schedule type selected, the schedule table displays the required columns.
For more information on types of schedules, see [Types of Schedules](#).
3. Enter the **Number of times to execute** the scheduled event.
The default is 1. To run the schedule infinitely, specify -1.
4. Click **Finish**.
5. Select **Auto Deploy** to deploy the schedule as soon as it is published.
If this option is selected, the schedule is sent to the Active Schedules tab directly.
6. Click  and specify the date and time at which the schedule is to be executed.

Use the following guidelines for date and time:

For	Do this
Hourly schedules	Select the minute of the hour from the Every hour at list
Daily schedules	Type the time of the day in Every day at
Weekly schedules	Select the day of the week from Every week on , and type the time of day
Monthly schedules	Select the day of the month from Every month on the , and type the time of the day
First day of month schedules	Type the time of the day in Every first day of the month at
Last day of month schedules	Type the time of the day in Every first week day of the month at

For	Do this
First week day of month schedules	Type the time of the day in Every first week day of the month at
Last week day of month schedules	Type the time of the day in Every last week day of the month at
Fortnightly schedules	Type the time of the day in Every fortnightly at
Duration schedules	Specify the duration after which the schedule should be triggered in Time
Instant schedule (Runnow)	Type the name of the schedule and provide the target information

7. In Target Document Type, select the target and click  to select the required Web service.
The name and the definition of the selected Web service is displayed in Target and Request XML respectively. Supply the appropriate parameter values to the Web service.
8. Click **Save**.
The Save Document dialog box opens.
9. Enter a **Name** and meaningful **Description** of the schedule.
10. Click  next to Save in Folder, select a location to save this schedule, and click OK.
11. Click **Save**.
12. To deploy the schedule, see [Deploying Schedules](#).

The Web service is scheduled to be invoked at the scheduled intervals.

Testing web service operations using service test tool

Before you begin:

You must have the role of a developer for testing web service operations.

The Service Test Tool is used to test the Web service operation and check if the SOAP Request and Response are generated properly to send and receive SOAP messages. The tool offers a feature to compose and edit the outgoing SOAP messages.

To test web service operations using service test tool:

1. On the Welcome page > My Applications, click  (Service Test Tool).
The Service Test Tool window opens.
2. From the Services list, select the Service Container.
The associated Web service Interfaces (method sets) are populated in the Web Services Interface list.

3. Select a Web services interface.
The associated operations (methods) are populated in the Operations list.
4. Select an operation and click **Compose the Request**.
The request SOAP message for the selected Web service operation is composed and is displayed in the Test Request frame.
5. Click **Test Operation** to send the message to the particular Service Group.
The Result Messages section is updated with the request and response messages.
6. Click a **Response or Request Message** for the results of the Web service operation for the given parameter value.
See Result Messages Interface in the *AppWorks Platform Administrator's Guide* for information on the Result Messages section.
The Service Test Tool - Response Message or Request Message dialog box opens with the message received in XML format.
7. Click **Clear Messages** to clear the request and response messages from the Result Messages section.

Certain parameters that are set in the SOAP request constitute the message options required for reliable messaging and include No Reply and Transactional. Transactional indicates that the message has a preference to be sent using a transactional transport such as JMS. If the receiving service has no transactional transport, the connection points in a non-transactional transport such as a socket will be used. Depending on the option selected, following values are set for the message options while sending the SOAP request.

- When neither of the options is selected, the default message options value is set to 0.
- Selecting No Reply sets the message options value to 1.
- Selecting Transactional sets the message options value to 2.
- Selecting both No Reply and Transactional sets the message options value to 3.

Triggering a business process from rules

You can trigger a business process from a rule using the Trigger Business Process option in a rule modeler. For instance, let us assume that a customer has placed an order. On inserting the Order, a rule is triggered, which checks whether the items exist in the inventory. If the items do not exist, the transaction should abort. If the items exist, the order processing business process is triggered.

Before you begin:

- You must have a business process created and published to runtime.
- A business object template / schema fragment must exist prior to the creation of a rule, and every rule must be associated to a rule group.

Depending upon how you access a document, creating a rule and accessing the business object can be done in various ways. See Accessing the Rule Modeler in the *AppWorks Platform Administrator's Guide* for more information.

To trigger a business process from rules:

1. On the **Properties** tab, set the properties of the rule. For more information on the various fields on the rule properties interface, see [properties of a rule](#).
2. To define rule behavior, click the **Behavior** tab.
3. To trigger a business process, while defining a rule action, do the following:
 - a. In the **Rule Definition** pane, right-click then > **Actions** > **Trigger Business Process**.
 - b. Type the name of the action in **Action Name**. For more information on naming the action, see Guidelines for Naming Rule Actions- in the *AppWorks Platform Administrator's Guide*.
 - c. Click  on the rule modeler toolbar. A Quick Access Menu appears.
 - d. Click  on the sidebar. All the related artifacts are displayed on the Quick Access Menu.
 - e. Drag the required business process from the **Quick Access Menu** to the **Action - Trigger Business Process** section. The Action - Trigger Business Process pane is populated with the Process Name. The input message of the selected process is displayed in the Message text box below.
 - f. In the message box, specify the required parameters of the input message to be sent to the process and click **Add**. The type of action along with the name is displayed next to then in the Rule Definition pane.
4. Click **Save**.

Triggering a business process from schedules

A business process can be triggered at scheduled intervals.

Before you begin:

- You must have a business process created and published to runtime.

To create a schedule template:

1. [Select a starting point](#) and click  to open the schedule modeler.
2. In the Schedule Modeler, select the **Schedule Type**. Depending upon the kind of schedule type selected, the schedule table displays the required columns accordingly. For more information on types of schedules, see [Types of Schedules](#).
3. Type the number of times that the schedule event should be triggered in **Number of times to execute**.

For Runnow type of schedules, the default value for **Number of times to execute** is 1. To run the schedule infinitely, specify -1.

4. Select **Auto Deploy** to deploy the schedule as soon as it is published. If this option is selected, the schedule is sent to the Active Schedules tab directly.
5. Click  and specify the date and time at which the schedule is to be executed.

For	Do this
Hourly schedules	Select the minute of the hour from the Every hour at list
Daily schedules	Type the time of the day in Every day at
Weekly schedules	Select the day of the week from Every week on , and type the time of day
Monthly schedules	Select the day of the month from Every month on the , and type the time of the day
First day of month schedules	Type the time of the day in Every first day of the month at
Last day of month schedules	Type the time of the day in Every first week day of the month at
First week day of month schedules	Type the time of the day in Every first week day of the month at
Last week day of month schedules	Type the time of the day in Every last week day of the month at
Fortnightly schedules	Type the time of the day in Every fortnightly at
Duration schedules	Specify the duration after which the schedule should be triggered in Time
Instant schedule (Runnow)	Type the name of the schedule and provide the target information

6. Select the target in Target Document Type and click  to select the required Business Process Model. The name and the definition of the selected Business Process Model is displayed in Target and Request XML. Supply the appropriate parameter values to the Business Process Model.
7. Click **Save** to save the schedule.
8. To deploy the schedule, refer to [Deploying Schedules](#).

The business process is scheduled to be executed at the scheduled intervals.

Triggering a business process model from a WS-AppServer application

WS-AppServer applications contain the necessary business logic to perform an activity. Usually, the logic within these applications is executed as part of a business process cycle. On the other hand, you can also trigger a business process model from WS-AppServer application, during runtime. The following procedure describes this process.

Before you begin:

- You must publish the business process model to runtime before using it.

To trigger a business process model from a WS-AppServer application:

1. [Creating a business process model](#).
2. In the Extension class of the Java class, add the code that triggers a business process model (see code snippet in the following example).
3. Regenerate the [Java code](#) and the Web service Interface and publish them to organization.

The application is equipped with the necessary logic to trigger a business process model.

Example

Assume that there is a sales application that works with data involving countries, regions, and places. In the application, you want to embed a logic that triggers a notification business process each time a new region is added. The process sends an email to all the concerned stakeholders.

The following content describes how this logic is integrated into the application code.

Assume that in your application, you have a class called Region. You also have a published business process model called "SendMail_vcmdemo10.bpm" that is programmed to send emails to the specified users.

In the Extension class of Region, append the following code.

```
public class Region extends RegionBase
{
    public Region()
    {
        this((BusObjectConfig)null);
    }
    public Region(BusObjectConfig config)
    {
        super(config);
    }
    @Override
```

```

public void onAfterCommit(AfterCommitObjectEvent event) {
    super.onAfterCommit(event);
    int messageXml = 0;
    try {
        messageXml = BSF.getXMLDocument().parseString(
            "<InputMessage>" +
            "<RegionID>" + getRegionID() + "</RegionID>" +
            "<RegionDescription>" + getRegionDescription
        () + "</RegionDescription>" +
            "</InputMessage>");
    } catch (UnsupportedEncodingException e) {
        throw new BsfRuntimeException(e);
    } catch (XMLException e) {
        throw new BsfRuntimeException(e);
    }
    //If inserting a Region, then notify all employees through a flow
    if (event.triggeredBy(StdTriggers.INSERT_OBJECT)) {
        int result =
WSUtil.executeProcess|WSUtil.executeProcess(
            ProcessType.DEFINITION,
            "3.Business Process Models/Test/SendMail_vcmdemo10.bpm",
            "Northwind Application",
            null,
            null,
            null,
            messageXml,
            true,
            true);
        System.out.println("result = "+Node.writeToString(result, true));
        //No need to delete messageXml node this is taken care by
        WSUtil.executeProcess() implementation
    }
}
}
}

```

During runtime, this code ensures that whenever a Region object is committed to the database, it sends out an e-mail to the receivers, with the predefined message content.

Thus, you trigger a business process model from a WS-AppServer application.

Triggering a notification request from an XForm

The Notification service container provides a mechanism to share information with a designated set of recipients. The information is sent as notifications through the Inbox or through an MS Outlook e-mail. Depending on the requirement, a notification may also contain information about alternate options available to you and the tasks that you need to perform to avail them.

When you [Creating a business process model](#), the Notification service container is available, by default. However, in some cases, it may be required to send the notification as an e-mail to the user; for example, to notify about incomplete information on a registration form. This topic describes the procedure to trigger a notification request directly from an XForm.

Before you begin:

- Ensure that the Notification, E-mail Connector, and XForms Runtime packages (including all dependencies) are installed, and specific roles for each application package are assigned to you.
- Ensure that the e-mail address is specified in the Welcome Page Preferences.
- Ensure that the required Web service operations of the Northwind database are created. For details on how to create Web service operations, see [Generating Standard Web Service Operations on a Database](#).

As an example, consider a business scenario where a product has been discontinued and the supplier must notify a customer about its discontinuation. Here, the supplier needs to choose a delivery mechanism for the notification, create a notification message containing additional information about the alternate products available, and send the notification message.

For this example, we use the Products table of the Northwind database (MS-SQL Server), for which the Northwind Web service operations must be first created. Also, consider the following XForm that is used to input product information.

To trigger a notification request from an XForm:

1. Create a model in XForms. In this case, use the DeliverMessage Web service operation of the Notification application package to create a non-transactional model in XForms. For details, see [Creating Models](#).
2. Design an XForm and attach the new model to it. In this case, create the Products Groupbox control and attach the non-transactional Web service operation created using the DeliverMessage Web service operation.
3. Add the custom request to the XForm using the XML Editor. In this case, add the following customized request (XML) to the XML Editor of the XForm. Because the DeliverMessageWeb service operation has a large number of fields, the request has been shortened to include only the mandatory fields.

```
<xml id="requestNotify">
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<DeliverMessage xmlns="http://schemas.cordys.com/1.0/notification">
<TARGET/>
<SOURCE>Notification</SOURCE>
<SENDER>cn=supplier1,cn=organizational
users,o=system,cn=cordys,o=northwind.com</SENDER>
<MESSAGE_TYPE>INFO</MESSAGE_TYPE>
<MESSAGE_DATA/>
<RECEIVER>
<PARTICIPANT_DN>cn=customer1,cn=organizational
```

```

users,o=system,cn=cordys,o=northwind.com</PARTICIPANT_DN>
<MAILING_LIST>
<cc>
<address>
<emailAddress/>
</address>
</cc>
</MAILING_LIST>
</RECEIVER>
<URL>TEST</URL>
</DeliverMessage>
</SOAP:Body>
</SOAP:Envelope>
</xml>

```

In the XML, the TARGET, SENDER, and PARTICIPANT_DN fields are hard-coded with the necessary values.

4. Add the following code for the Notify button of the XForm.

```

function btnNotify_Click(eventObject)
{
var request = requestNotify.documentElement.cloneNode(true);
//Target denotes the delivery mechanism - it can be inbox or mail, picked from radio
buttons
request.selectSingleNode('TARGET').text = targettype.getValue();
//URL denotes the URL of the task
request.selectSingleNode('URL').text = taskurl.getValue();
//This is the message sent to customer
request.selectSingleNode('MESSAGE_DATA').text = message.getValue();
//Send request
NotifyModel.setMethodRequest(request);
NotifyModel.reset();
}

```

The XForm for sending notifications is created.

Triggering rules from a business process model

In a Business Process Model, you can model activity behavior using rules, developed in WS-AppServer. You can build rules on the WS-AppServer generated templates and enhance your application's functionality, thereby streamlining the business process.

To trigger rules from a business process model:

1. Create WS-AppServer Service Container and point it to your database.
Select **Enable Rules** when you create the WS-AppServer Service Container.
 2. [Creating database metadata](#) to get the tables from database to the AppWorks Platform environment.
 3. Create a WS-AppServer package.
Generate Web services for this package. WS-AppServer templates are generated along with the package.
 4. [Build a rule on a WS-AppServer template](#).
You can trigger rules on insert, update, or delete.
 5. [Select a starting point](#) and click  (Business Process Model) to open a business process model.
 6. From **Workspace Documents** <Solution> > <Project>, drag the required Web service on to the business process model.
 7. Select the Web service activity and click the **Message Map** tab on the toolbar.
The message map for the Web service activity appears.
 8. Do one of the following to create an assignment:
 - Drag the source element from the **Source** column and drop it on the left box in the **Assignments** column. Likewise, drag the target element from the **Target** column and drop it on the left box in the **Assignments** column.
 - Right-click the element or message in the **Target** column, and select **Create Assignment**.
By default, this creates a Fixed Value assignment.
 9. Complete the **Message Map** assignments for other activities in the process model.
 10. [Configure the other constructs used in the process model](#).
 11. Right-click in the business process modeling environment and select **Business Process Execution** > **Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select **Business Process Execution** > **Validate and generate BPML**.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings, resolve them and re-validate the business process model.
 12. Right-click in the business process modeling environment and select **Business Process Execution** > **Publish to Organization**. Alternatively, go to **Workspace Documents** <project>, right-click the <business process model> and select **Publish to Organization**.
The business process model is published to the organization.
 13. Right-click in the business process modeling environment and select **Business Process Execution** > **Run**.
The business process model is instantiated.
- The rule is triggered from a business process model.

Example of triggering rules from a business process model

The following example describes the procedure to use a rule from a business process model to decide the salary of a new candidate (Software Engineer) based on work experience and skill set rating.

Salary for the candidate is determined based on a rule:

- If employee experience is \geq 3 years, and employee skillset rating is \geq 5, then assign employee salary as 6.0 K, else if employee experience $<$ 3 years, and employee skillset rating is \leq 5, then assign employee salary as 4.0 K

Business requirements

The business process should satisfy the following requirements:

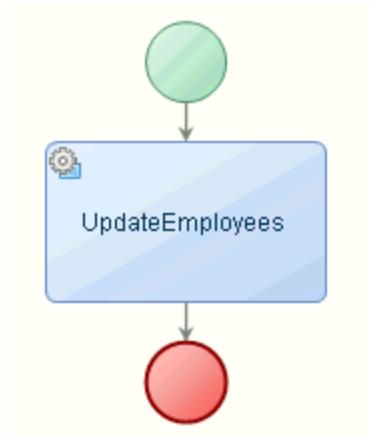
- WS-AppServer Service Container
- Database metadata
- WS-AppServer package and Web services generated
- Rule created on WS-AppServer template
- Business process model to use the rule

To satisfy the prerequisites:

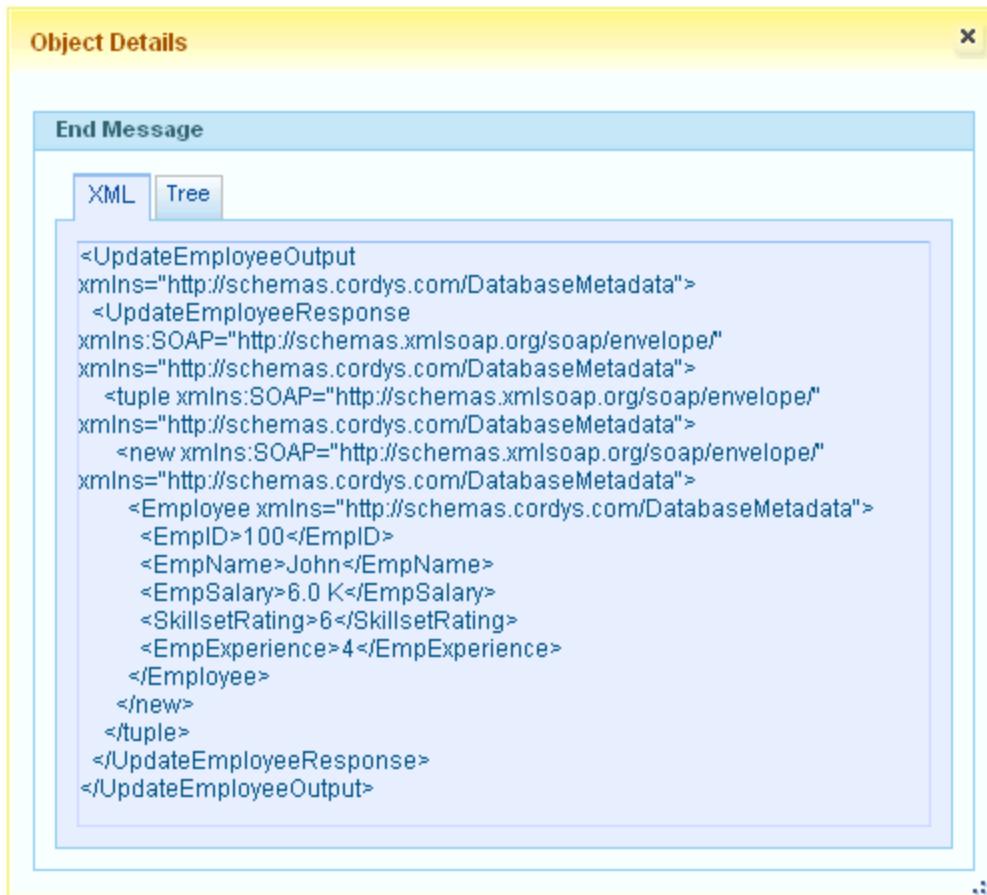
1. Create a WS-AppServer Service Container and point to your database. Select **Enable Rules** when you create the WS-AppServer Service Container.
2. Create a Database Metadata to get tables from the database to the AppWorks Platform environment.
3. Create a WS-AppServer package.
4. Generate the Web services on the WS-AppServer package.
5. Generate the **UpdateEmployee** Web service. WS-AppServer templates are generated along with the package.
6. Create a rule group, **EmpRule Group**.
7. Build a rule **EmpSalRule** on Employee model.

To trigger rules from a business process model, create a simple BPM, CalEmpSalary:

1. Drag the generated **UpdateEmployee** Web service on to the business process modeling environment.
When you drag UpdateEmployee on to the business process model, an input, the following are also created: UpdateEmployeeInput and output, UpdateEmployeeOutput, and UpdateEmployeeFaultDetail.
2. Drag the **UpdateEmployeeOutput** on to the End event of the business process model.
3. Design the business process model that appears as follows.



4. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**.
Any warnings appear in the Warning List pane.
5. If there are no warnings, right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**.
6. Right-click in the business process modeling environment and select **Business Process Execution > Run**.
7. Right-click in the business process modeling environment and select **Business Process Execution > Show Process Instances**.
The Instances by Process Definition window opens.
8. In the **Status** column, right-click **Complete**, and select **Show End Message** to view the employee salary.
The Object Details dialog box opens showing 6.0 K as the employee salary.



You can determine the employee salary by triggering rules in a business process model.

Triggering a business process model from an XForm

A Business Process Model is an executable model that comprises a set of activities or processes designed to produce a specific output. A business process model can be designed for a particular type of customer or market. The activities in a business process model may be automatic or may require manual intervention to proceed.

Before you begin:

- Ensure that the Business Process Designer, Business Process Engine, and Notification application packages (including all dependencies) are installed, and the appropriate roles of each application package are assigned to you, as applicable.
- Ensure that the e-mail address is specified in the Welcome page preferences. To use MS Outlook, ensure that the AppWorks Platform E-mail Connector application package is installed. If the notification is to be sent to the Inbox, ensure that the Inbox menu is available to you.

In business process model scenarios requiring manual inputs, you can use XForms to capture user specifications. For example, an XForm can be used in a workflow that involves the approval of a leave request. Alternately, XForms can be used to initialize a business process model.

As an example, consider a business scenario where you complete a loan application to apply for a loan. After you fill in the necessary set of details, and click the Submit button, the loan process begins. Submitting the form triggers a business process model using the inputs that you provided to process your loan request. Consider the following example for this business process.

Loan Application Form

Name	Nancy Davilio
Phone Number	6120023
Income	1000000
Property Value	8000000
Loan Amount	3000000

Submit

After the data is entered, clicking Submit triggers the business process model.

To use the data provided, attach a model to the XForm:

1. Identify the required data, and create the appropriate XForm.
2. Add the following XML to the XML Editor of the XForm:

```
<xml id="loanrequest">
    <LoanRequest xmlns="http://schemas.cordys/com/loanrequest">
        <Name/>
        <Phone/>
        <Income/>
        <PropertyValue/>
        <RequestedAmount/>
    </LoanRequest>
</xml>
```

The XML represents the data to be provided in the XForm.

3. Create a non-transactional model, and bind it to the controls in the XForm.
For more information, see [Creating Models](#). In this case, a data model called `LoanModel`

is created and bound using the `InitDone` event of the XForm.

```
function Form_InitDone(eventObject) { //Bind model to empty data when the form loads
LoanModel.putData(cordys.cloneXMLDocument(loanrequest.XMLDocument),true); }
```

4. Create a non-transactional Model that uses the `ExecuteProcess` Web service operation to fire a request to initiate the business process model from the XForm.
In this case, add the following XML to the XML Editor.

```
<xml id="executeprocess">
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<ExecuteProcess xmlns="http://schemas.cordys.com/bpm/execution/1.0">
<type>definition</type>
<receiver>3.Business Process Models/LoanApproval_
northwind1.0.bpm</receiver>
<message>
<Input/>
</message>
</ExecuteProcess>
</SOAP:Body>
</SOAP:Envelope>
</xml>
```

The XML creates a non-transactional model called `BPMModel` that fires the request to be sent to the Business Process Management service container to initiate the business process model. The XML contains only the mandatory parameters that are required to start the business process model. Now, the `ExecuteProcess` Web service operation is associated with data.

5. Write the code to send requests to the business process model.
Add the following code for the *onclick* event of the **Submit** button.

```
function btnSubmit_Click(eventObject)
{
//Get the request for execute process
var request = cordys.cloneXMLDocument(executeprocess.XMLDocument);
//Get data from what customer typed
var data = cordys.cloneXMLDocument(LoanModel.getData());
//put data inside the request, and send request
var inputNode = cordys.selectXMLNode(request,"/*[local-name()='Input']");
cordys.appendXMLNode(data.documentElement,inputNode);
BPMModel.setMethodRequest(request);
BPMModel.reset();
}
```

The code sends a request with the specified input parameters to the business process model, and the business process model is started.

The XForm from which you can trigger the business process model is created.

Using data transformation web service in a business process model

Data transformation converts data from a source data format into a destination data format. Data map is a model that captures the transformation logic between the source and destination data models. You can create a Web service that captures this logic and use it in a business process model.

To create a web service in a business process model:

1. [Create a data transformation model](#).
2. [Create a Web service to execute the data transformation model](#).
3. [Select a starting point](#) and create a simple business process model.
4. Drag the required application service (data transformation Web service) from the project content tree in Workspace Documents to the business process modeling environment to attach to the business process model.
An input data transformation and output data transformation also appear under the business process model in the project content tree.
5. Drag the input data transformation to the **Start Event** of the business process model.
6. Drag the output data transformation to the **End Event** of the business process model.
7. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Business Process Execution > Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings, the number of new warnings are displayed in the status bar. Resolve the errors and re-validate the process model.
8. Right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization. The business process model is published to the organization.
9. Right-click in the business process modeling environment and select **Business Process Execution > Run**.
The Start Message dialog box opens.
10. Enter the input parameter values and click **OK**.
The business process model is instantiated.
11. Right-click in the business process modeling environment and select **Business Process Execution > Show Process Instances**.
The Instances by Process Definition window opens.
12. In the **Status** column, right-click **Complete**, and select **Show Start Message** to view the source data XML format.
The Object Details dialog box opens with the source data XML format.

13. Similarly, select **Show End Message** to view the converted destination data XML format.

The Object Details dialog box opens with the transformed destination data XML format.

The source data in one XML format is converted into another XML format by using a Data Transformation Web service in a business process model.

Example of using a data transformation in a business process model

The following example describes the procedure to convert data of one XML format to another XML format.

For example, you have a Source Template in the following XML format:

```
<employee_record>
  <employee_name/>
  <employee_number/>
  <employee_designation/>
</employee_record>
```

and a Target Template in the following XML format:

```
<emp_rec>
  <emp_name/>
  <emp_no/>
  <emp_desig/>
</emp_rec>
```

The Source Template has different field names to represent the same kind of data than that of the Target Template. For example, `<employee name>` in Source Template is `<emp_name>` in Target Template, and applies to the remaining details.

- Create EmployeeDataTransformation, a data transformation, to convert data from Source Template XML format to Target Template XML format.
- Create a Web service to execute the data transformation and use it in a business process model.

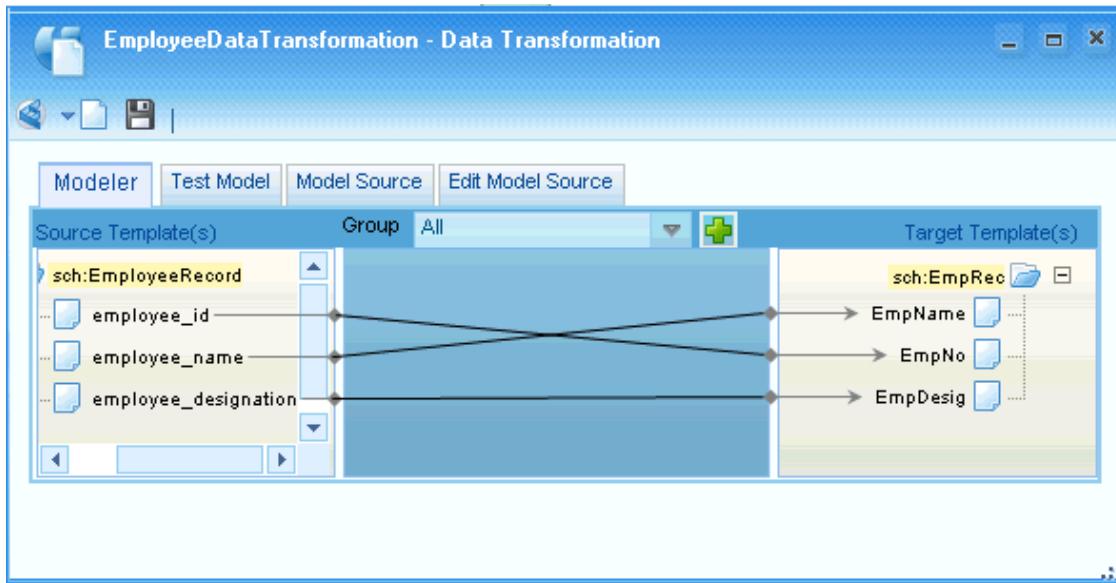
Business requirements

The business process should satisfy the following requirements:

- A Data Transformation model to convert data of one XML format to another XML format.
- A Web service to execute the data transformation.
- A business process model to use the data transformation model.

Ensure that you:

- Create a Source Template and Target Template.
- Create a Data Transformation model, EmployeeDataTransformation.
- Map the Source Template and Target Template as:



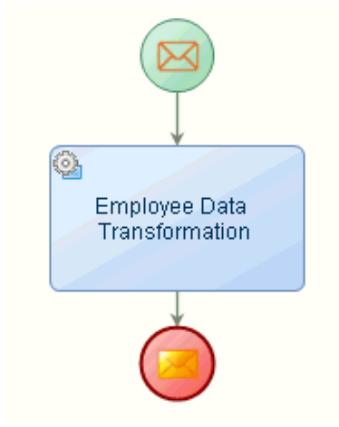
- A Web service, Employee Data Transformation, is created to use the data transformation in a business process model. You may note that when you create a Web service for the data transformation, an input, DataTransformation_EmployeeDataTransformationInput and output, DataTransformation_EmployeeDataTransformationOutput data transformations are also created.

Using the data transformation in a business process model

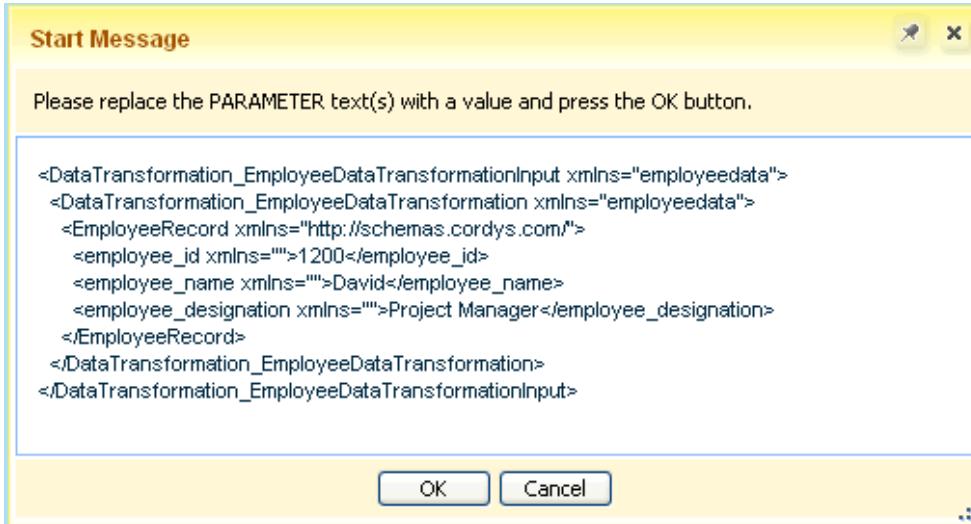
To create a simple business process model, DataTransformBPM:

1. Drag the created **Employee Data Transformation** onto the business process modeling environment.
The DataTransformation_EmployeeDataTransformationInput and DataTransformation_EmployeeDataTransformationOutput appear under the business process model in the project content tree.
2. Drag the **DataTransformation_EmployeeDataTransformationInput** input data transformation on to **Start** event.
3. Drag the **DataTransformation_EmployeeDataTransformationOutput** output data transformation on to the **End** event.

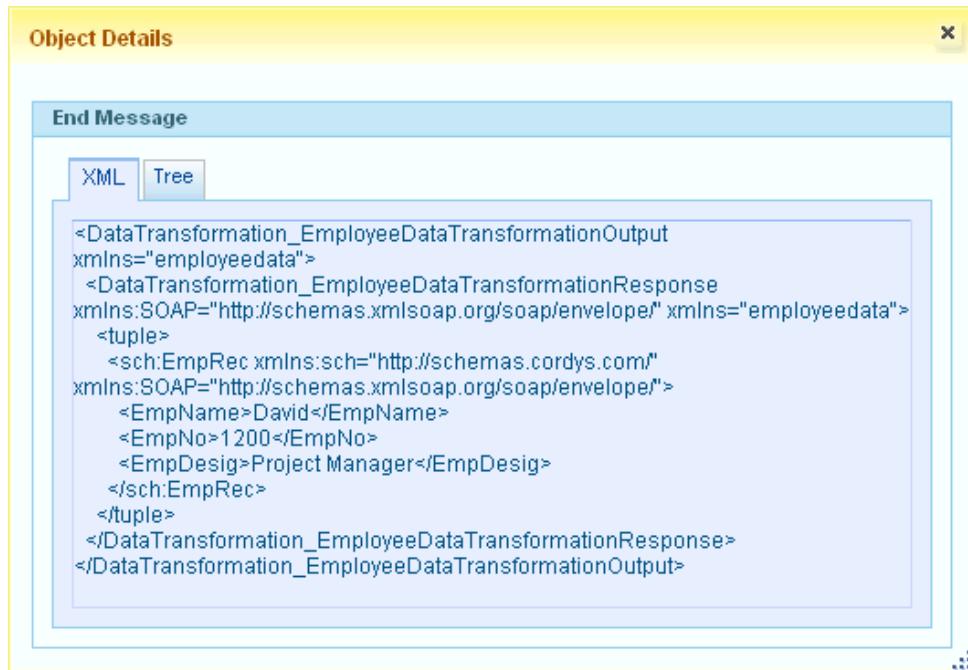
The business process model appears as:



4. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**.
If there are any warnings they will be displayed in the Warning List pane.
5. If there are no warnings, right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**.
6. Right-click in the business process modeling environment and select **Business Process Execution > Run**.
The Start Message dialog box opens.
7. Enter the values **1200**, **David**, and **Project Manager** as parameters for employee_id, employee_name, and employee_designation as shown:



8. Click **OK**.
9. Right-click in the business process modeling environment and select **Business Process Execution > Show Process Instances**.
The Instances by Process Definition window opens.
10. In the **Status** column, right-click **Complete**, and select **Show End Message** to view the converted destination data XML format.
The Object Details dialog box opens with the transformed destination data XML format.



The source data in one XML format is converted into another XML format by using a Data Transformation Web service in a business process model.

Using a web service in a data transformation model

AppWorks Platform provides a set of Web service interfaces, each of which can be used for various transactions on the back-end. You can invoke these Web service operations while transforming data.

For example, assume that the source template has an element called EmpID that holds the employee ID. During transformation, the employee ID needs to be replaced with the employee name, before being mapped to the empName element in the target template. You can have the function run a Web service that would retrieve the information from the database. The function can scan the employee ID against the retrieved value pairs, and substitute it with the corresponding employee name. The result is then mapped to the target.

To use a web service in a data transformation model:

1. Access the Data Transformation Modeler () to create a data transformation model.
The Data Transformation modeler page appears.
2. Drag the required source schema or object template to the **Source Template** pane of the Modeler tab.
The schema of the object is displayed in the Source Template pane.
3. Drag the required target schema or object template onto the **Target Template** pane of the Modeler tab.
The schema of the object is displayed in the Target Template pane.
4. Drag the required Web service, from the Workspace Explorer, or the Quick Access Menu from My Documents view, to the modeler.
The Organization DN and the Web service Request fields automatically populate.
5. In the **Source Template**, click the attribute and in the request block, select the required parameter.
The XPath of the attribute is displayed as a parameter.
6. In **Response Element to be Mapped**, type the element/node of the response that is to be mapped to the target element/node.
7. In **Suppress Errors**, type **true** or **false** to suppress any errors from the execution of Web services.
 - If the Boolean value is set to **true**, errors (SOAP fault) during the execution of the Web service are suppressed. However, data transformation is carried out and the target element takes the value provided in the Default Value.
 - If the Boolean value is set to **false**, in the event of errors during the execution of the Web service, data transformation is aborted, and the error details are displayed in the response.
8. Type the default value that is to be provided in the target element, when there is a suppressed error in the Default Value.
9. On the Map Canvas, select the  (Trigger Web service) function, press the CTRL key, and select the target element.
A line connecting the (Trigger Web service) and the target element appears. This line indicates that the output value, based on the inputs to the Trigger Web service from the source template, is mapped to the selected target element. The target can be a node, an attribute, or a function block.
10. Click  to save the transformation model.

A data transformation model is created using a Web service.

To test the transformation model:

- Click the **Test** tab.
See [Testing a data transformation model](#) for more information.

Using a web service in a business process model

Whenever there is a need to achieve a piece of automated functionality or when you need to re-use that functionality over and again, you need to create a Web service so that it is available for use when required. You can also make the Web service accessible over Internet so that it is within your reach independent of geographical location or the network that you are connected to. With AppWorks Platform Web service , you can create Web services and use them in a business process model.

Before you begin:

You must have the Process Developer role in AppWorks Platform Business Process Engine package to use a Web service in a business process model.

To use a web service in a business process model:

1. [Select a starting point](#) and select  (Business Process Model).
2. Drag the required application service (Web service) from the Workspace Documents to the business process modeling environment to attach to the business process model.
3. Select the Web service activity and click the **Message Map** tab.
The Message Map window for the Web service activity opens.
4. Do one of the following to create an assignment:
 - Drag the source element from the **Source** column and drop it on the left box in the **Assignments** column. Likewise, drag the target element from the **Target** column and drop it on the left box in the **Assignments** column.
 - Right-click the element or message in the Target column, and select **Create Assignment**. By default, this creates a Fixed Value assignment.
5. Complete the Message Map assignments for other activities in the process model.
6. [Configure the other constructs used in the process model](#).
7. Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Business Process Execution > Validate and generate BPML.
The Validate progress window opens and when the validation is over, if there are any warnings or errors, then the warnings window opens. If there are warnings resolve them and re-validate the business process model.
8. Right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization.
The business process model is published to organization.
9. Right-click in the business process modeling environment and select **Business Process Execution > Run**.
The business process model is instantiated.

The Web service is used in the business process model.

Example of designing a business process model using a web service

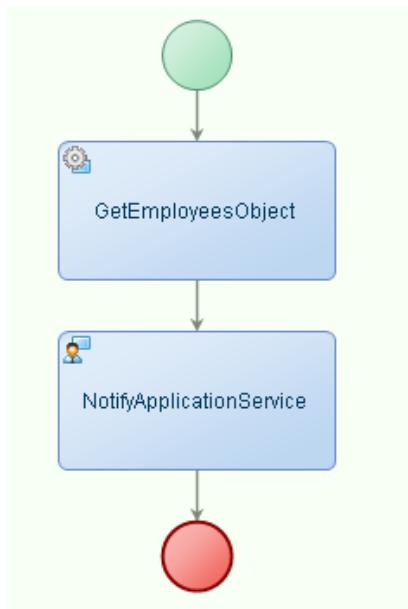
The following example describes the procedure to create a business process model using a Web service to check the 'title' of an employee.

The business process must satisfy the following requirements:

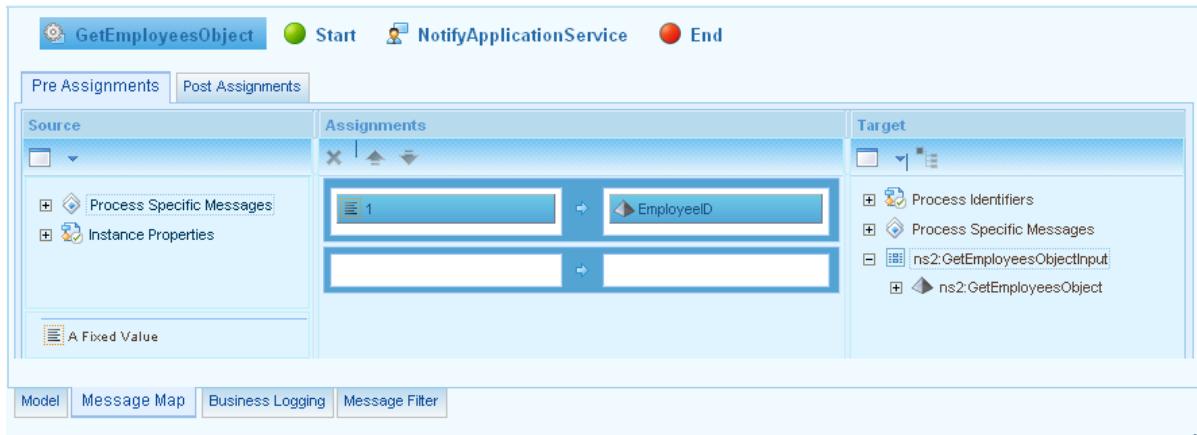
- Employee ID is provided as an input for the process.
- The first name, last name, and the title of the employee are displayed.

1. Generate a Web service on the Employee table and create **Notify Application Service** XForm to display the employee first name and last name.

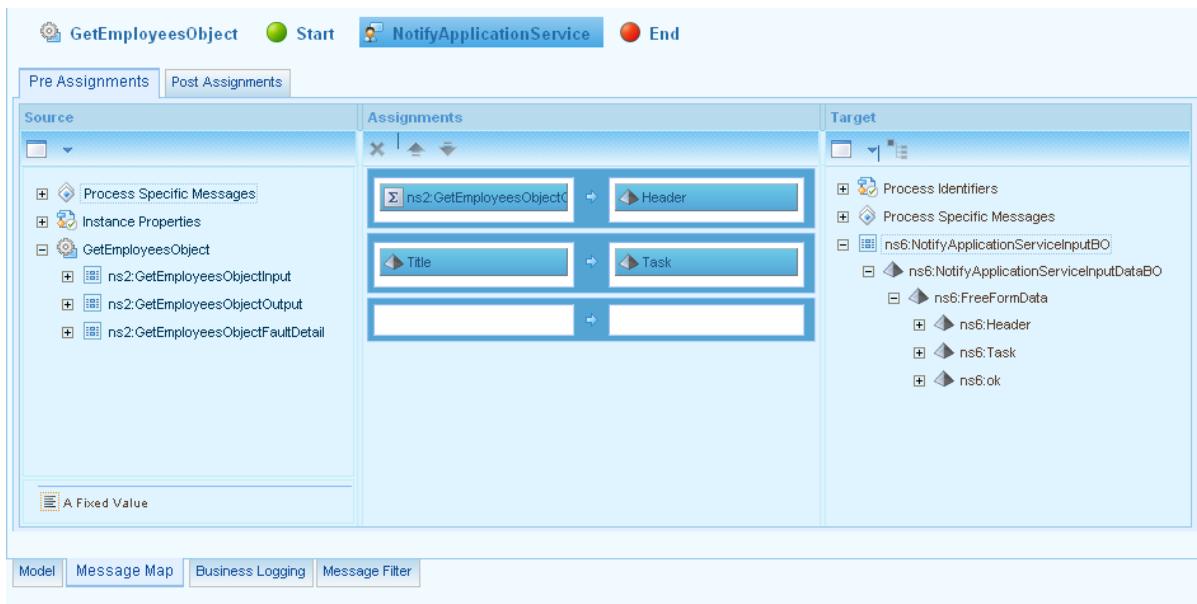
Using a Web service in a Business Process Model, You must next model the business process model and complete the Message Map for each activity in the business process model.



2. Create a business process diagram using the requirements listed above.
3. Click the **Message Map** tab at the bottom of the business process model window.
The Message Map appears displaying three columns: Source, Assignments, and Target.
4. Click **Get Employees Object** and create an assignment for the element as follows.



- Click **Notify Application Service** activity and create an assignment for the element as follows.



- Right-click in the business process modeling environment and select **Business Process Execution > Validate and generate BPML**. Any warnings display in the Warning List pane.
- If there are no warnings, right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**.
- Right-click in the business process modeling environment and select **Business Process Execution > Debug** to verify that the business process model is executed properly.

In the example, based on the employee ID, the first name, last name, and the title of the employee are displayed.

Note: When more than one instance of the same Web service is used in a business process model, the following approach can help avoid conflicts in the message map:

- Store the output of the Web services in a Process Specific Message so that the contextual values are saved for reuse later in the process flow.
- Use a multiple subprocess with a single read/write action, which then stores the contextual information of the Web services.
- Model the flow in a way that the message map data is used as soon as possible, before it gets updated with data from another Web service.

Using an external web service in a business process model

AppWorks Platform supports the use of external Web services to create a business process model. A Web service is a simple, open-standard, and XML-based Web application that allows interoperability between various platforms. A Web service may contain multiple operations or methods to provide a wide range of services.

Before you begin:

1. Navigate to **Tools > Internet Options > Security > Internet** (or Local Intranet) > **Custom Level > Miscellaneous**.
2. Set Access data sources across domains option to **Enable or Prompt**.
3. Select Internet or Local Intranet based on the way you are accessing AppWorks Platform.

To use an external web service in a business process model:

1. Generate AppWorks Platform Web Service operations for External WSDL.
The Web service is imported into the AppWorks Platform environment and appears in the project content tree view in Workspace Documents.
2. Create a UDDI Service Container to process the request or response of SOAP messages.
(See *AppWorks Platform Administration Guide* for more information.)
3. **Select a starting point** and click  (Business Process Model).
4. Drag the imported Web service from the project content tree view in Workspace Documents on to the business process modeler to design the business process model.
5. Select the Web service activity in the business process model and click the **Message Map** tab.
The Message Map for the Web service activity is displayed.
6. **Create assignments for the Web service activity** and for other activities in the business process model as necessary.
7. **Configure the remaining constructs used in the process model**.
8. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents > <Solution> > <Project> and right-click the <business process model> and select Business Process Execution > Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no

warnings. If there are warnings, the number of new warnings appear in the Warning List pane. Resolve the errors and re-validate the process model.

9. Right-click in the business process modeler and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents > <Solution> > <Project> and right-click the <business process model> and select Business Process Execution > Publish to Organization. The business process model is published to organization.
10. Right-click in the process modeler and select **Business Process Execution > Run**. Alternatively, go to Workspace Documents > <Solution> > <Project> and right-click the <business process model> and select Business Process Execution > Run. The business process model is instantiated.

You have successfully used an external Web service in the business process model.

Example of using an external web service in a business process model

A Web service that is exterior to AppWorks Platform environment is referred to an external Web service. The following example describes the procedure to use an external Web service in a business process model to convert temperature from one unit to another unit.

Business requirement

The business process should satisfy the following requirement:

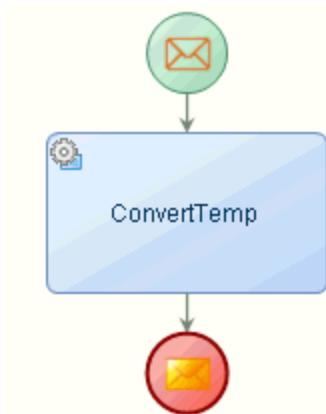
- An external Web service to convert temperature from one unit to another unit.

Prerequisites:

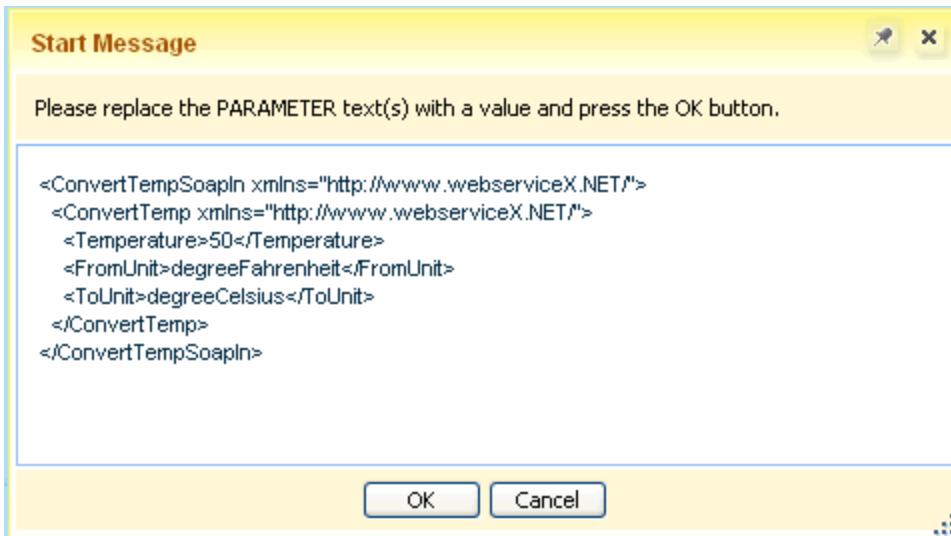
1. Create an external Web service, **External Web service**, with WSDL URL:
<http://www.webservicex.net/ConvertTemperature.asmx?WSDL>.
 The ConvertTemp Web service is imported into the AppWorks Platform environment and appears in the project content tree.
2. Create a UDDI Service Container.

Using the external Web service in a business process model, create a simple business process model, ExtWebSerBPM:

1. Drag the created **ConvertTemp** Web service on to the business process modeler. You may note that when you drag ConvertTemp on to the business process model, an input, ConvertTempSoapIn and output, ConvertTempSoapOut are also created.
2. Drag the **ConvertTempSoapIn** on to the Start event of the business process model.
3. Drag the **ConvertTempSoapOut** on to the End event of the business process model.

**Design the business process model that appears:**

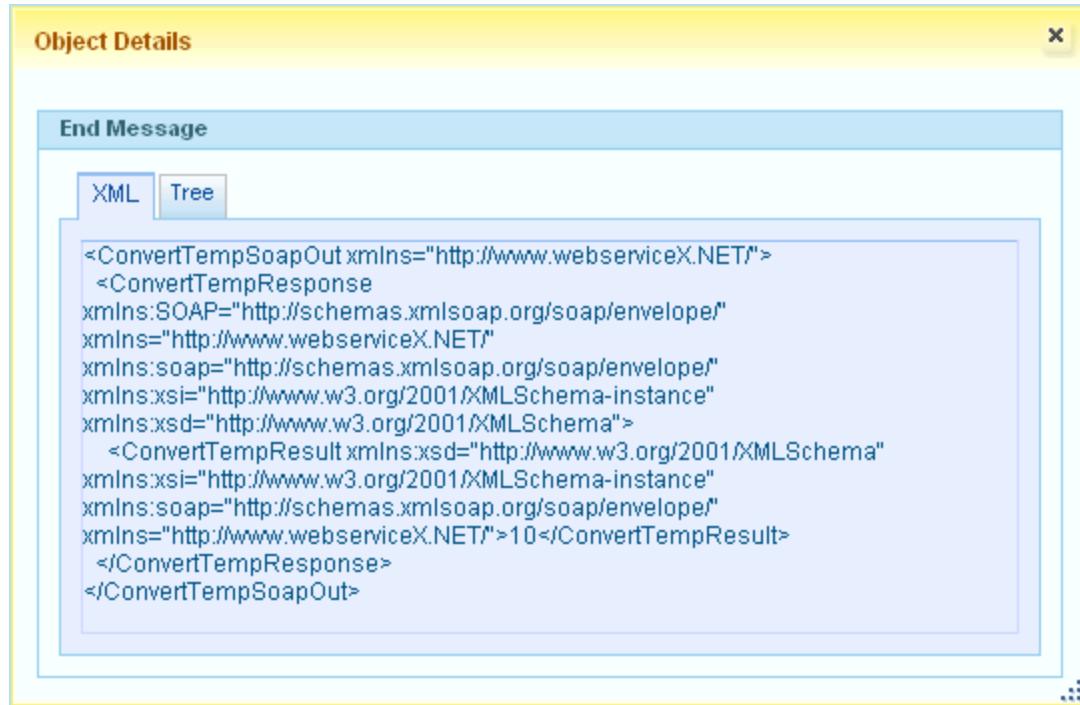
4. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPMN**.
If there are any warnings they appear in the Warning List pane.
5. If there are no warnings, right-click in the business process modeler and select **Business Process Execution > Publish to Organization**.
6. Right-click in the business process modeler and select **Business Process Execution > Run**. the Start Message dialog box opens.
7. Enter the values as **50, degreeFahrenheit, and degreeCelsius** as parameters for Temperature, FromUnit, and ToUnit.



8. Click **OK**.
9. Right-click in the business process modeler and select **Business Process Execution > Show Process Instances**.
The Instances by Process Definition window opens.

- Right-click **Complete** under Status column, and choose **Show End Message** to view the converted temperature.

The Object Details dialog box opens with 10, as the converted temperature in Fahrenheit.



The temperature is converted from one unit to another using an external Web service in a business process model.

Using an Inbox model in a business process model

An Inbox model helps you determine the entities in a message that must be displayed when a user receives a task. These tasks can be viewed in your My Inbox. You can use the Inbox model in a business process model to send tasks to a user's My Inbox.

To use an Inbox model in a business process model:

- [Create a delivery model using Inbox Model format.](#)
A delivery model using Inbox Model format is created.
- [Design a business process model.](#)
- From **Workspace Documents** <Solution> > <Project>, drag the required <User Interface> for which you created the delivery model (in Inbox model format) and drop it on the required activity in the business process model.
- Double-click the human activity (associated with User Interface on which you created the delivery model in Inbox model format). Alternatively, right-click the activity and select Properties.

The Activity -Task Properties pane appears. See [Activity Properties Interface](#) for details on using the Workflow Model tab.

5. Click the **Workflow Model** tab, and select **Use Inbox Model**.
6. To create an assignment by dragging the source element, drag the source element from the **Source** column and drop it to the left box in the **Assignments** column. Likewise, drag the target element from the **Target** column and drop it on the left box in the **Assignments** column.
7. To create an assignment by right-clicking the element, right-click the element or message in the **Target** column, and select **Create Assignment**.
By default, this creates a Fixed Value assignment.
8. Complete the **Message Map** assignments for other activities in the process model.
9. [Configure remaining constructs in the process model](#).
10. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Business Process Execution> Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings resolve them and re-validate the business process model.
11. Right-click in the business process modeler and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization. The business process model is published to the organization.
12. Right-click in the business process modeler and select **Business Process Execution > Run**.
The business process model is instantiated.

You have successfully used Inbox model in a business process model.

Example of using an Inbox model in a business process model

The following example describes the procedure to use an Inbox model in a business process model to display the business attribute, Emp ID, in My Inbox when a user receives a task.

Business requirements

The business process should satisfy the following requirements:

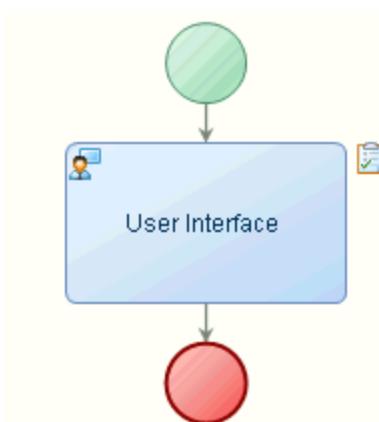
- A User Interface.
- A Delivery Model with Inbox Model format.
- A business process model to use the Inbox Model.

Ensure that:

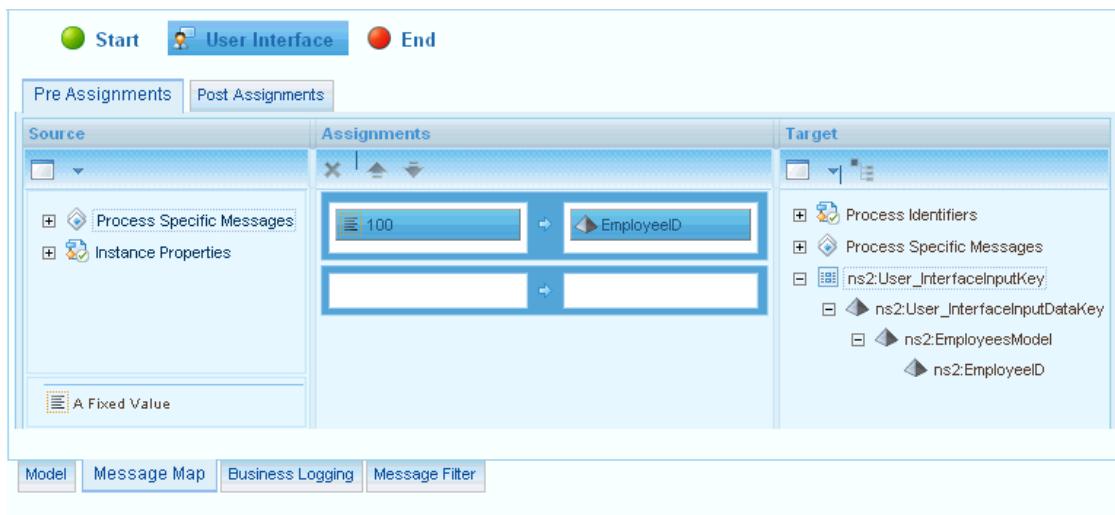
- A User Interface is created.
- A Delivery Model, with Inbox Model format is created on the User Interface.

To create a simple business process model (InboxModelBPM):

1. Drag the **User Interface** task on the business process model.
2. Right-click **User Interface** task, and select **Properties**.
The Activity - Task Properties pane appears.
3. Click the Workflow Model tab, and click to select an existing delivery model (Inbox Model format) from the **Select a Delivery Model** window and select **Use Inbox Model**.
4. Design the business process model as:



5. Complete the Message mapping as:



6. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Business Process Execution> Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no

warnings. If there are warnings resolve them and re-validate the business process model.

7. Right-click in the business process modeler and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization. The business process model is published to organization.
 8. Right-click in the business process modeler and select **Business Process Execution > Run**.
- The business process model is instantiated.
9. In My Applications palette, click  (My Inbox).
- You can see the business attribute, EmpID 100 as an additional column in My Inbox.



Activity	Process Name	Assignee	Received Date	Due date	EmpID
User Interface	DocExamples\InboxModel	None	17 Feb 2009 15:06	None	100

The EmpID 100 is displayed in My Inbox as a business attribute, using an Inbox Model in a business process model.

Using an Email model in a business process model

An Email model allows you to send customized Emails along with attachments to a group of customers. You can receive notification messages by Email. You can also initiate assigned tasks from a link within the message, containing information specific to a context, and are in a format to suit business needs.

To use an email model in a business process model:

1. [Create a delivery model using Email Model format](#).
2. [Design a business process diagram](#).
3. From **Workspace Documents <Solution> > <Project>**, drag the required <User Interface> for which you created the delivery model (in Email model format) and drop it on the required activity in the business process model.
4. Double-click the human activity (associated with User Interface on which you created the delivery model in Email model format). Alternatively, right-click the activity and select Properties. The Activity -Task Properties pane appears.
See [Activity Properties Interface](#) for details on using the Workflow Model tab.

5. Click the **Workflow Model** tab, and select <Email Model> option from the **Email Model** list.
6. Do one of the following to create an assignment:
 - Drag the source element from the **Source** column and drop it on the left box in the **Assignments** column. Likewise, drag the target element from the **Target** column and drop it on the left box in the **Assignments** column.
 - Right-click the element or message in the **Target** column, and select **Create Assignment**. By default, this creates a Fixed Value assignment.
7. Complete the **Message Map** assignments for other activities in the process model.
8. [Configure remaining constructs in the process model](#).
9. Right-click in the process modeling environment and select **Business Process Execution > Validate and generate BPML**. Alternatively, in the Workspace Documents <Solution> <Project>, right-click the <Business Process Model> and select Business Process Execution > Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings, resolve them and re-validate the process model.
10. Right-click in the business process modeling environment and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization. The business process model is published to the organization.
11. Right-click in the business process modeling environment and select **Business Process Execution > Run**.
The business process model is instantiated.

You have successfully used the Email model in a business process model.

Example of using an Email model in a business process model

The following example describes the procedure to use an Email model in a business process model to display the business attribute, Emp ID, in Email when a user receives a notification.

Business Requirements

The business process should satisfy the following requirements:

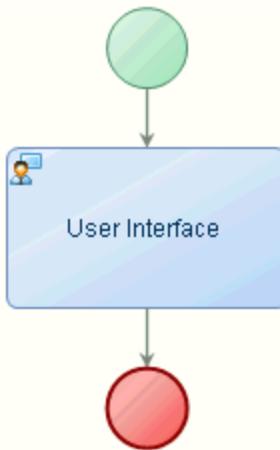
- A User Interface.
- A Delivery Model with Email Model format.
- A business process model to use the Email Model.

Ensure that:

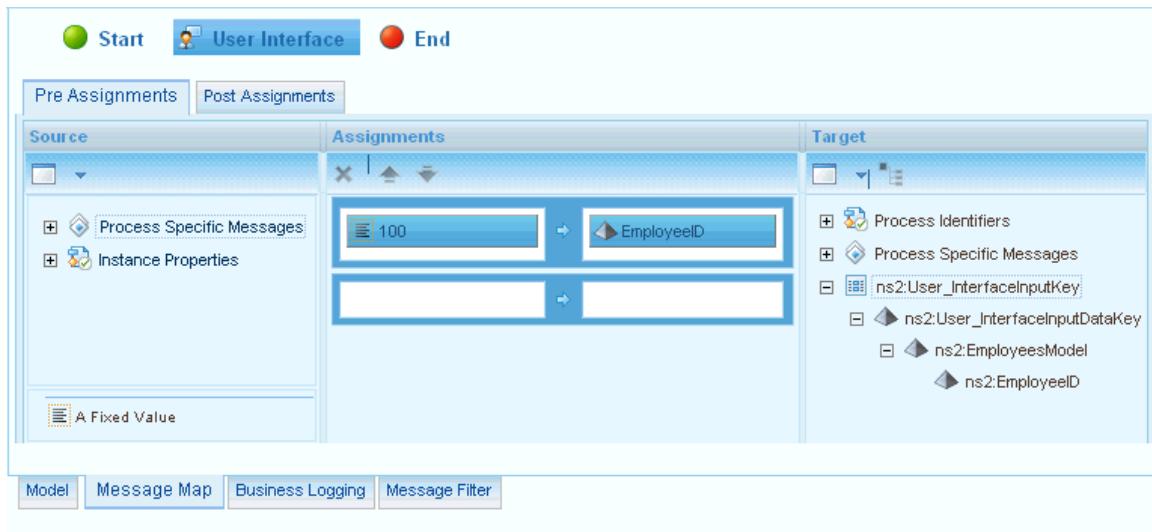
- A User Interface is created.
- A Delivery Model with Email Model format, is created on the User Interface. Ensure that you have set email notification preferences for your required email id.

To create a simple business process model, EmailBPM:

1. Drag the **User Interface** onto the required activity of the business process model.
2. Right-click **User Interface** activity and select **Properties**.
The Activity - Task Properties pane appears.
3. Click the **Workflow Model** tab and click to select an existing delivery model (Email Model format).
The Select a Delivery Model window opens.
4. Select <Email Model> from the Email Model list.
5. Design the business process model as:



6. Complete the Message mapping as:



7. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPML**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Business Process Execution >

Validate and generate BPML.

If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings resolve them and re-validate the business process model.

8. Right-click in the business process modeler and select **Business Process Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <business process model> and select Publish to Organization. The business process model is published to organization.
9. Right-click in the business process modeler and select **Business Process Execution > Run**.
10. Login to your Email account for which you have set notification preferences. You can see the Email as per the Email template that you specified.

Attaching files to an email

You can add attachments to the emails you send to your customers. For example with the email New List of Products, if you also want to send an Order Placement Details to your customers, you can use the attachment feature in the Email Template Service.

To attach files to an email:

1. To open an existing business process model, [select a starting point](#) and click  (Business Process Model).
2. Right-click the activity to which the email service is attached and select **Message Map > Message Map before Activity**.
The Message Map pane appears.
3. Expand the **Attachments** element.
4. Right-click the **Attachment** element and select **Create Assignment** from the menu.
A row is added to the value assignment table with the XPath of the element filled in the Target column.
5. Select **Replace Content With Fixed Value** in the **Operation** column and paste the content of the attachment file in the **Source** column.
6. Expand the **Attachment** element.
7. Right-click the name attribute and select **Create Assignment**.
A row is added to the value assignment table containing the XPath of the attribute in the Target column.
8. Type the name of the file in the **Source** column. The file can be in any file format. For example, Order Placement Details.doc.
9. Right-click the **encoded** attribute and select **Create Assignment**. This is a Boolean value that specifies whether the attachment is encoded or not.

10. Type one of the following in the **Source** column.
 - true: The contents of the attachment are Base64 encoded.
 - false: The attachment is sent as plain text.

11. Click .

The attachment details are saved.

The file is attached to the email.

Using human interaction in a business process model

Business processes typically comprise automated activities as well as activities that need human intervention. When you have the need to perform an activity that needs human intervention, you need to create and use the User Interface (XForm) to complete that activity.

Before you begin:

- [Create a user interface \(XForm\)](#).

To use human interaction in a business process model:

1. Do any one of the following:
 - [Select a starting point](#) and click  (Business Process Model) to open a business process model.
The business process model appears in the business process modeler.
 - If you have the business process model already opened in the business process modeler, then perform Step 2.
2. Drag required <User Interface> from the **Workspace Documents** <Solution> > <Project> content tree on to a required activity in the process model that needs human intervention.
The User Interface is associated with the activity and [a delivery model is created](#) for the User Interface.
3. Double-click the activity to which the User Interface is attached. Alternatively, right-click the activity and select Properties.
The <Activity>- Task Properties pane appears.
4. [Set the properties](#) of the User Interface activity (human activity).
5. Click .
The business process model with User Interface activity is created.
6. Right-click in the business process modeler and select **Business Process Execution** > **Validate and generate BPML**. Alternatively, in Workspace Documents > <Solution> > <Project>, right-click the business process model and select Business Process Execution > Validate and generate BPML.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings, resolve the errors and re-validate the business process model.
7. Right-click in the business process modeler and select **Business Process Execution** > **Publish to Organization**. Alternatively, in Workspace Documents > <Solution> > <Project>, right-click the business process model and select Business Process Execution > Publish to Organization.
The business process model is published to organization.
8. Right-click in the business process modeler and select **Business Process Execution** > **Run**. Alternatively, in Workspace Documents > <Solution> > <Project>, right-click the business process model and select Business Process Execution > Run.
The business process model is instantiated.

You have successfully achieved human interaction by using required User Interface in a business process model.

Example of designing a business process model with user interface

The following example describes the procedure to model a business process model to display information on a product, which can be modified and displays a non-transactional form to fill in the UnitPrice and ReorderLevel details.

Business requirements

The business process model should satisfy the following requirements:

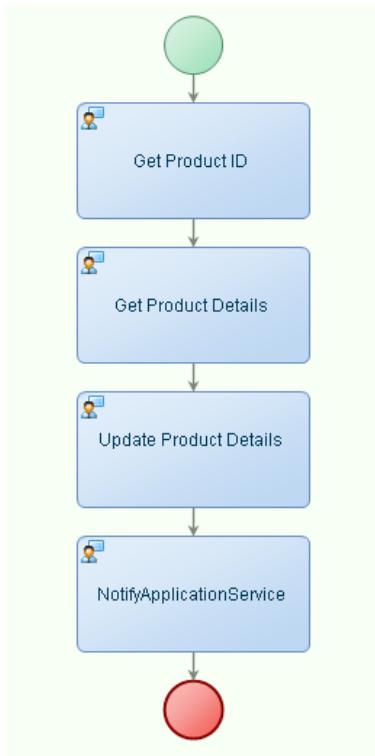
- Display an XForm to enter the Product ID.
- Display another XForm, which shows details about the product based on the information provided in Step 1. These details should be editable.
- Display a third XForm, which shows the Product ID and ProductName. The XForm should also display two other fields, UnitPrice and ReorderLevel, which are editable.
- Display a fourth XForm, notification activity to the Purchase Manager displaying the ProductID and UnitPrice of the product.

Prerequisites

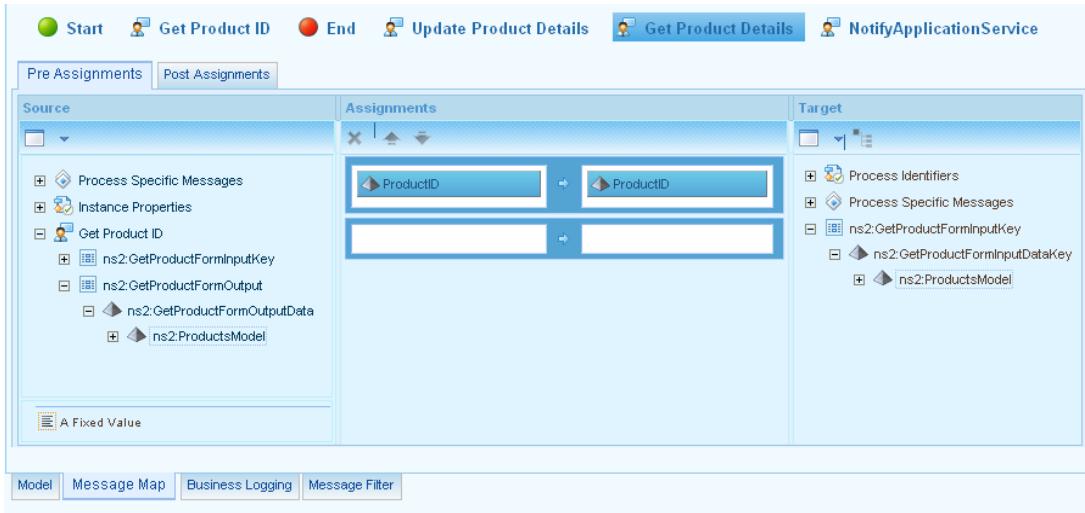
- A Get Product ID User Interface is created with Product ID as an input field.
- A Get Product Details User Interface is created to get the product details.
- An Update Product Details User Interface is created from the GetProduct method to update the product details in the database.
- A Notify Application Service User Interface is created to display the ProductID and UnitPrice of the product.
- You can use the above User Interfaces directly in a business process model as the WSDLs are already available which describe the input and output messageof the User Interface.

To design the business process model:

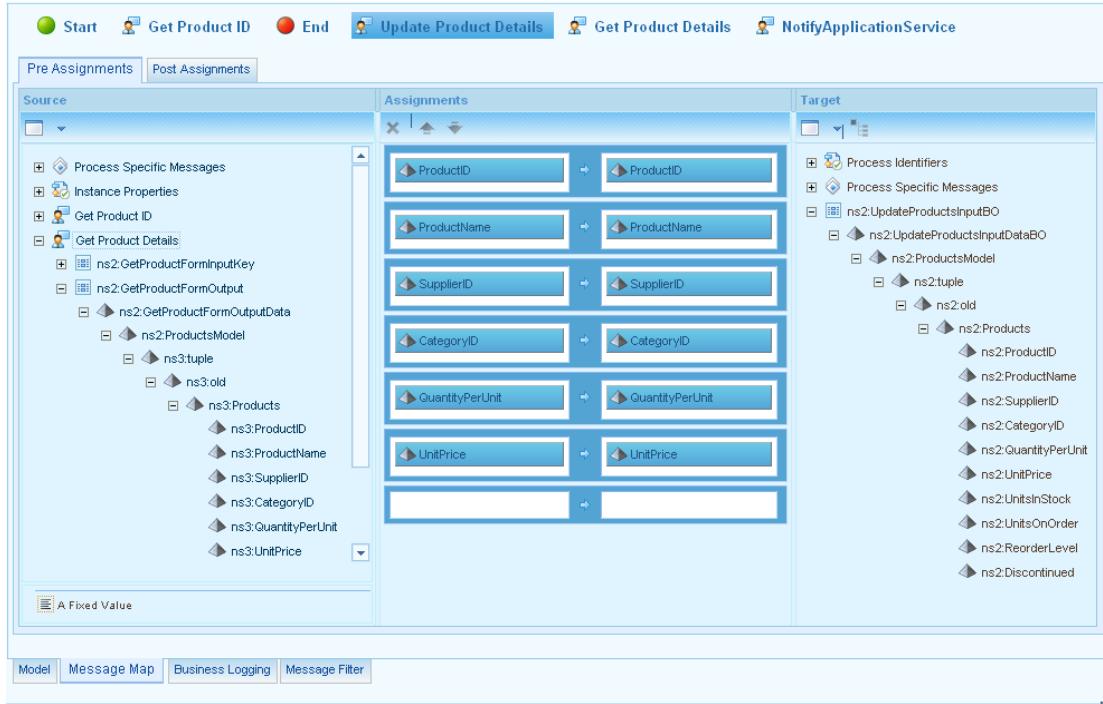
1. Drag required User Interface to create a business process.



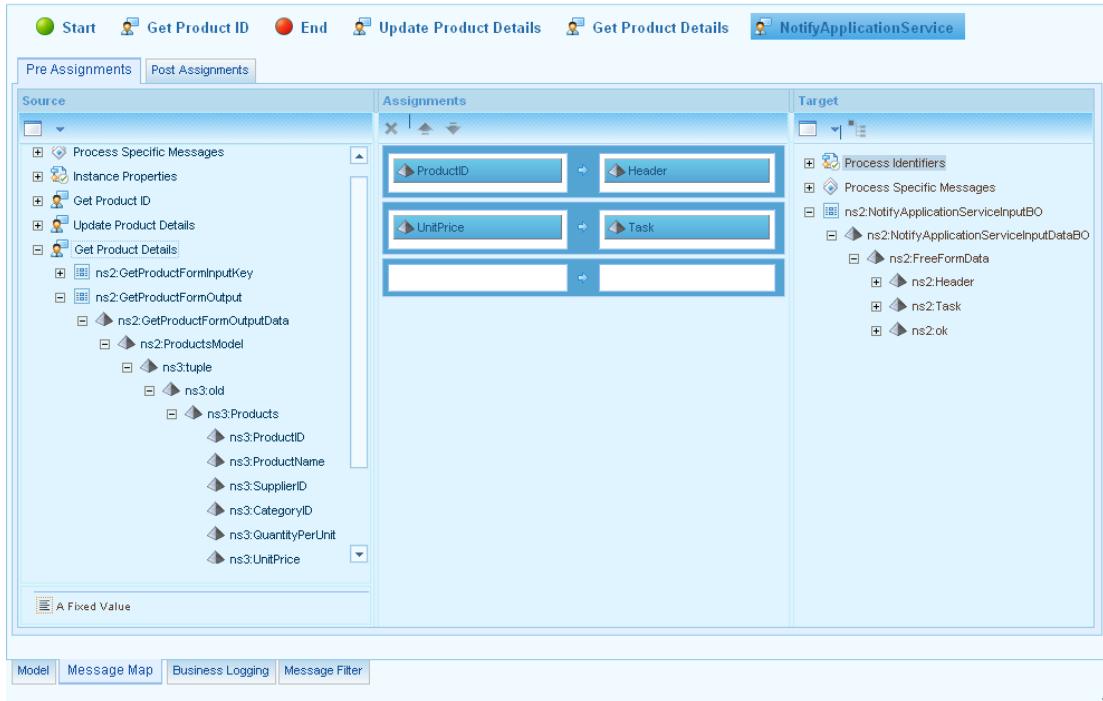
2. For the Get Product ID User Interface, map the Product ID.



3. For the Update Product Details User Interface, map ProductID, ProductName and, UnitPrice and ReorderLevel.



- For Notify Application Service User Interface, map ProductId to Header and UnitPrice to Task.



- Save and validate the process model.
- Publish the model to runtime.

7. Run the process to execute it.

Using human interaction in a case model

You can attach XForm to a case model activity when you want to call an external Web service or use it to perform a manual task.

Before you begin:

- Create a user interface (XForm)

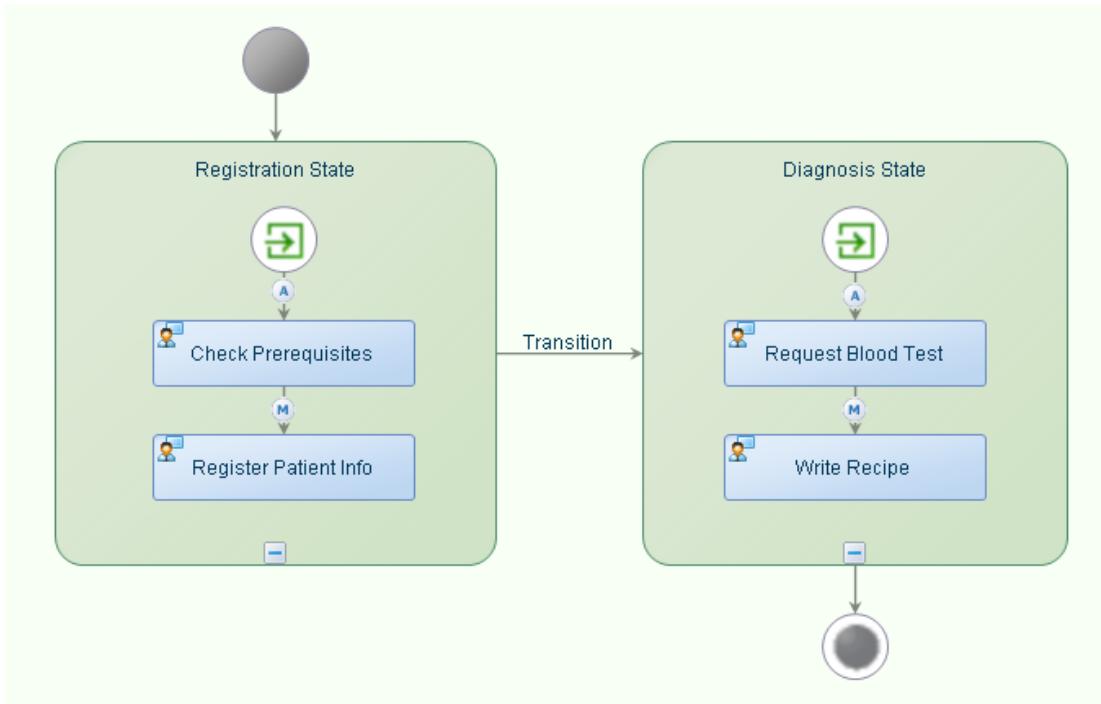
To use human interaction in a case model:

1. To open a case model, [select a starting point](#) and click  (Case Model).
The case model appears.
2. Drag required <User Interface> from the **Workspace Documents** <Solution> > <Project> content tree on to required State in the case model. Alternatively, drag the Activity construct on to the required State and then drag-drop the <User Interface> from the Workspace Documents > <Solution> > <Project> on to that Activity.
The required task User Interface is attached to the case model and a delivery model is created for the User Interface.
3. Double-click the User Interface activity in the case model. Alternatively, right-click the activity and select Properties.
The Activity Properties - New Activity pane appears.
4. [Set the properties of the User Interface/human activity](#) as required.
5. Click .
6. Right-click in the case modeler and select **Execution > Validate**. Alternatively, go to Workspace Documents > <Solution> > <Project>, right-click the case model and select Execution > Validate.
If there are no warnings, a status message appears indicating that there are no warnings. If there are warnings, resolve the errors and re-validate the case model.
7. Right-click in the case modeler and select **Execution > Publish to Organization**. Alternatively, go to Workspace Documents > <Solution> > <Project>, right-click the case model and select Execution > Publish to Organization.
The case model is published to the organization.
8. Right-click in the case modeler and select **Execution > Run**. Alternatively, go to Workspace Documents > <Solution> > <Project>, right-click the case model and select Execution > Run.
The case model is instantiated.

The required Task is used in a case model.

Sample case model using an XForm

1. Create the following tasks:
 - CheckPrerequisites
 - RegisterPatientInfo
 - RequestBloodTest
 - WriteRecipe
2. Go to **Workspace Documents** > <Solution>.
3. Right-click the <Project> and select **New** > **Other**.
The New AppWorks Platform Document dialog box opens.
4. Select **Case Management Model**.
5. [Design a case model](#) PatientRegistration as shown in the diagram to create the Initial State, Registration State, Diagnosis State, and Final State.
6. From **Workspace Documents** > <Solution> > <Project> drag and drop the created tasks corresponding to the activity names as shown in the diagram.
Required tasks are appropriately attached to the case model.
7. From the Case toolbar, drag the  (On Entry) construct and connect it to the Check Prerequisites and Request Blood Test activities in respective States as shown in the diagram.
8. From the Case toolbar, drag the **Transition** connector to show transition from one State to another as shown in the diagram.
9. [Set the properties of each User Interface/human activity in the case model](#).
10. Click .
11. Perform Steps 7 through Step 10.



Using a business rule in an XForm

AppWorks Platform supports the use of business rules to help define a business process. In a business workflow comprising multiple business processes, the business rules set for each process facilitate intra- and inter-enterprise collaboration. In a workflow implementation, the business rules determine how various business functions are structured towards the completion of the workflow.

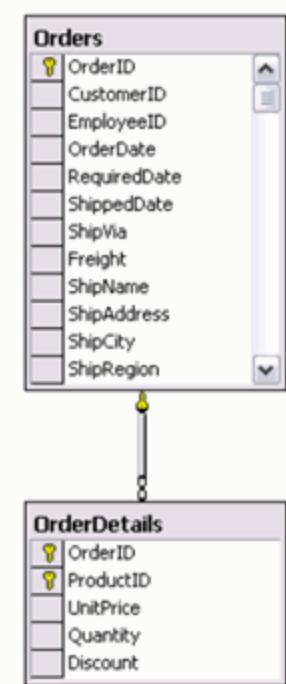
Before you begin:

- Ensure that the XForms Runtime, and WS-AppServer application packages are installed.

Business rules apply on Business Objects and comprise the logic according to which a business object must function in a given situation. Using WS-AppServer, you can apply business rules to the business objects in an application created using AppWorks Platform XForms. The integration of XForms with WS-AppServer enables the execution of these rules in an XForm.

For example, consider a business scenario in which it is required that an order placed by a customer must not exceed 100 units. In this case, a rule needs to be defined to reject additional units, after 100 units are already selected for an order. Additionally, a notification needs to be sent to the customer specifying these order details. For information about business rules applicable in AppWorks Platform, see [Rules](#).

For this example, we use the Orders and Order Details tables of the Northwind database (MS-SQL Server). The following figure depicts the schematic representation of the tables and their relationship.



To use a business rule in an XForm:

1. Configure the WS-AppServer service container to point to the appropriate database (in this case, the Northwind database).
See [WS-AppServer service container configuration](#) for details.
Also, ensure that the related packages, Java Classes, and Web service interfaces exist or are created for the Orders and Order Details table. These classes contain basic implementation for the generated Web service interfaces. For details, see [Extending the Business Logic using Custom Classes](#).
2. [Enable WS-AppServer integration with XForms](#) by selecting the WS-AppServer Integration option in the Model Properties dialog box.
3. Use the Rule Editor to create a rule:
 - a. Select the package for which to create the rule from the WS-AppServer Package window (navigate to Workspace Documents (Explorer) > <solution> > <project> and click), and provide appropriate details.
 - b. Right-click a model and select **Create Rule**.
The Rule Editor appears.
 - c. In the **Properties** tab, enter the appropriate details.
For information about the options available in the Properties tab, see [properties of a rule](#).
 - d. In the Behavior tab, click in the Rule Definition pane to define the condition.
The Set Condition pane appears.

- e. From the **Business Objects** pane, drag the **Quantity** attribute of the **OrderDetails** object to the **Set Condition** pane and build the required expression using the functions in Function Library.
 - f. Click **Add**.
The specified condition is displayed next to  in the Rule Definition pane.
 - g. In the Rule Definition pane, right-click **then** and select **Send Notification**.
The Action - Send Notification pane appears.
 - h. [Specify the appropriate details](#) for the notification message and click **Add**.
The defined action is displayed next to  in the Rule Definition pane.
 - i. In the Rule Definition pane, right-click **then** and select **Abort Transaction**.
The Action - Abort Order pane appears.
 - j. [Specify the appropriate details](#) for the notification message and click **Add**.
The defined action is displayed next to  in the Rule Definition pane.
 - k. Click  to save the rule.
4. [Create the XForm](#) using the base Web service operation and associate controls to the appropriate Web service operations.
In this case, drag the GetOrdersObjects and GetOrderDetailsObjectsByOrderID Web service operations from the Insert tab to create the XForm, and associate them to get the right data. For information about associating Web service operations, see [Adding and Associating Multiple Models](#).
- By default, a SOAP fault is displayed when you enter data for a new order, or change an existing order quantity to more than 100. Use the following code to display a custom message instead.

```
if (eventObject.faultString.indexOf("Order can not exceed 100 units") > 0) {
    application.showErrorMessage("Order aborted. Order units can not exceed 100. See mail in
your Inbox for more details.");
    eventObject.showError = false;
    OrderDetailsModel.undo();
}
```

Now you can use the business logic in the XForm.

Using WS-AppServer business logic in XForms

Using the WS-AppServer, you can auto-generate and extend the required business logic for web applications. Additionally, AppWorks Platform supports the integration of WS-AppServer with XForms such that, based on the conditions specified in it, the business logic can be directly executed on data in an XForm.

Before you begin:

- Ensure that the XForms Runtime and WS-AppServer application packages are installed.

You can integrate the WS-AppServer business logic with an XForm to achieve any of the following:

- Define the Enable, Disable, Hide, and Show access modes and use them for controls.
- Enable the auto-initialization of new data inserted in an XForm.
- Specify validation constraints to validate changes made to data.
- Enable the automatic pre filling of Select controls with data based on the business logic.

Consider the possible relationships between the Orders table and the Employees and Customers tables of the Northwind database (MS-SQL Server).

To use WS-AppServer business logic in XForms:

1. Configure the WS-AppServer service container to point to the appropriate database (in this case, the Northwind database).
See [WS-AppServer service container configuration](#), for details.
Also, ensure that the related packages, Java Classes, and Web service interfaces exist or are created for the Orders table.
2. [Create the XForm](#).
 - a. From the Insert tab of the XForms Designer, drag the appropriate Web service operation (in this case, GetOrdersObject) to the XForm to generate the default user interface.
 - b. Alternately, you can right-click in the Designer Area and select Insert > Web Service to view available Web services and add the required Web service to the XForm.
3. Enable WS-AppServer integration with XForms in order to access and utilize the features offered by the integration.
4. Write business logic on the extended Java class (in this case, Orders.java).
5. Define access modes.

In this case, access modes with the following logic need to be created for fields in the Orders table.

Freight	* Hide Freight if its value is less than ten. * Show Freight for new orders.
OrderID	* Hide OrderID for new orders * Disable OrderID for existing orders

To achieve this functionality, the necessary supporting packages need to be imported as follows:

```
import com.cordys.cpc.bsf.event.AccessMode;
import com.cordys.cpc.bsf.event.AttributeAccessEvent;
```

The java code for the OrderIDfield is as follows:

```

public void onDisplay_OrderID(AttributeAccessEvent context)
{
    if(isNewObject())
    {
        context.setAccess(AccessMode.HIDE);
    }
    else
    {
        context.setAccess(AccessMode.READONLY);
    }
}

```

In this case:

- the OnDisplay keyword is used to specify and set the access modes.
- the context event object applies the logic to the current order.
- the isNewObject() Web service operation is used to indicate if the current order object is new or if it already exists.

```

private boolean isNewObject()
{
    //an Order object is new if its OrderID is 0
    return this.getOrderID() == 0;
}

```

Similarly, the java code for the Freight field is as follows:

```

public void onDisplay_Freight(AttributeAccessEvent context)
{
    //hide the field for existing objects, where Freight < 10
    if(! isNewObject() && this.getFreight() < 10)
    {
        context.setAccess(AccessMode.HIDE);
    }
    //else: default is ReadWrite
}

```

Ensure that **Apply Access Control** in the WS-AppServer tab of the Model Properties dialog box of the XForm is selected.

6. Enable the auto-initialization of new data.

In this case, define the following auto-initialization.

OrderDate	Present date. Specify for a new order.
RequiredDate	The date one week later than the order date of a new order.

To achieve this, add the following packages:

```
import com.cordys.cpc.bsf.event.AttributeInitializeEvent;
import com.cordys.cpc.bsf.util.DataConverter;
import java.util.Date;
```

The code to auto-initialize data is as follows:

```
public void onInitialize_OrderDate(AttributeInitializeEvent context)
{
    context.setInitialValue(DataConverter.Date2String(new Date()));
}
public void onInitialize_RequiredDate(AttributeInitializeEvent context)
{
    //calculate the date for today+1 week
    Date today = new Date();
    long oneWeekMSec = 7*24*3600*1000;
    Date plusOneWeek = new Date(today.getTime() + oneWeekMSec);
    //the initial value is passed via the context
    context.setInitialValue(DataConverter.Date2String(plusOneWeek));
}
```

To be able to auto-initialize data when new orders are added, ensure that the Initialization Required check box in the WS-AppServer tab of the Model Properties dialog box of the XForm is selected.

7. Specify validation constraints.

In this case, the following validation constraints are required.

CustomerID	Required. Should always be present in the Customers table.
OrderDate	Cannot specify a date that is earlier than the present date.

To achieve this, the following package needs to be added:

```
import com.cordys.cpc.bsf.event.AttributeConstraintEvent;
import java.util.Calendar;
```

The code to set the specific constraints is as follows:

```
public void onConstraint_CustomerID(AttributeConstraintEvent context)
{
    //check the available Customers whether the Customer exists
    //the new CustomerID is passed via the context
    String customerID = (String)context.getValue();
    if (Customers.getCustomersObject(customerID) == null)
    {
```

```

        context.addError(this, ATTR_CustomerID, "Customer " + customerID + " does not
exist !!!");
    }
}
public void onConstraint_OrderDate(AttributeConstraintEvent context)
{
    Date objectDate = this.getOrderDate();
    if(objectDate != null)
    {
        Calendar today = Calendar.getInstance();
        today.clear(Calendar.HOUR);
        today.clear(Calendar.MINUTE);
        today.clear(Calendar.SECOND);
        today.clear(Calendar.MILLISECOND);
        Calendar currentDate = Calendar.getInstance();
        currentDate.setTime(objectDate);
        currentDate.clear(Calendar.HOUR);
        currentDate.clear(Calendar.MINUTE);
        currentDate.clear(Calendar.SECOND);
        currentDate.clear(Calendar.MILLISECOND);
        if (this.isPersistentNew() && currentDate.compareTo(today) < 0)
        {
            context.addError(this, ATTR_OrderDate, "Date cannot be less than today!");
        }
    }
}

```

In the code, we check if CustomerID already exists in the Customers table. After all related table classes are created by the WS-AppServer (by default), the getCustomersObject() Web service operation is used to retrieve the customer ID.

- To enable the constraint validations, ensure that the Constraint Validation checkbox in the WS-AppServer tab of the Model Properties dialog box of the XForm is selected.
- Also, ensure that the property is enabled for each control for which the constraints need to be evaluated. So, the Constraint Validation On Change check box needs to be checked in the CustomerID and OrderDate property sheets.

9. Enable the automatic pre-filling of the Select control.

You can pass multiple data values from a database to a Select control that display as options in the control. This ensures that only the possible, valid values are displayed at run time, which can be selected from the list of the Select control.

In this case, the values from the database need to be pre-filled in the EmployeeID field:

Field	Validation	Description
EmployeeID	Choice Values	Employee will comprises the FirstName and LastName of the employee

To achieve this, add the following packages:

```
import com.cordys.cpc.bsf.event.AttributeValuesEvent;
import com.cordys.cpc.bsf.busobject.BusObjectIterator;
```

The following code is used to iterate through employee records, combine the first and last names for each record, and add it to the event object that is rendered in the XForm.

```
public void onValues_EmployeeID(AttributeValuesEvent context)
{
    //Get all Employees
    BusObjectIterator<Employees> iter = Employees.getEmployeesObjects(0, 999);
    while (iter.hasMoreElements())
    {
        Employees c = (Employees)iter.nextElement();
        //filter out the valid ones
        String value = ""+c.getEmployeeID();
        String label = c.getFirstName() + " " + c.getLastName();
        //add value to the context
        context.addValue(value, label);
    }
}
```

You can also use a Select control instead of the default EmployeeID input control that is generated on the Orders table of an XForm. Here, instead of setting a Dataset for the Select control, the Initialize Value Sets WS-AppServer property must be set to Once or Always.

Note: Steps 4, 5, 6, and 7 are optional and can be used to achieve specific requirements.

Using an automatic follow-up connector in a case model

When you want to ensure that case activities are performed automatically one after another in a sequential order, you need to use the automatic follow-up connector.

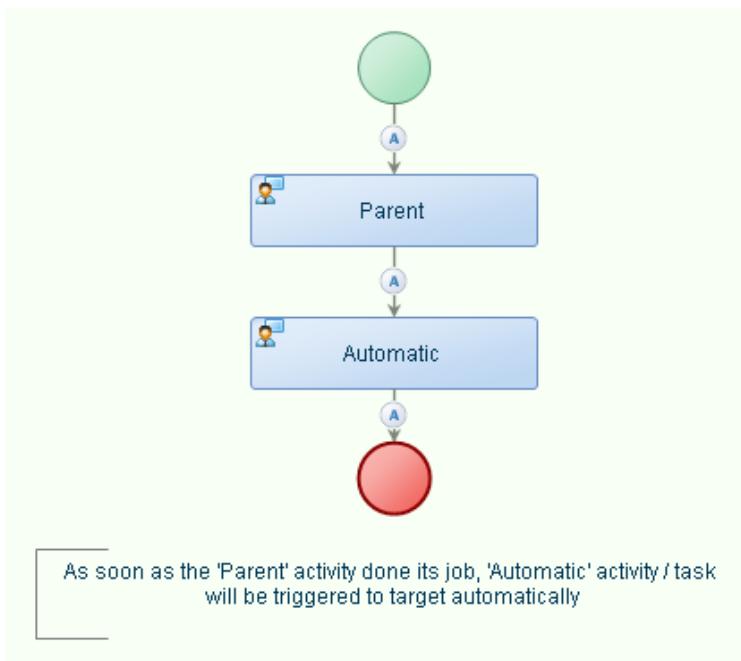
To use an automatic follow-up connector in a case model:

1. **Select a starting point** and click  (Case Management Model) to create a case model. The case model appears in case modeler.
2. To handle the start of a case, drag  from the toolbox to the case modeler to create a case model as shown in the example.
3. Drag  from the toolbox to the case modeler. If the toolbox is not visible in the modeler, click  (Options) and select Toolbox.
4. Repeat Step 3 to drag another  from the toolbox to the case modeler.
5. Drag  from the toolbox to the modeling environment.

6. To link the constructs click  on the toolbar and select each construct in the case modeler. Alternatively, hold the CTRL key and select the constructs in the case modeler with the mouse.
All the constructs of the case model are now linked.
7. Click  to save the case model.

You have successfully used the automatic follow-up connector in a case model.

Sample snapshot of a case model using the automatic follow-up connector



After you complete this task:

- Publish the case model.

Using free follow-up in a case model

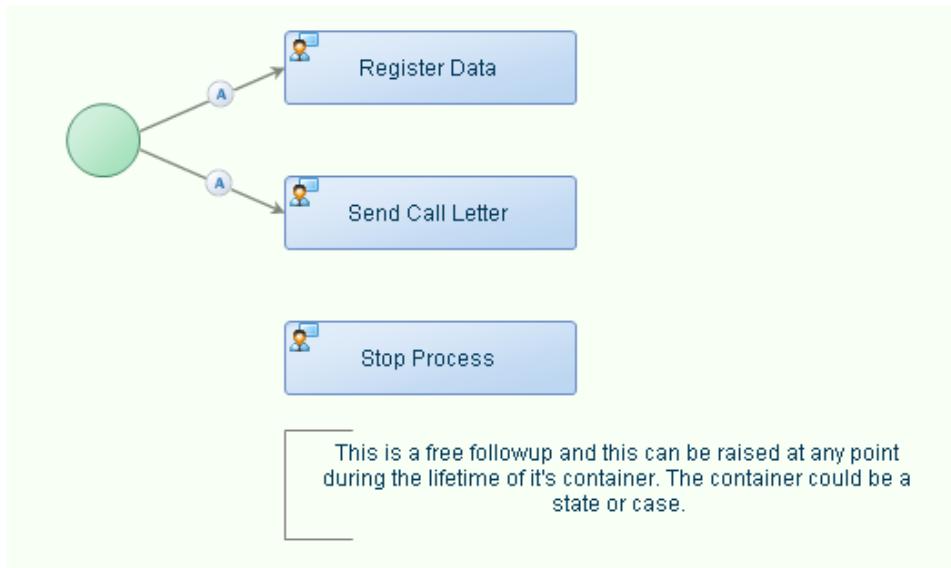
To use free follow-up in a case model:

1. Select a starting point and click  (Case Management Model) to create a case model.
The case model appears in case modeler.
2. To handle the start of a case, drag  from the toolbox to the case modeler to create a case model as shown in the example below.
3. Drag  from the toolbox to the case modeler. If the toolbox is not visible in the modeler, click  (Options) and select Toolbox.
4. Repeat Step 3 as required to drag another  from the toolbox to the case modeler.

5. To link the activities to , click  on the toolbar and select Register Data and Send Call Letter activities in the case modeler. Alternatively, hold the CTRL key and select the required activities in the case modeler with the mouse.
Selected activities of the case model are now linked with Start Case construct.
6. Ensure that the Stop Process activity is not connected to any construct as shown in the following example.
7. Click  to save the case model.

You have successfully used the free follow-up in a case model.

Sample snapshot of a case model using the free follow-up



After you complete this task:

- Publish the case model.

Using a document received event in a case model

When you want to ensure that relevant case documents are acted upon when the case activity is in a specific state, use the Document Received Event.

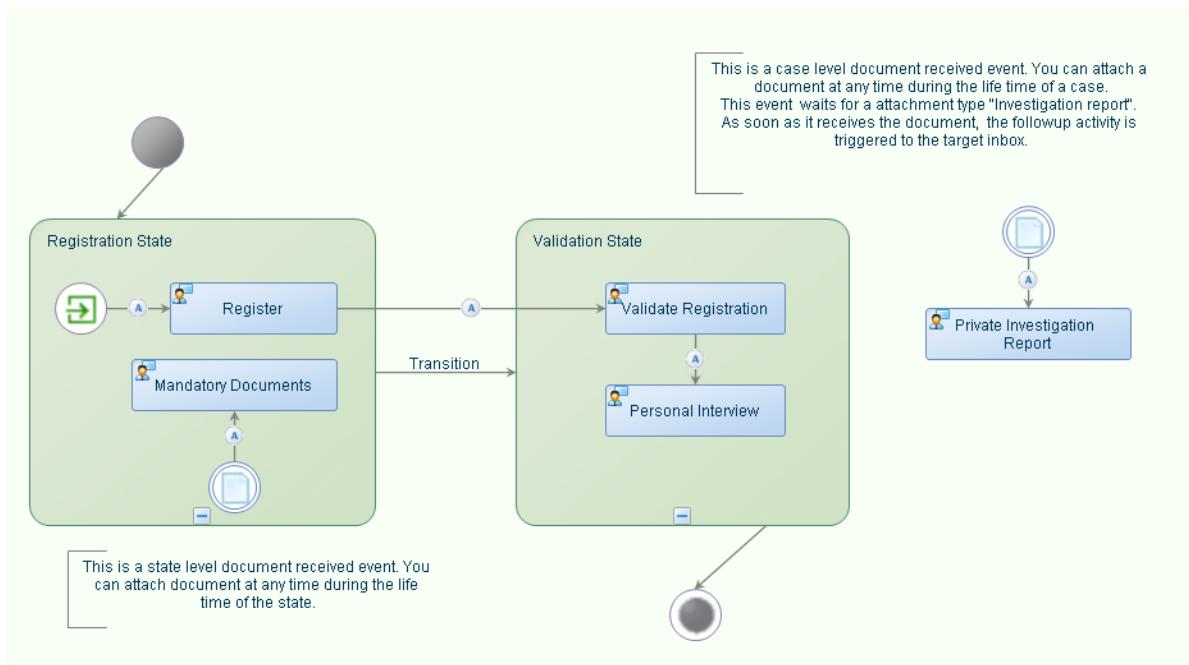
To use a document received event in a case model:

1. Select a starting point and click  (Case Management Model) to create a case model. The case model appears in case modeler.
2. To handle the start of a state within the case, drag  (Initial State) from the toolbox to the case modeler to create a case model as shown in the example.
3. Drag  (State) from the toolbox to the case modeler to create a state as shown in the example.

4. Create Registration State and Validation State as shown in the example.
A state is created.
 5. Drag  from the toolbox to the Registration State as shown in the example.
 6. Drag  from the toolbox to within the state.
 7. Create the following activities as shown in the example:
 - Registration
 - Mandatory Documents
 - Validate Registration
 - Personal Interview
 - Private Investigation Report
- If the toolbox is not visible in the modeler, click  (Options) and select Toolbox.
8. **Create (User Interface)** one each for Registration, Mandatory Documents, Validate Registration, Personal Interview and Private Investigation Report case activities.
 9. From **Workspace Documents (Explorer) > Solution > Project**, drag the relevant Interface one by one on to the case activities.
 10. Drag  (Document Received Event) from the toolbox to the Registration State and at the case level.
 11. Drag  (Final State) from the toolbox to the Validation State as shown in the example.
 12. To link the constructs click  on the toolbar and select each construct in the case modeler. Alternatively, hold the CTRL key and select the constructs in the case modeler with the mouse.
All the constructs of the case model are now linked.
 13. Click  to save the case model.

You have successfully used the Document Received Event in a case model.

Sample snapshot of a case model using the document received event



After you complete this task:

- Publish the case model.

Using an intermediate follow-up connector in a case model

When you want a dependent activity to complete prior to completion of the main activity, you can use the Intermediate Follow-up connector. Select this connector when you have linked activity or activities that have to be completed only after which the current activity can resume to completion. For example, while Activity A is active, the case worker can release Activity B and Activity A is automatically suspended. Activity A is automatically resumed only after completion of Activity B.

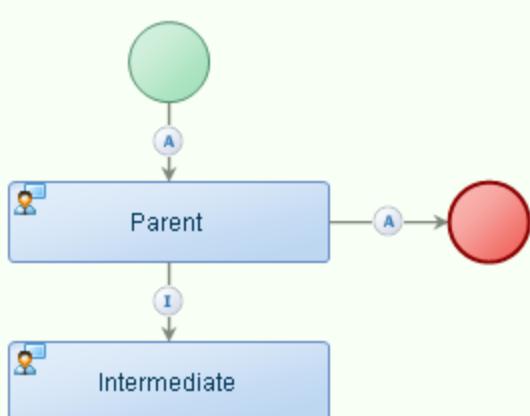
To use an intermediate follow-up connector in a case model:

1. Select a starting point and click (Case Management Model) to create a case model. The case model appears in case modeler.
2. To handle the start of a case, drag from the toolbox to the case modeler to create a case model as shown in the example.
3. Drag from the toolbox to the case modeler. If the toolbox is not visible in the modeler, click (Options) and select Toolbox.
4. Create (User Interface) one each for Parent and Intermediate case activities.

5. Repeat Step 3 to drag another  from the toolbox to the case modeler.
6. To link the constructs Start event and End event to the Parent activity, click  on the toolbar and select each construct in the case modeler. Alternatively, press and hold the CTRL key and select the constructs in the case modeler with the mouse.
All the constructs of the case model are now linked.
7. To link the Parent and Intermediate case activities, click  (Intermediate Follow-up) connector on the toolbar and select each case activity in the case modeler. Alternatively, hold the CTRL key and select the case activities in the case modeler with the mouse.
The Parent and Intermediate activities of the case model are now linked with Intermediate Follow-up connector.
8. Drag  from the toolbox to the modeling environment.
9. Click  to save the case model.

You have successfully used the Intermediate Follow-up connector in a case model.

Sample snapshot of a case model using the intermediate follow-up connector



Here, the parent activity triggers the intermediate follow-up. Till the intermediate activity is complete, the parent activity will not appear in the Inbox. This also means that the parent activity is completed only after the intermediate task is completed.

After you complete this task:

- Publish the case model

Using the applicability service in a case model

While working on a Case model, you may need to take decisions to choose the most applicable sub-set from next set of follow-up activities, based on the context and data available for a Case. In such scenario, it becomes necessary to consider all the possible options and exceptions available for executing a task.

The Applicability Service is a feature that helps you to accomplish tasks easily, while considering all the possible options and exceptions by implementing the required business logic on a Web service. When the Applicability Service feature is selected for the State, it becomes applicable for all the activities within the State. When the activities within the State are executed, based on the work option or exception the Applicability Service is triggered and recommendations are made for the set of applicable follow-up activities.

Recommendation types

The recommendation for a follow-up activity can be of the following types:

isRecommended	Refers to an activity recommended for execution. For example, if a customer's credit history is poor, an optional activity of verifying the previous banking transactions must be performed.
isNotRecommended	Refers to an activity that is not recommended for execution. However, the activity can still be manually executed, based on the discretion of the case worker. For example, if a patient's medical condition is clear, further diagnostic checks may not be required.
isWarning	Refers to an activity that can be executed but with a bit of caution as it may have negative results. For example, if a patient's medical history mentions an allergy to anesthesia, providing it may risk the patient's condition.

Implementing applicability service

The applicability logic can be implemented in the form of a Web service or a Business Process Model (BPM) directly. When Web service is considered, it can be implemented using a Java class, a decision table, or a BPM. This topic explains the process of implementing the Applicability logic using a Web service generated from a BPM.

Scenario

Consider a Case model that is designed to handle the situation of providing facilities for passengers of the trains that get delayed due to technical reasons. In such a situation, the administration needs to provide facilities to the train passengers based on the current condition in a station. A case model containing a set of defined activities handles the process.

The business logic for this situation is defined based on the parameters - Temperature and Duration of the delay. Typically, this information is captured in the Case model in the form of Case Data. Based on this information, the evaluation of applicability for the follow up activities is done.

The following table explains the parameter values and the applicable activities.

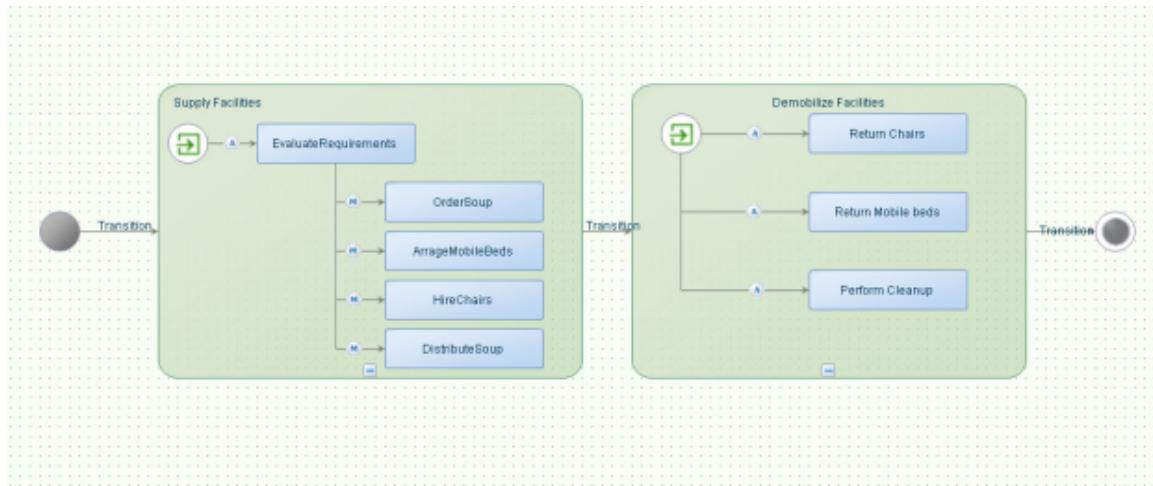
		Situation A	Situation B	Situation C
Conditions	Temperature (deg Celcius)	< 5		

		Situation A	Situation B	Situation C
	Delay (minutes)	> 120	90-120	<=90
Activities	OrderSoup	Recommended	NotRecommended	NotRecommended
	DistributeSoup	Recommended	NotRecommended	NotRecommended
	ArrangeMobileBeds	Recommended	Warning	Warning
	HireChairs	NotRecommended	Recommended	Warning

In the table, Situation A states that if the temperature is less than 5 and the delay is more than 120, the recommended activities are OrderSoup, DistributeSoup, and ArrangeMobileBeds. The activity HireChairs is not recommended.

The Case management model below handles the above scenario. It has two states Supply Facilities and Demobilize Facilities. We will define and apply the Applicability Service to the Supply Facilities state, and will implement the Applicability Service using a BPM. In the BPM, the business logic will be implemented using a decision table.

Case Management Model



To build the Applicability Service:

1. Create the Case data schema fragment.
2. Create the decision table schema fragment.
3. Create the Applicability Service using contract.
4. Implement the Web service using a BPM.
5. Create the decision table.
6. Update the BPM.
7. Build the Message Map assignments.
8. Configure the Applicability Service in the Case model.
9. Execute the Case model.

Step 1: Create the case data schema fragment

- Create an XML Schema document to define the Case data with below instance XML and name it SupplyFacilitiesData.

```
<ns:CaseData xmlns:ns="http://schemas.applicability.demo/1.0">
<ns:Temperature>string</ns:Temperature>
<ns:ExpectedDelay>string</ns:ExpectedDelay>
</ns:CaseData>
```

Step 2: Create the decision table schema fragment

- Create an XML Schema document to define the facilities data with the following XML instance and name it FacilitiesEvaluationSchema.

```
<ns:FacilitiesData xmlns:ns="http://schemas.applicability.com/implementation/1.0">
  <ns:CaseData>
    <ns:Temperature>string</ns:Temperature>
    <ns:ExpectedDelay>string</ns:ExpectedDelay>
  </ns:CaseData>
  <ns:Activities>
    <ns:OrderSoup>string</ns:OrderSoup>
    <ns:DistributeSoup>string</ns:DistributeSoup>
    <ns:ArrageMobileBeds>string</ns:ArrageMobileBeds>
    <ns:HireChairs>string</ns:HireChairs>
  </ns:Activities>
</ns:FacilitiesData>
```

Step 3: Create the Applicability Service using contract

1. Copy the latest WSDL contract to the development server.
2. If not already present, you need to create a UDDI Service Container.
3. Create a new Web service using the above contract.
 - a. Right-click a folder and select New -> Web Service.
 - b. In **Select the source** control, select **Import WSDL**.
 - c. Specify **Name** and **Description**, and click **Next**.
 - d. In the URL input field, provide the URL of the WSDL copied to the development server (in Step 1).
 - e. Select the Web service operation and click **Finish**.

Step 4: Implement the Web service using a BPM

1. Right-click a folder and select **New > Business process model**.
2. Right-click the editor of the BPM and select **Properties**.
3. Select **Contract**.
4. Click  to select the Web service created in Step 3.
5. Save the BPM.

This generates a skeleton of the BPM with the defined contract. You need to implement this logic in the BPM. For this example, we will create a decision table to define our logic.

Step 5: Create the decision table

1. Right-click a folder and select **New > Rule Group**.
2. Provide a name and description and save the Rule Group.
3. Right-click the **Rule Group** and select **Add > Decision Table**.
4. On the left-hand pane of the Decision Table editor, click  to open the FacilitiesData schema fragment that was created in the Step 2.
The schema fragment is available under the FacilitiesEvaluationSchema XML Schema document.
5. Model the [Building a decision table](#) as shown.

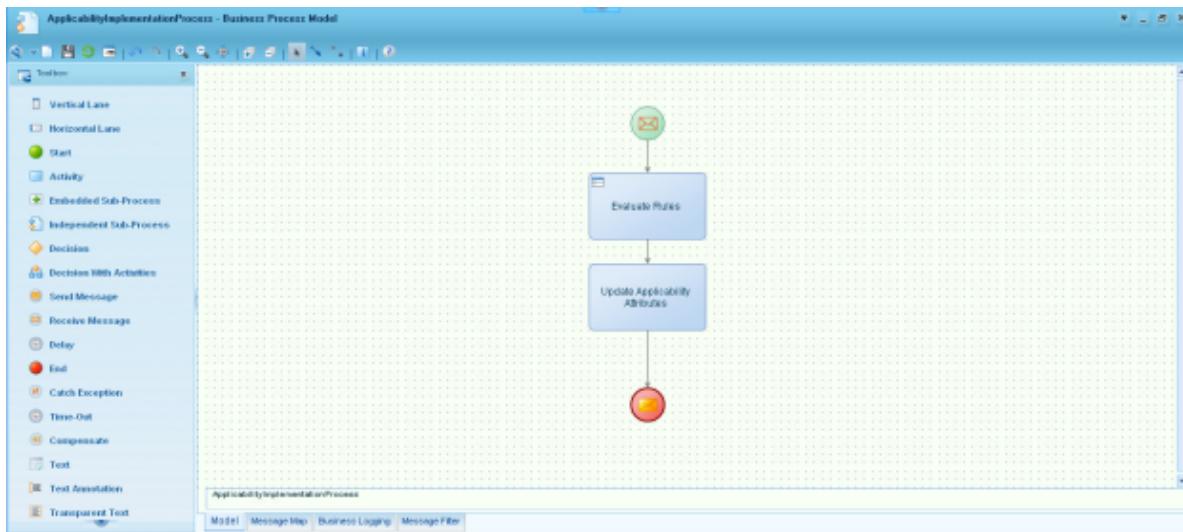
	Rule 1	Rule 2	Rule 3
OrderSoup	Recommended	NotRecommended	NotRecommended
DishreteSoup	Recommended	NotRecommended	NotRecommended
ArrageMobileBeds	Recommended	Warning	Warning
HireChair	NotRecommended	Recommended	Warning

- The assignment values for the activities are different with labels. This is important because these values will be used during evaluation of activities. For example: Value of OrderSoup element in Rule1 is set to isRecommended and the label is defined as Recommended. Refer to Recommendation Types section above for more information.

- The elements in FacilitiesData schema fragment of FacilitiesEvaluationSchema XML Schema are defined based on the names of the Case activities.

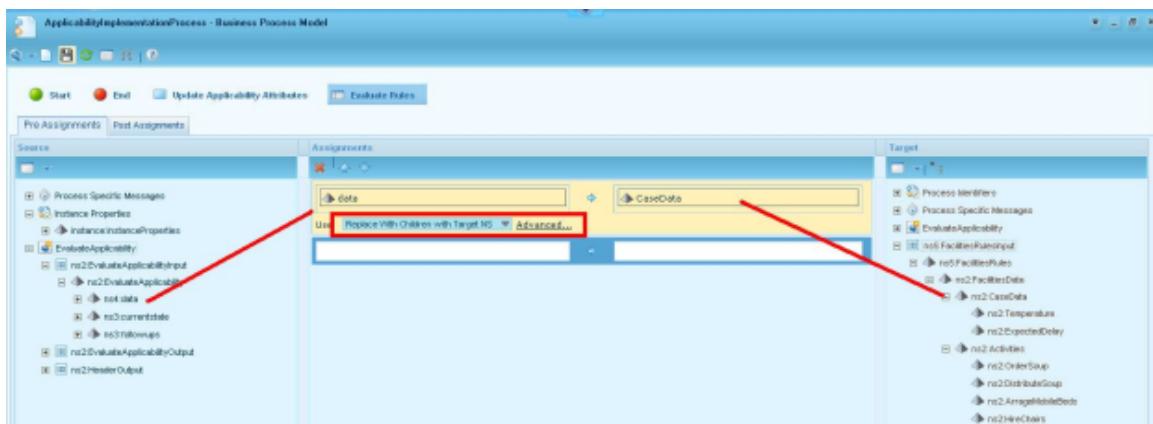
Step 6: Update the BPM

- Access the BPM created in Step 4 and drag the decision table onto the empty activity. This will replace the empty activity with the decision table activity.
- Add a new empty activity as shown.

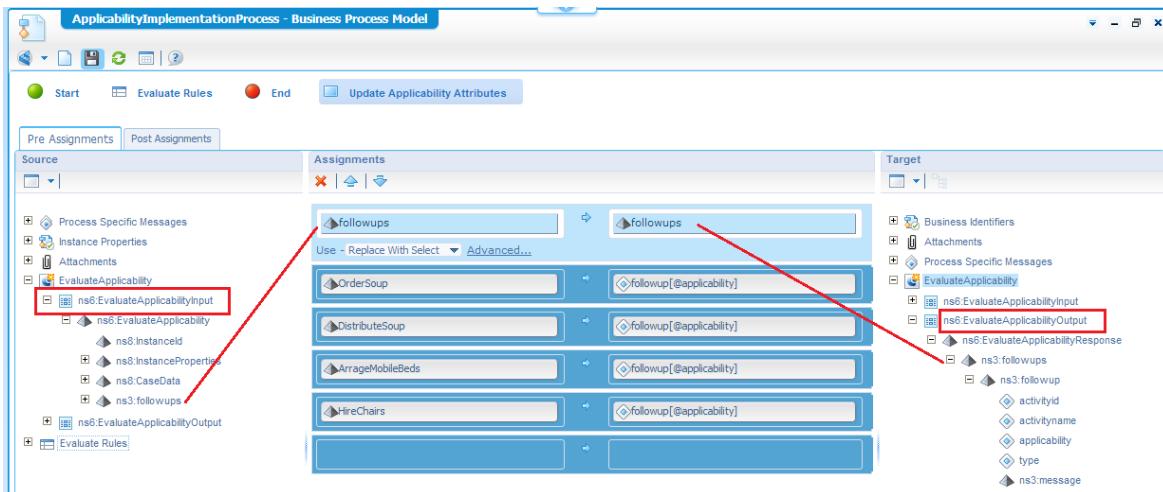


Step 7: Build the Message Map assignments

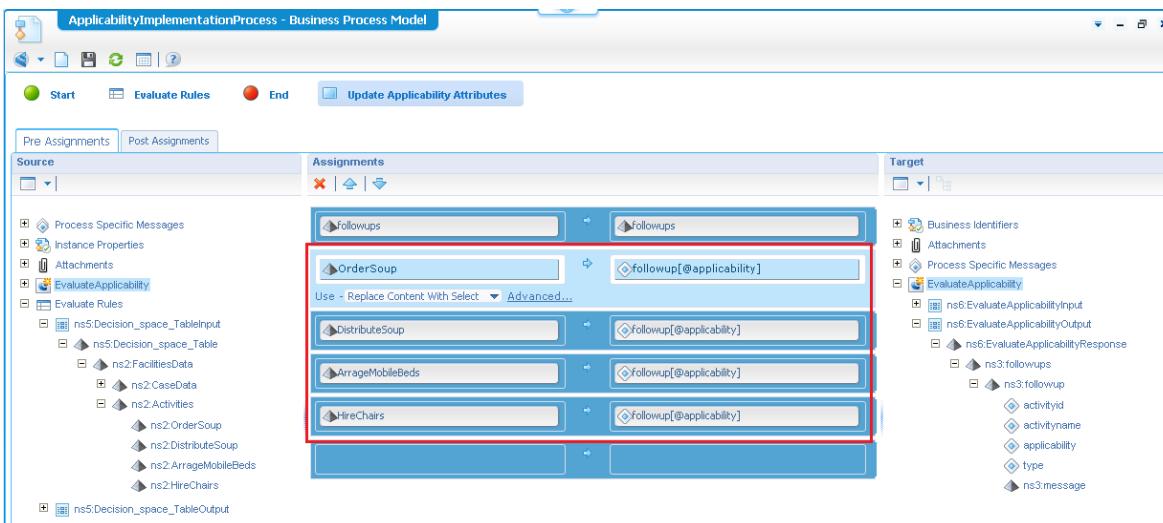
- Select Evaluate Rules decision table by clicking on the activity and go to the Message Map tab.
- Set data element as source for the target CaseData element of FacilitiesRulesInput Message as shown.



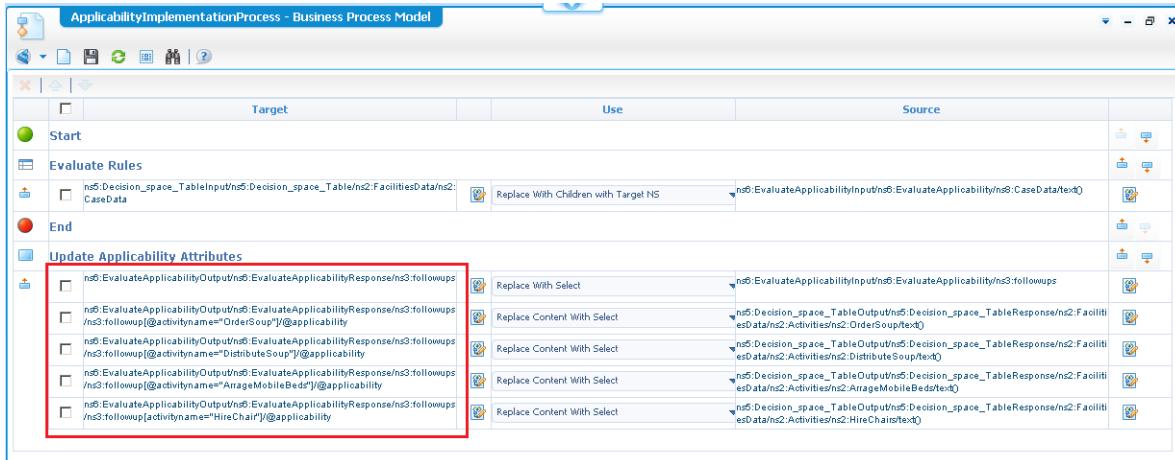
3. Change the assignment operation to **Replace With Children with Target NS**.
4. Click the **Update Applicability Attributes** activity in the activities artifact viewer.
5. Replace the followups element of **EvaluateApplicabilityOutput** message with the followups of element of **EvaluateApplicabilityInput** message.



6. Modify the assignment operation to **Replace With Select**.
7. Map the activity **OrderSoup** element value from **FacilitiesRulesOutput** message to applicability attribute of **EvaluateApplicabilityOutput** message.
8. Modify the assignment operation to **Replace Content with Select**.
9. Repeat Step 6 for all activity elements.



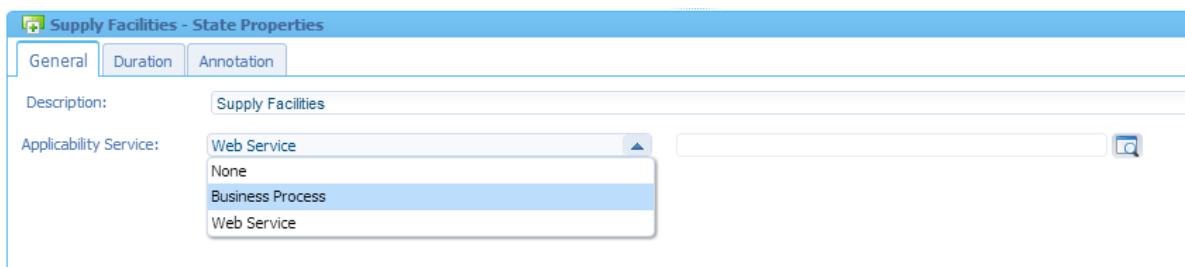
10. Click **Switch to consolidated view** on the toolbar to switch to the consolidated view.
11. Update all the target assignments in target column with expressions as displayed below.



- Save the business process model.

Step 8: Configure the Applicability Service in the case model

- Create a Case model.
- Right-click the model and select **Properties** for the properties view.
- Click the **Data** tab.
- Select **Use XML Schema Fragment (XSD)**.
- Use look-up to open the CaseData schema fragment of the SupplyFacilitiesData XSD created in Step 1.
- Right-click the **Supply Facilities** state and select **Properties** for its properties view.



- Select the type of **Applicability Service** that you want to configure.
- Click and select the related Web Service created in Step 3 or Business Processes in Step 5 according to the option chosen.
- Save the Case model.

Step 9: Execute the case model

- Start the **Rule repository** Service group.
- Right-click the project and select **Publish to organization**.

3. Attach the Web service interface created in Step 3 to the BPM Engine Service group.

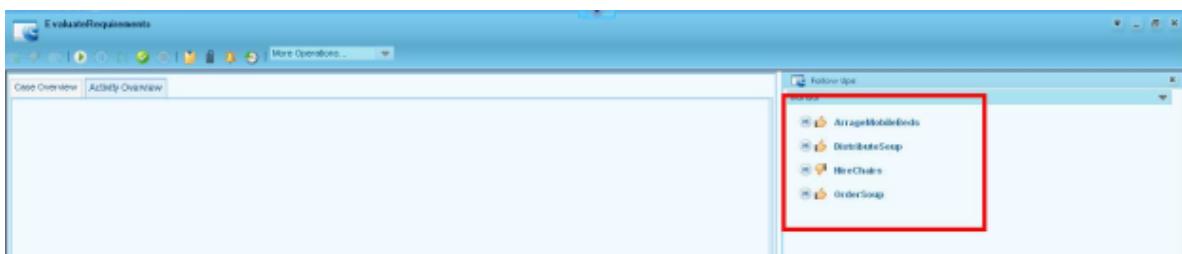
Note: If you are working in a Non-System organization, create a new Business process management Service group in that organization and attach the Web service interface.

4. Trigger the Case model with the Case data as parameters.

An easy way to trigger a Case model is to create another BPM and use this Case model in that BPM. Map the input message of Case model to the start event of the BPM.

5. Go to **My Inbox** and select the Case model.
6. Click on the Case instance from the list.
7. Select and double click the **EvaluateRequirements** activity.

This action internally invokes the Applicability Service defined on the state and evaluates the given inputs. When evaluated, the output information is sent to Inbox and the inbox renders the Follow-Ups view with appropriate details.



Triggering a case model from an XForm

[CreateCase] Web service operation is used to trigger a case model. This Web service operation is also used to pass case data and case variable details.

You can use a User interface document to capture input details and map them to case data and case variables.

Steps to trigger a case model

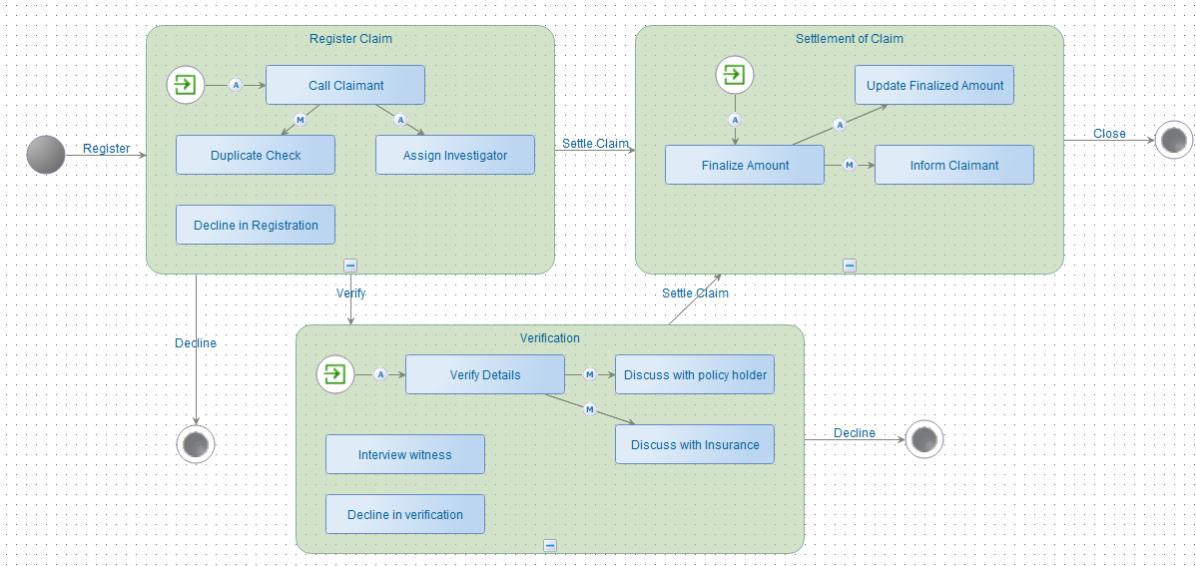
As an example, consider a business scenario where you fill details of a claim and trigger a case model. After you fill in the necessary set of details in the form, and click the Submit button, the claim case model should be triggered.

To trigger a case model:

1. Create an [Creating an XML Schema](#) document with the following XML data.
It creates a Schema fragment named ClaimDetails.

```
<ns:ClaimDetails xmlns:ns="http://myinsurance/claim/1.0">
  <ns:ClaimAmount/>
  <ns:CustomerID/>
  <ns:PolicyID/>
  <ns:PolicyType/>
</ns:ClaimDetails>
```

2. Create a case model and associate the above Schema fragment document as case data.



3. Create a User Interface document and drag the above Schema fragment document onto the User interface. This will automatically create the view with group box and input controls based on the schema.

Claim Details

Claim Amount	<input type="text"/>
Customer ID	<input type="text"/>
Policy ID	<input type="text"/>
Policy Type	<input type="text"/>

Submit

4. Double click on the User interface model editor area and go to the properties view. Add Init Done event and create a data model with below code.

```
function Form_InitDone(eventObject)
```

```
{
    claimDetailsGroupBox.create();
}
```

5. Create a non-transactional Model TriggerCaseModel that uses the [CreateCase] Web service operation to fire a request to initiate the case model from the User Interface. In this case, add the following XML to the XML Editor.

```
<xml id="triggerCaseReqXML">
    <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP:Body>
            <CreateCase
                xmlns="http://schemas.cordys.com/casemanagement/execution/1.0">
                <model space="organization">Case/ClaimCase</model>
                <casedata>
                    <data name="ClaimDetails" xmlns="http://myinsurance/claim/1.0"/>
                </casedata>
            </CreateCase>
        </SOAP:Body>
    </SOAP:Envelope>
</xml>
```

6. Write the code to send requests to the business process engine to trigger the case model. Add a button and add the following code for the onclick event of the button.

```
function submitBtn_Click(eventObject)
{
    var caseSOAPRequest = cordys.cloneXMLDocument(triggerCaseReqXML.XMLDocument);
    var modelDataNode = cordys.cloneXMLDocument(ClaimDetailsModel.getData());
    var claimDetailsNode = cordys.selectXMLNode(modelDataNode,"/*[local-name()
()='ClaimDetails']]");

    var casedataNode = cordys.selectXMLNode(caseSOAPRequest,"/*[local-name()
()='data']]");
    cordys.appendXMLNode(claimDetailsNode,casedataNode);
    TriggerCaseModel.setMethodRequest(caseSOAPRequest);
    TriggerCaseModel.reset()
}
```

The script is called on click of a button. It sends the SOAP request with specified input parameters.

The User interface from which you can trigger the case model is created.

Configuring attachments in BPM

Attaching a file to a task is an easy way to send important documents or images that may contain additional information about the task or instructions on performing certain actions. In a complex business scenario, we may have many Human Tasks, sub-BPMs and sub-Cases in a BPM. When the files are attached, the subsequent activities in the business process must also contain these attachments, and the users who have the necessary permissions must be able to act upon those attachments. The developer, while designing a business process model, can assign the attachments to the subsequent activities in a business process.

Before you begin:

- Ensure that the Business Process Designer and Business Process Engine application packages are installed, and the appropriate roles of each application package are assigned to you, as applicable.

The following sections discuss how to configure attachments from one activity to another.

To attach a user interface to a sub-BPM:

1. In BPM properties, go to the **Attachment** tab, and then click **+** to add an attachment definition.
2. Provide a proper attachment name, and select the supported document type from the list.
3. Go to the **Message Map** tab and under the process specific messages, create a message.
4. Right-click the message, and select **Paste XML as Element**, and paste the following XML and click **Import**.

```
<attachments>
    <attachment acl="" mime="" multiplicity="" name="">
        <instance description="" modifiedby="" modifiedon=""
name="">PARAMETER</instance>
    </attachment>
</attachments>
```

The attachment tag represents the attachment definition and has four attributes where name is the attachment definition name, mime is the document type, which this attachment definition supports (for example: txt, pdf, xls, and so on), multiplicity is "*", and ACL is the access permission, which can be read, update or delete.

The instance tag represents the attachment document and has four attributes where name is the file name, modified by is the username of the user who modified the document, modified on is the last modified date, and description is any description of the document. The instance tag takes the document store file path as the parameter.

5. Expand the newly created message on the left side.
The attachments element can be seen.
6. Expand the **attachments** tree from the right side.
The attachment defined in BPM can be seen as an element.
7. Drag and drop the **attachment** element from the process specific messages on the left side of the assignments and BPM attachment definition on the right side.
8. Select the **Replace Content With Children with Target NS** operation.
9. To map to the specific attachment definition, the XPath of the attachment element on the left side can be edited as follows.

```
bpm:inputMessage/bpm:attachments/bpm:attachment[@name="Attachment name"]
```

10. Click the **Model** tab, and in start properties, select **Trigger Type** as message and in **Input Message**, select the message that you created under process specific messages.
11. The user interface must trigger this BPM with a message in the following format.

```
<inputMessage xmlns="http://schemas.cordys.com/default">
  <inputElement xmlns="http://schemas.cordys.com/default">
    <attachments>
      <attachment acl="" mime="" multiplicity="" name="">
        <instance description="" modifiedby="" modifiedon="" name="">PARAMETER</instance>
      </attachment>
    </attachments>
  </inputElement>
</inputMessage>
```

To attach a human task to a human task:

1. In BPM properties, click the **Attachment** tab, and then click **+** to add an attachment definition.
2. Provide a proper attachment name and select the supported document type from the list.
3. Select the human task to which the user must be able to attach files.
4. Go to the human task properties, click the **Attachment** tab, and then click **+** to assign an attachment to the Human Task.
5. Select the attachment definition from the list and provide the necessary permissions by selecting the check box next to read, write, and delete.
6. Select the subsequent human task, which must also contain the attachments added in the previous human task.
7. Go to the human task properties, click the **Attachment** tab, and then click **+** to assign an attachment to this human task.

8. Select the same attachment definition that is selected for the previous activity from the list, and provide the necessary permissions by selecting the check box next to read, write, and delete.

To attach BPM to sub-Case:

1. In Case properties, click the **Attachment** tab, and then click **+** to add an attachment definition.
2. Provide a proper attachment name, select the supported document type from the list, and provide the necessary permissions by selecting the check box next to read, write, and delete.
3. Use this Case as a sub-Case in the BPM, which has a human task and sub-Case.
4. In BPM properties, go to the **Attachment** tab, and then click **+** to add an attachment definition.
5. Provide a proper attachment name, and select the supported document type from the list.
6. Select the human task to which the user must be able to attach files.
7. Go to the human task properties, click the **Attachment** tab, and then click **+** to assign an attachment to this Human Task.
8. Select the attachment definition from the list and provide the necessary permissions by selecting the check box next to read, write, and delete.
9. Click the **Message Map** tab and perform a pre-assignment for the Case.
10. Expand the attachments tree from the left side.
The attachment defined in BPM can be seen as an element.
11. Expand the Case tree from the right side.
The attachment defined in Case can be seen as an element.
12. Drag and drop the BPM attachment definition on the left side of the assignments and Case attachment definition on the right side.

Caution: When an attachment is passed from a business process to a case, users cannot update or delete it from the case instance view because the attachment is not owned by the case instance.

Publishing business identifier values

The PersistBusinessIdentifiers WSDL Web service is invoked after the execution of any activity which updates business identifiers value. It enables you to persist the changes to the business identifiers in custom repositories. Such storage is useful if applications need to have specific queries over the business identifiers and their values.

PersistBusinessIdentifiers WSDL

```
<?xml version="1.0"?>

-
<wsdl:definitions
  xmlns:tns="http://schemas.cordys.com/businessidentifiers/persistence/1.0"
  targetNamespace="http://schemas.cordys.com/businessidentifiers/persistence/1.0"
  name="PersistBusinessIdentifiers" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">-
  <wsdl:types>-
    <xsd:schema
      targetNamespace="http://schemas.cordys.com/businessidentifiers/persistence/1.0"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      attributeFormDefault="unqualified">-
      <xsd:element name="PersistBusinessIdentifiers">-
        <xsd:complexType>-
          <xsd:sequence>-
            <xsd:element name="Context">-
              <xsd:complexType>-
                <xsd:sequence>
                  <xsd:element name="Organization"
                    type="xsd:string"/>
-
-
      <xsd:element name="Model">-<xsd:complexType>-
<xsd:simpleContent>-<xsd:extension base="xsd:string"><xsd:attribute name="type"
type="xsd:string" use="required"/><xsd:attribute name="modelID" type="xsd:string"
use="required"/><xsd:attribute name="revisionID" type="xsd:string"
use="required"/><xsd:attribute name="modelSpace" type="xsd:unsignedByte"
use="required"/>
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
-
-
      <xsd:element name="Instance">-
        <xsd:complexType>-
          <xsd:simpleContent>-
            <xsd:extension
base="xsd:string"><xsd:attribute name="rootType" type="xsd:string"
use="required"/><xsd:attribute name="rootInstance" type="xsd:string"
use="required"/>
          </xsd:extension>
```

```

        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>

-
    <xsd:element name="status">-
        <xsd:complexType>-
            <xsd:sequence>
                <xsd:element name="processinstance"
type="xsd:string"/>

-
        <xsd:element
name="activityinstance">-<xsd:complexType>-<xsd:simpleContent>-<xsd:extension
base="xsd:string"><xsd:attribute name="id" type="xsd:string"
use="required"/><xsd:attribute name="name" type="xsd:string"
use="required"/><xsd:attribute name="corelationID" type="xsd:string"
use="required"/><xsd:attribute name="activityType" type="xsd:string"
use="required"/><xsd:attribute name="iterationCount" type="xsd:unsignedByte"
use="required"/>
                </xsd:extension>
                </xsd:simpleContent>
            </xsd:complexType>
        </xsd:element>
        <xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

-
    <xsd:element name="Content">-
        <xsd:complexType>-
            <xsd:sequence>-
                <xsd:element name="BusinessIdentifier"
maxOccurs="unbounded">-
                    <xsd:complexType>-
                        <xsd:simpleContent>-
                            <xsd:extension base="xsd:string">
                                <xsd:attribute name="name"
type="xsd:string" use="required"/>
                                <xsd:attribute
name="description" type="xsd:string" use="required"/>
                                <xsd:attribute name="type"
type="xsd:string" use="required"/>
                            </xsd:extension>

```

Before you begin:

- Provide implementation to `PersistBusinessIdentifiersOperation`.

To publish business identifiers:

1. Create the UDDI service group.
2. Copy the `PersistBusinessIdentifiers` WSDL onto a local Web server, by placing it in the webroot folder.
3. Configure the Web service with Java implementation.
4. Provide the Java implementation for the Web service created in Step 3.
5. Add the new Web service to a WS-Apps service group.
6. Select **Publish Business Identifier Values** in the BPM property sheet as follows:
 - a. Click the **Business Identifier** tab in the BPM property sheet.
 - b. For type of execution, select **Non-Transactional No-Reply**.
 - c. Add some business identifiers and also assignments for these business identifiers. See [Table 3. Fields on the Business Identifiers tab section](#) in the Business Process Model Properties Interface topic for more information on business identifier values.

A Web service can also be configured with CFD BPM, but it is not recommended, as BPM and the Web service that is created will be running in the same BPM SOAP processor. It may happen that the Web service is not notified when the SOAP processor is busy with the BPM processing. The work around for this is to create another BPM SOAP processor and attach only the newly created Web service.

Using a manual follow-up connector in a case model

When you want to ensure that case activities are performed manually, you need to use the manual follow-up connector.

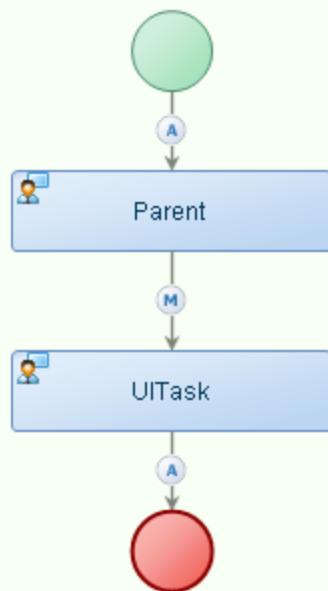
To use a manual follow-up connector in a case model:

1. To create a case model, [select a starting point](#) and click  (Case Management Model). The case model appears in case modeler.
2. To handle the start of a case, drag  from the toolbox to the case modeler to create a case model as shown in the example below.
3. Drag  from the toolbox to the case modeler. If the toolbox is not visible in the modeler, click  (Options) and select Toolbox.
4. Repeat Step 3 to drag another  from the toolbox to the case modeler.

5. Drag  from the toolbox to the modeling environment.
6. To link the constructs click  on the toolbar and select each construct in the case modeler. Alternatively, hold the CTRL key and select the constructs in the case modeler with the mouse.
All the constructs of the case model are now linked.
7. Click  to save the case model.

You have successfully used the manual follow-up connector in a case model.

Sample snapshot of a case model using the manual follow-up connector



Before completing the task, the parent activity should select the manual followup activity / task that should be executed.

After you complete this task:

- Publish the case model.

Enabling debugging on previous business process models

Testing an application includes finding issues, fixing them, and repeating the test scenarios. Once an issue is fixed, you may not analyze the results of the tests that are conducted on older versions of your applications. Similarly, developers may not be interested in debugging the instances of older versions of BPMs under test.

Before you begin:

- You need the AppWorks Platform Administrator role in the System organization and access to the file system of the AppWorks Platform server to perform this task.

On a default AppWorks Platform development environment, it is meaningful to open the graphical view of a BPM or its instances only if that BPM or instance is still in sync with the version as it resides in the development workspace from which it originates. In other words, while opening the graphical view of a BPM, the system will open the BPM designer using the original BPM from the development workspace. As a result, if the BPM has been modified after it has last been published, the graphical view does not correspond to the BPM, as it is available in the runtime.

To enable the functionality to debug old process instances:

1. Add the following property to the `cws.properties` file located at <AppWorks Platform_installdir>/components/cws/config:

```
development.publish.to.psl.enabled=true
```

2. Restart the Collaborative Workspace service container.

Enabling the feature to debug the instances of the previous versions affects the performance of publishing the BPM. Because the Published Source Layers that support this feature are stored in the form of stacks, publishing a BPM or opening the graphical view of a BPM will take much longer than usual.

Chapter 20

Application development (How Tos)

This chapter contains topics that help you use the various features available in AppWorks Platform to accomplish specific tasks while creating a Web application. The examples included in these topics help understand how a task can be performed. The topics included in this section are:

- [Developing Web services using the HTTP connector](#)
- [Accessing applications within applications using iframecontainer](#)
- [Calling a web service from a WS-AppServer application](#)
- [Creating an XForm using web service operation generated on data source](#)
- [Downloading a file from a service](#)
- [Extending a Connection in a WS-AppServer Application](#)
- [Handling globalization aspects in web application development](#)
- [Reordering rows in a table](#)
- [Using Inbox as a Composite Control](#)
- [Uploading a file to a service](#)
- [Using WS-AppServer Custom Class as an Alternative to Object Template](#)
- [Developing Applications with HTML5 SDK](#)
- [Invoking AppWorks Platform REST APIs](#)

Developing Web services using the HTTP connector

This topic describes the procedure to develop Web services using HTTP connector with the OpenData.Education service.

Using REST with HTTP connector

You must create a new CWS project which uses the HTTP connector to integrate with the OpenData.Education service. The OpenData API is a REST-based API that accepts GET requests on specific URLs with certain URL parameters. Token-based authentication may be required for some operations, but is not needed for these samples.

The following example uses the v2 feeds API. See <https://opendata.education/> for more information.

You must set up a basic CWS project.

To set up a basic CWS project:

1. Create a new CWS project in which you want to use the HTTP connector.
2. In the CWS project, add a Runtime Reference to the HTTP connector:
 - a. Create a new folder **Runtime Reference** in the project.
 - b. In this folder, add a run-time reference to an application connector.
 - c. From the folder **OpenText HTTP Connector**, select **OpenText HTTP Connector**, and click **Finish**.
3. Add **HttpConnector** configuration as follows:
 In run time, HTTP connector searches for configuration in the XML store under `OpenText/HttpConnector` path. Hence, it is important to define the XML store path correctly in the CWS.
 - a. Create a folder **XML Store**.
 - b. Right-click **XML Store** and select **Set Start Point of Qualified Name_Make**.
 - c. In the **XML Store** folder, add an XML Store Definition and name it **HttpConnector Config**.
 - d. Create a folder **OpenText** in the **XML Store** folder.
 - e. Open the **OpenText** folder and add a sub-folder **HttpConnector**.
 - f. Open **HttpConnector** folder and add the XML object **config.xml**.
 This object contains the actual HTTP connector configuration.
 - g. Add the following HTTP configuration and save the XML object.

```
<configurations xmlns="http://httpconnector.opentext.com/1.0/configuration">
  <connections>
    <connection id="OPENDATA-EDUCATION">
      <url>https://api.opendata.education</url>
    </connection>
  </connections>
</configurations>
```

The namespace for the HTTP connector configuration is `http://httpconnector.opentext.com/1.0/configuration`. It is added as the default namespace on the configuration tag.

The configuration includes one connection with the ID CONN-DELICIOUS-FEED.

The following parameters are used in the HTTP connector connection configuration.

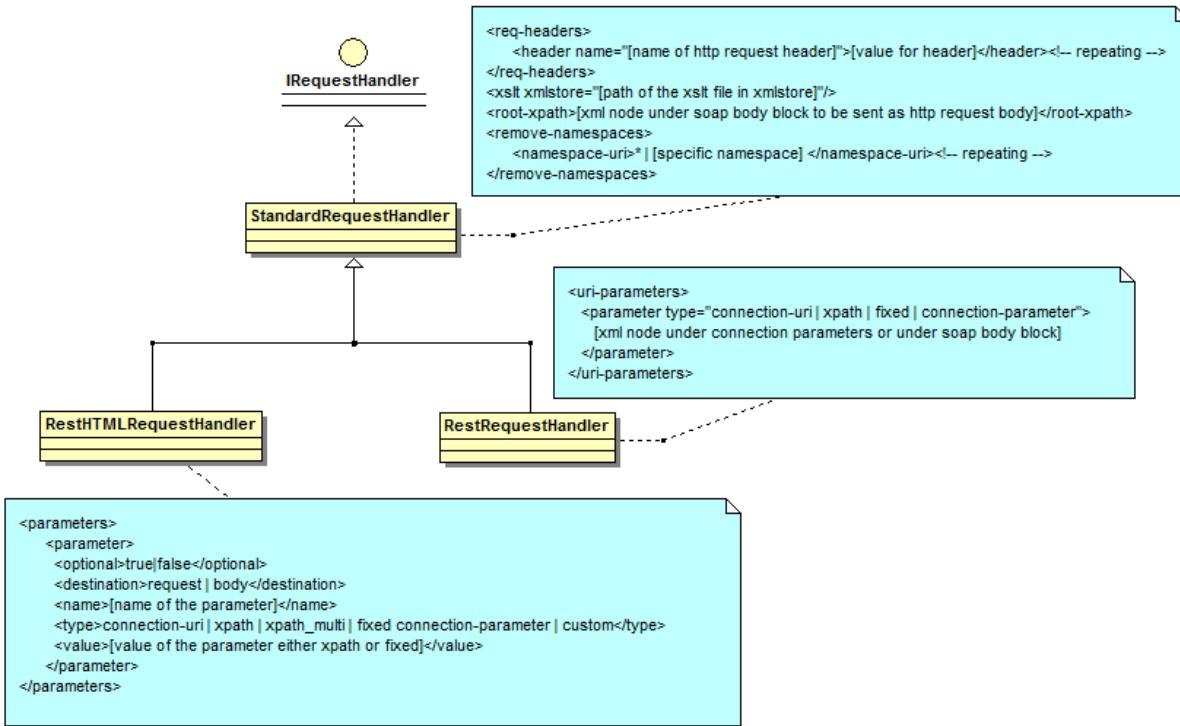
<code>@id</code>	Attribute of the connection tag and is the name of the connection.
<code>url</code>	The server endpoint URL. It can include a part or the complete URL

	of the service endpoint.
authenticate-always	Optional. Always sends the authentication information, also known as Preemptive Authentication. Default value is false.
check-certificate	Optional. Used with HTTPS connections only. It can be used as an option to check the SSL certificate. In a test environment, you can choose to ignore any errors while validating the SSL certificate. When it is set to false, any certificate that is either valid or invalid, expired, or not expired is accepted. Default value is false.
username	Optional. Refers to the username used for authentication at the service endpoint.
password	Optional. The password in base64 encoding. It is used for authentication at the service endpoint.
timeout	Optional. HTTP request timeout value in milliseconds. Default value is 30000 (30 seconds).
ignore-cookies	Optional. Set ignore-cookies to true to prevent the HttpClient from sending or receiving cookies. Default value is false.

Creating a Web service using the HTTP connector

Available handlers

The following diagram shows the various handlers available in the HTTP connector and the XML for the parameters in implementation.



This is a sample implementation XML and the related explanation on how the extended handlers reuse the implementation XML from the super or generic handlers, and how to configure such implementation XML to leverage the functionality of multiple handlers.

HTTP Get request without parameters

Add the Web service

1. From the project root, create a folder named **Web Services**.
2. Right-click the Web Services folder and add a Web service. Use **Custom Web Service** as the source.
3. Name the Web service with a name such as **OpenDataEducation Service**.
4. Provide a **Description** or use the default description (same as the name).
5. Provide a **Namespace** for the service (Example: odes).
6. Enter **OpenDataEducation Service** as the **Web Service Interface Name**.
7. Select the **OpenText HTTP Connector** as the **Implementation Class**.
8. Add a Web service operation **GetBirminghamInfo**.
9. Click **Finish**.
10. Open **GetBirminghamInfo** and paste the following XML in the implementation field.

```
<implementation type="HTTP">
```

```

<xmlns="http://httpconnector.opentext.com/1.0/implementation"
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <connection-id>OPENDATA-EDUCATION</connection-id>
  <uri>/rest/sif/requests/SchoolInfos/C693E567C79259EA95669EA2F811ED63</uri>
  <http-method>GET</http-method>
  <request-handler
    class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler"/>
    <response-handler
      class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandle
      r"/>
      <valid-response-code>200</valid-response-code>
      <namespaces/>
    </implementation>

```

The completes and describes how the OPENDATA-EDUCATION connection is used.

uri	Contains the additional URL of the service endpoint. The service endpoint is a concatenation of the connection-id URL and the URI.
http-method	Used for GET method. HTTP connector also supports POST, PUT, and DELETE.
request-handler class	The <code>RestRequestHandler</code> builds the correct REST request.
response-handler class	The <code>StandardResponseHandler</code> returns the response XML.
valid-response-code	Check if the HTTP response code is a valid one. This indicates whether the request was successful or not.
namespace	Parameter is empty and not used with the <code>RestRequestHandler</code> .

Run the Web service

1. Right-click the project root and select **Publish to Organization**.
Part of this process is adding `config.xml` to the organization XML store.
2. In the System Resource Manager, create a new HTTP Connector Service Group and service container if one does not already exist. Be sure to select the interface you created above when prompted and ensure that you add `config.xml` to the **XMLStore Configuration File Path** field.
3. Start the HTTP connector container to load the configuration. After changing and publishing a Web service implementation, use the Reset option to clear the internal method definition cache of the HTTP connector.
4. To test the Web service operation, right-click the **GetBirminghamInfo** Web service operation, and select **Test Web Service Operation**.
The Operation Test Tool appears with a SOAP request for the GetBirminghamInfo Web service operation.
5. Remove the placeholder PARAMETER text (this operation does not require additional input) and Invoke the operation. The HTTP connector calls the OpenData Rest service to get the requested school info.

Sample SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetBirminghamInfo xmlns="odes"/>
  </SOAP:Body>
</SOAP:Envelope>
```

Sample SOAP response

```
<data>
  <GetBirminghamInfoResponse xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" xmlns="odes">
    <URN xmlns="http://www.sifassociation.org/datamodel/uk/2.0">139269</URN>
    <LAId xmlns="http://www.sifassociation.org/datamodel/uk/2.0">330</LAId>
    <EstablishmentId
      xmlns="http://www.sifassociation.org/datamodel/uk/2.0">2121</EstablishmentId>
    <SchoolName xmlns="http://www.sifassociation.org/datamodel/uk/2.0">Hawkesley
      Church Primary Academy</SchoolName>
    <SchoolFullName xmlns="http://www.sifassociation.org/datamodel/uk/2.0">Hawkesley
      Church Primary Academy</SchoolFullName>
    <SchoolURL
      xmlns="http://www.sifassociation.org/datamodel/uk/2.0">www.hawkesley.bham.sch.uk/<SchoolURL>
    <SchoolPhoneNumber xmlns="http://www.sifassociation.org/datamodel/uk/2.0">
      <Number Type="W">01214596467</Number>
    </SchoolPhoneNumber>
    <OperationalStatus
      xmlns="http://www.sifassociation.org/datamodel/uk/2.0">1</OperationalStatus>
    <GenderOfEntry
      xmlns="http://www.sifassociation.org/datamodel/uk/2.0">C</GenderOfEntry>
    <Boarders xmlns="http://www.sifassociation.org/datamodel/uk/2.0">No</Boarders>
    <Special xmlns="http://www.sifassociation.org/datamodel/uk/2.0">No</Special>
    <SchoolAddress
      xmlns="http://www.sifassociation.org/datamodel/uk/2.0"
      Type="Current">
      ...
    </SchoolAddress>
    <HeadTeacherInfo xmlns="http://www.sifassociation.org/datamodel/uk/2.0">
      <ContactName>Mr Derek Higgins</ContactName>
    </HeadTeacherInfo>
    <Phase xmlns="http://www.sifassociation.org/datamodel/uk/2.0">PY</Phase>
    <NCYearGroupList xmlns="http://www.sifassociation.org/datamodel/uk/2.0">
      <NCYearGroupList>N2</NCYearGroupList>
      <NCYearGroupList>R</NCYearGroupList>
      <NCYearGroupList>1</NCYearGroupList>
      <NCYearGroupList>2</NCYearGroupList>
      <NCYearGroupList>3</NCYearGroupList>
      <NCYearGroupList>4</NCYearGroupList>
      <NCYearGroupList>5</NCYearGroupList>
      <NCYearGroupList>6</NCYearGroupList>
    </NCYearGroupList>
  </GetBirminghamInfoResponse>
</data>
```

```

<SIF_ExtendedElements xmlns="http://www.sifassociation.org/datamodel/uk/2.0">
  ...
</SIF_ExtendedElements>
<SIF_Metadata xmlns="http://www.sifassociation.org/datamodel/uk/2.0">
  ...
</SIF_Metadata>
</GetBirminghamInfoResponse>
</data>

```

HTTP Get request with parameters service URI

The previous example has a fixed service URI in the Web service operation.

To add a new Web service operation:

1. Select **Add/Update** operations on the **OpenDataEducation Service** Web service.
2. From **Select the source**, select **Custom Web Service**.
3. Select **Overwrite to existing Web Service Interface** and select **OpenDataEducation Service** Web service.
4. Add a new Web Service Operation and name it **GetInfoForSchool**.
5. Add the following **Web Service Operation** implementation.
6. Publish the Web service **OpenDataEducation Service** to organization
7. Reset the HTTP connection service container.
8. Test the new Web service operation **GetInfoForSchool**.

GetInfoForSchool implementation

```

<implementation type="HTTP"
  xmlns="http://httpconnector.opentext.com/1.0/implementation"
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <connection-id>OPENDATA-EDUCATION</connection-id>
  <http-method>GET</http-method>
  <uri>/rest/sif/requests/SchoolInfos/{0}</uri>
  <request-handler
    class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
    <uri-parameters>
      <parameter type="xpath">./School</parameter>
    </uri-parameters>
  </request-handler>
  <response-handler
    class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandle
r"/>
  <valid-response-code>200</valid-response-code>
  <namespaces/>
</implementation>

```

The URI parameter formats the URI according to the given format. For example /rest/sif/requests/SchoolInfos/{0} or /project/{0}/task/{1}. The parameters {0}, {1}, and so on are filled with the parameters included in the URI parameters. Note: The connection URI parameters start at 0.

Example SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
    <GetInfoForSchool xmlns="odes">
        <School>C693E567C79259EA95669EA2F811ED63</School>
    </GetInfoForSchool>
</SOAP:Body>
</SOAP:Envelope>
```

On execution of the above Web service operation, the value of the School tag is used as a URL parameter.

The following is another example of using the URI parameters, this time with multi-level input. It has two replaceable parameters: the type of information requested (SchoolInfos) and the school ID.

Add it as Web service operation GetInfoWithMultiLevelInput:

```
http://feeds.delicious.com/v2/rss/cordysrest/cloud+cordys
```

Add it as Web service operation GetBookmarksForUserWithMultiLevelInput:

```
<implementation type="HTTP"
xmlns="http://httpconnector.opentext.com/1.0/implementation"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <connection-id>OPENDATA-EDUCATION</connection-id>
    <http-method>GET</http-method>
    <uri>>/rest/sif/requests/{0}/{1}</uri>
    <request-handler
class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
        <uri-parameters>
            <parameter type="xpath">./InfoType</parameter>
            <parameter type="xpath">./Type/RefID</parameter>
        </uri-parameters>
    </request-handler>
    <response-handler
class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandle
r"/>
        <valid-response-code>200</valid-response-code>
        <namespaces/>
    </implementation>
```

SOAP request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <GetInfoWithMultiLevelInput xmlns="odes">
      <InfoType>SchoolInfos</InfoType>
      <Type>
        <RefID>A45EBD4D670356C5A3E145F83E769CB3</RefID>
      </Type>
    </GetInfoWithMultiLevelInput>
  </SOAP:Body>
</SOAP:Envelope>
```

HTTP PUT/POST request

The steps to create a Web service operation for HTTP Put/Post are same as those for HTTP Get.

The request handler to be used in the Web service operation implementation is based on the service type, for example, for standard Web services use

`com.opentext.applicationconnector.httpconnector.impl.StandardRequestHandler` or for REST services use

`com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler`. The request handler picks up the content of the SOAP request, XML node below the SOAP body, and applies necessary transformation and put it inside the HTTP request body.

HTTP Delete request

The steps to create a Web service operation for HTTP Delete including the request and response handlers that are applicable are exactly the same as those for HTTP Get.

Transforming request and response using XSLT

To specify request and response XSLTs:

1. Create a folder named **xslts** within your XML Store folder. Create and save XSLT files into this folder by right-clicking on the `xslts` folder and creating new XML objects.
2. Modify the implementation of the operation to which you want to apply the XSLT. Add the XSLT element to either the request-handler or response-handler (or both). The element format is: `<xslt xmlstore="<path>" />`
3. The request XSLT is applied to the incoming SOAP request and the output of the transformation is put inside the HTTP request body.
This is applicable only to HTTP Put and Post as these are the requests that can contain a payload in the request body.
4. The response XSLT is applied to the HTTP response received and the output of the transformation is put inside the SOAP response under the SOAP Body element.
This is applicable to all the four HTTP request methods (Get,Put,Post, and Delete) supported by the HTTP connector.

The following is a sample XSLT that can be applied to the GetBirminghamInfo operation. This XSLT formats the response to include only the head teacher and address information. This should be saved in the `xslts` folder as `schoolResponseXSLT.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:odes="http://www.sifassociation.org/datamodel/uk/2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
    <xsl:element name="ContactInfo">
        <xsl:for-each select="odes:SchoolInfo/odes:HeadTeacherInfo">
            <xsl:element name="HeadInstructor">
                <xsl:value-of select="odes:ContactName" />
            </xsl:element>
        </xsl:for-each>
        <xsl:for-each select="odes:SchoolInfo/odes:SchoolAddress">
            <xsl:element name="Address">
                <xsl:element name="Street">
                    <xsl:value-of select="odes:Street" />
                </xsl:element>
                <xsl:element name="Locality">
                    <xsl:value-of select="odes:Locality" />
                </xsl:element>
                <xsl:element name="Town">
                    <xsl:value-of select="odes:Town" />
                </xsl:element>
                <xsl:element name="County">
                    <xsl:value-of select="odes:County" />
                </xsl:element>
                <xsl:element name="PostCode">
                    <xsl:value-of select="odes:PostCode" />
                </xsl:element>
                <xsl:element name="Country">
                    <xsl:value-of select="odes:Country" />
                </xsl:element>
            </xsl:for-each>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

You can then update the operation implementation to trigger the transformation when the response is received. The process is the same for adding a transformation to the request. The updated implementation:

```
<implementation xmlns:c="http://schemas.cordys.com/cws/1.0"
    xmlns="http://httpconnector.opentext.com/1.0/implementation"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">
    <connection-id>OPENDATA-EDUCATION</connection-id>
    <uri>/rest/sif/requests/SchoolInfos/C693E567C79259EA95669EA2F811ED63</uri>
```

```

<http-method>GET</http-method>
<request-handler
class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler" />
<response-handler
class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandle
r">
    <xslt xmlstore="xslts\schoolResponseXSLT.xml" />
</response-handler>
<valid-response-code>200</valid-response-code>
<namespaces />
</implementation>

```

When the implementation has been updated, publish the changes to the organization. You must publish the **xslts** folder with the XML Store as well as the updated operation. Restart the HTTP Connector service container for the changes to be effective.

At this point you can test the operation. The response will be formatted differently and will include only the information specified in the XSLT.

```

<data>
    <GetSchoolInfoResponse xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <HeadInstructor xmlns:odes="http://www.sifassociation.org/datamodel/uk/2.0">Mr
Derek Higgins</HeadInstructor>
        <Address xmlns:odes="http://www.sifassociation.org/datamodel/uk/2.0">
            <Street>376 Shannon Road</Street>
            <Locality>Kings Norton</Locality>
            <Town>Birmingham</Town>
            <County>West Midlands</County>
            <PostCode>B38 9TR</PostCode>
            <Country>GBR</Country>
        </Address>
    </GetSchoolInfoResponse>
</data>

```

Handling JSON as input and output parameters of REST service

JSON is an open standard format used to transfer data between a Web-based application or service and server and is very popular as an alternative to XML data format in the Web. JSON objects, in the form of attribute-value pairs, can be used as an input to a Web service and the response of the Web service can be in the JSON format.

Using HTTP connector, you can invoke Web service that accepts JSON objects as input or the Web service that returns the JSON objects. For HTTP GET request, there is no change in the request format.

In case of HTTP POST request, set Content-Type as 'application/json' in the header part of RestRequestHandler as follows.

```

<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation"
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">
  .....
    <connection-id>JSON-TEST</connection-id>
    <request-handler
      class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
      <req-headers>
        <header name="Content-Type">application/json</header>
      </req-headers>
    </request-handler>
    <response-handler
      class="com.opentext.applicationconnector.httpconnector.impl.RestResponseHandler"/>
    <valid-response-code>200</valid-response-code>
  .....
</implementation>

```

If it is a JSON response, then the HTTP connector transforms that to XML before sending it to the caller.

Using authentication

The HTTP connector only supports basic authentication. Other forms of authentication (such as token-based authentication) may be available by using custom request headers.

To use authentication:

1. Modify or create a new configuration.
2. Include the username, password, and authenticate-always parameters. The password must be Base64 encoded.

There are no changes required to a service operation to use authentication. A sample configuration follows.

```

<configuration xmlns="http://httpconnector.opentext.com/1.0/configuration">
  <connections>
    <connection id="CONNECTION-ID">
      <url>https://server.example.com</url>
      <username>username</username>
      <password>cGFzc3dvcmQ=</password>
      <authenticate-always>true</authenticate-always>
    </connection>
  </connections>
</configuration>

```

3. Restart the HTTP connector service container to load the changes.

Using HTTPS

When attempting to connect to a service that is secured (https) you must either disable the check-certificate functionality within the connection configuration (this is disabled by default), add the certificate to the Java keystore, or add a valid certificate to the platform through the Security Administration tool.

To disable the certificate check:

- Set the check-certificate parameter value in the connection configuration to false (or remove the parameter).

Caution: Disabling the certificate check may be sufficient in a test or debug environment, but should be considered an unsafe operation in production environments.

To add the certificate to the Java keystore or to the Certificates within the Security Administration tool:

- First obtain a copy of the client certificate. This can usually be done through a Web browser or through the use of an external tool such as OpenSSL. Follow your organizations security protocols when obtaining a certificate.

After a certificate has been obtained you can begin the import.

Importing through the Security Administration tool:

1. Open the security administration tool and click the Certificates tab.
2. Click the Add button.
3. To upload the certificate:
 - a. Set the mode to **Upload certificate**, and supply a path.
 - b. Once the upload is complete, the name will be populated.
 - c. Click **OK** to finish.
4. To paste the certificate:
 - a. Set the mode to **Paste Base64 Certificate**.
 - b. Copy the contents of the certificate file, including the BEGIN and END certificate tags.
 - c. Paste the contents into the Text control.
 - d. The name will be filled in automatically.
 - e. Click **OK** to finish.
5. Close the Security Administration tool. You may be required to restart your HTTP Connector service group to take advantage of the added certificates.

Importing to the default Java keystore (cacerts):

Another, lesser preferred option, is to import the certificate in the Java keystore.

1. Copy the certificate file to the <JRE_HOME>/lib/security directory.
2. Open a terminal or command prompt window and navigate to that directory.

3. Run the following command to import the certificate. You can name the alias whatever you want. When prompted, enter the keystore password (default password is **changeit**):

```
keytool -import -alias <name> -keystore cacerts -file <certificate file>
```

4. Accept the trust and it will be added to the keystore.

Adding custom HTTP headers

Many REST APIs expect custom HTTP request headers. The following table lists a few examples.

REST Service Provider	Custom Headers	Value
Azure	x-ms-version	2011-08-18
Google Data API	GData-Version	X.0
Tibbr API (Authenticated Session)	client_key	[applicable value]
	auth_token	[applicable value]

The custom HTTP headers can be set by configuring the <implementation> for <req-headers> as follows:

```
<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation"
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">
  <connection-id>AZURE_MANAGEMENT</connection-id>
  <uri>/{0}/services/hostedservices/{1}/deployments/{2}/roles</uri>
  <http-method>POST</http-method>
  <request-handler
    class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
    <uri-parameters>
      <parameter type="connection-parameter">subscriptionID</parameter>
      <parameter type="xpath">ServiceName</parameter>
      <parameter type="xpath">DeploymentName</parameter>
    </uri-parameters>
    <req-headers>
      <header name="x-ms-version">2012-03-01</header>
      <header name="Content-Type">application/xml</header>
    </req-headers>
    <root-xpath>./PersistentVMRole</root-xpath>
  </request-handler>

  <valid-response-code>201</valid-response-code>
  <namespaces>
    <binding prefix="ns" uri="http://schemas.microsoft.com/windowsazure" />
  </namespaces>
</implementation>
```

Organization awareness

When HTTP connector is configured in the system organization, it becomes organization-aware. The behavior of the HTTP connector running in the system organization is as follows:

- The HTTP connector configuration file is read from the organization of the user sending the request.
 - If the configuration file does not exist in the organization space, then it is picked up from the shared (ISV) space.
- The XSLTs is read from the organization of the user sending the request.
 - If the XSLT does not exist in the organization space, then it is picked up from the shared (ISV) space.

When HTTP connector is configured in any other organization, then the configuration files and the XSLTs are picked up from the organization in which the service container is running. If not found in the organization space, the HTTP connector searches for the files in the shared (ISV) space.

Adding HTTP header content dynamically

When there is a need to pass dynamic value for a HTTP header parameter, it should be defined in the Web service implementation and the value must be an XPath expression that points to an element in the SOAP body. This type of header variable is termed as Dynamic Header Variable.

Before calling the end point URL, the value of the Dynamic Header Parameter is set to the value of the element pointed to by the XPath expression.

Sample interface

Implementation

Create a custom Web service with the implementation as follows:

```
<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation"
  xmlns:c="http://schemas.cordys.com/cws/1.0"
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">
  <connection-id>SFDC</connection-id>
  <uri>/services/data/v36.0/sobjects/Account/001g000000w5Eza</uri>
  <http-method>GET</http-method>
  <request-handler
    class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
    <req-headers>
      <header name="Authorization" type="xpath">./Authorization </header>
      <header name="Content-Type">application/xml</header>
      <header name="Accept">application/xml</header>
    </req-headers>
  </request-handler>
</implementation>
```

```

</request-handler>
<response-handler
class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandler"
/>
<valid-response-code>200</valid-response-code>
<namespaces />
</implementation>

```

SOAP request

When triggering the request, the request must be created as follows:

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
    <getSFDC_Rest xmlns="http://com.TWPS.test1/rest">
        <Authorization>Bearer
00Dg0000006SEja!AQ0AQFuSgwBXZVxUfRhaGnp1fc5IxSwaveACOI393L6oX_MfY7JwN2c1QD_
MFCCjo_.Ap0sgBp5KXZc5NHB8tiQ4jshxAy6G</Authorization>
    </getSFDC_Rest>
</SOAP:Body>
</SOAP:Envelope>

```

SOAP response (success case)

When the request is triggered, the Request Handler must read the value of Authorization Parameter, set it to the header, and return the response as follows:

```

<data>
    <getSFDC_RestResponse xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="http://com.TWPS.test1/rest">
        <Id xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">001g000000w5EzaAAM</Id>
        <IsDeleted xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">false</IsDeleted>
        <Name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Pizza
Express</Name>
        <RecordTypeId xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">0122000000LkpAAC</RecordTypeId>
        <BillingStreet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">123,
Kings Treet</BillingStreet>
        <BillingCity xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">Swindon</BillingCity>
        <BillingPostalCode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">RG1
8DB</BillingPostalCode>
        <BillingCountry xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">United
Kingdom</BillingCountry>
        <BillingAddress xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="address">
            <latitude xsi:nil="true" />

```

```

<longitude xsi:nil="true" />
<city>Swindon</city>
<country>United Kingdom</country>
<countryCode xsi:nil="true" />
<geocodeAccuracy xsi:nil="true" />
<postalCode>RG1 8DB</postalCode>
<state xsi:nil="true" />
<stateCode xsi:nil="true" />
<street>123, Kings Treet</street>
</BillingAddress>

</getSFDC_RestResponse>
</data>

```

SOAP response (failure case)

When the session fails, the following SOAP fault is displayed:

```

<data>
<SOAP:Fault xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <faultcode
    xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/">ns0:Server</faultcode>
  <faultstring xml:lang="en-US">Invalid status code received: 401 (expected: 200).</faultstring>
  <faultactor>http://com.TWPS.test1/rest</faultactor>
  <detail>
    <cordys:FaultDetails xmlns:cordys="http://schemas.cordys.com/General/1.0/">
      <cordys:LocalizableMessage
        xmlns:cordys="http://schemas.cordys.com/General/1.0/">
        <cordys:MessageCode
          xmlns:cordys="http://schemas.cordys.com/General/1.0/">com.opentext.applicationconnector.httpconnector.exception.ConnectorExceptionMessages.invalidStatusCodeReceived</cordys:MessageCode>
          <cordys:Insertion
            xmlns:cordys="http://schemas.cordys.com/General/1.0/">401</cordys:Insertion>
          <cordys:Insertion
            xmlns:cordys="http://schemas.cordys.com/General/1.0/">200</cordys:Insertion>
          </cordys:LocalizableMessage>
        </cordys:FaultDetails>
        <cordys:FaultRelatedException
          xmlns:cordys="http://schemas.cordys.com/General/1.0/">
          <![CDATA
[com.opentext.applicationconnector.httpconnector.exception.ConnectorException:
Invalid status code received: 401 (expected: 200).
          at
com.opentext.applicationconnector.httpconnector.execution.MethodExecutor.sendRequest
(MethodExecutor.java:165)
          at com.opentext.applicationconnector.httpconnector.CustomMethod.prepare
(CustomMethod.java:56)
          at com.opentext.applicationconnector.HttpTransaction.process

```

```
(HttpTransaction.java:147)
    at com.eibus.soap.SOAPTransaction.processApplicationTransaction
(SOAPTransaction.java:1357)
    at com.eibus.soap.SOAPTransaction.handleBodyBlock(SOAPTransaction.java:1281)
    at com.eibus.soap.SOAPTransaction.<init>(SOAPTransaction.java:558)
    at com.eibus.soap.SOAPTransaction.<init>(SOAPTransaction.java:210)
    at com.eibus.soap.Processor.onReceive(Processor.java:1335)
    at com.eibus.soap.Processor.onReceive(Processor.java:1316)
    at com.eibus.connector.nom.Connector.onReceive(Connector.java:483)
    at com.eibus.transport.NonTransactionalWorkerThreadBody.doWork
(NonTransactionalWorkerThreadBody.java:61)
    at com.eibus.transport.NonTransactionalWorkerThreadBody.run
(NonTransactionalWorkerThreadBody.java:26)
    at com.eibus.util.threadpool.WorkerThread.run(WorkerThread.java:67)
]]>
    </cordys:FaultRelatedException>
</detail>
</SOAP:Fault>
</data>
```

Common errors

**IllegalOperationException: Prefix and/or URI cannot be NULL.
[addNamespaceBinding:166]**

Solution: You do not have a default namespace on the configuration or implementation. Ensure that you have the following:

```
<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">

<implementation xmlns="http://httpconnector.opentext.com/1.0/implementation"
    xmlns:c="http://schemas.cordys.com/cws/1.0"
    xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" type="HTTP">
    <connection-id>SFDC</connection-id>
    <uri>/services/data/v36.0/sobjects/Account/001g00000w5EZa</uri>
    <http-method>GET</http-method>
    <request-handler
        class="com.opentext.applicationconnector.httpconnector.impl.RestRequestHandler">
        <req-headers>
            <header name="Authorization">./
                Authorization </header>
            <header name="Content-Type">application/xml</header>
            <header
                name="Accept">application/xml</header>
        </req-headers>
    </request-handler>
    <response-handler
        class="com.opentext.applicationconnector.httpconnector.impl.StandardResponseHandler"
    />
```

```

<valid-response-code>200</valid-response-code>
<namespaces />
</implementation>

```

Accessing applications within applications using iframecontainer

The application framework provided in the AppWorks Platform application enables you to invoke other applications using the `application.select` method. In several cases, applications may need to be nested within other applications, and remain within the AppWorks Platform application framework.

AppWorks Platform provides a web component called `iframecontainer`. Using this component, any application can contain another application nested inside it. The applications contained within applications in this manner will also be listed in the `system.windows` and `system.containers` collection, and can be accessed by all other applications inside the AppWorks Platform framework. Applications opened inside the iframes behave like any other normal application in AppWorks Platform framework. To view a description about the `system.windows` and `system.containers` API, see *System in the AppWorks Platform API Guide*.

To access applications using `iframecontainer`:

1. Define an iframe using the `iframecontainer` component"

```
<iframe class="container" id="myFrame"/>
```

2. Define an application within the iframe using the application definition:

```

<script id="appDefinition" type="cordys/xml">
  <Application>
    <url>/cordys/wcp/demo/multifunctionalgrid.htm</url>
    <id>myApplication</id>
    <frame> myFrame </frame>
    <description>Multi-function Grid</description>
    <caption>Multi-function Grid</caption>
  </Application>
</script>

```

3. Invoke the application defined in the iframe, from another application in AppWorks Platform, using the following method:

```
application.select(appDefinition.XMLDocument.documentElement);
```

To view a description about the application.select API, see Application in the *AppWorks Platform API Guide*.

This enables the application in the iframe to be invoked from the specified application.

Example

The following example demonstrates the use of `iframecontainer` component to view applications.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html onapplicationready='init()'>
<head>
    <title>Applications in an iFrame</title>
    <script src='/cordys/wcp/application.js'></script>
    <script type='text/javascript'>
function init()
{
    application.select(appDefinition.XMLDocument.documentElement);
}
</script>
<script type="cordys/xml" id='appDefinition'>
<Application>
    <url>/cordys/wcp/demo/multifunctionalgrid.htm</url>
    <id>myApplication</id>
    <frame> myFrame</frame>
    <description>Multi-function Grid</description>
    <caption>Multi-function Grid</caption>
</Application>
</script>
</head>
<body>
    <iframe class='container' id='myFrame' style='width:500px;height:400px'>
</body>
</html>
```

Calling a web service from a WS-AppServer application

While building a WS-AppServer application, you may need to implement additional business logic that resides in a separate Web service. To fulfill this requirement, you need a provision within WS-AppServer application to call that Web service.

The Extension class in WS-AppServer provides this flexibility, where you can add the required code to call the Web service. During runtime, the application logic calls the Web service and executes the functionality.

To call a web service from a WS-AppServer application:

1. Identify the Web service that you want to integrate in your application logic.
2. In the Extension class that contains the custom logic, add the code that calls the external Web service during runtime (see code snippet in the following example).
3. Regenerate the Java code and the Web service Interface.

The application is equipped with the necessary logic to call a Web service.

Example

Assume that you are an automobile dealer with various automobile manufacturers. You want to procure spare parts for a particular make and model. The information about a specific spare part resides with the automobile manufacturer.

Through your WS-AppServer application, you want to call the Web service (published by that manufacturer) that contains information on the spare part. To enable this, write a method in your application to send a SOAP request to the Web service with specific parameters, and get the spare part details in the response. Based on the results, you can program further action.

The following sample code illustrates how to call such a Web service:

```
int sparePartNumber = 110; //request parameter value
String[] paramNames = {"PartNo"};
Object[] paramValues = {sparePartNumber}; //Construct a
SOAPRequestObject that calls AppWorks Platform web services
SOAPRequestObject sro = new SOAPRequestObject("http://schemas.cordys.com/automotive", "GetSparesObject",
paramNames, paramValues); //Execute the soap request. This is a synchronous call.
int response = sro.execute(); //the above method call will be transformed into below
soap call /* <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body> <GetSparesObject xmlns="http://schemas.cordys.com/automotive">
<PartNo>110</PartNo> </GetSparesObject> </SOAP:Body> </SOAP:Envelope> */
```

In the code, `GetSparesObject` is the method, and `110` is the parameter value of the spare part number. At run time, the code constructs a new SOAP request, interacts with the Web service, and gets the required information in a response.

Creating an XForm using web service operation generated on data source

AppWorks Platform provides you the facility to create an interface for the Web service operation that you generated. At run time, when users work with the user interface, it triggers the related Web service operation to execute necessary actions on the configured data source.

Before you begin:

- Create a blank XForm.
- Generate Web service operations on a data source using Database Metadata.

To create an XForm using a web service operation generated on data source:

1. Open the XForm that you created.
The XForm window opens displaying the name of the XForm on its title bar.
2. Locate the Web service operation  that you generated, and drag-and-drop it in the empty space on the **Design** tab of the XForm.
The Generate Input and Output UI dialog box opens displaying the details from the Web service operation.
3. On the toolbar of the XForm, click .
The XForm is saved with the generated user interface details.
4. On the toolbar, click  (Preview).
The preview of the XForm opens.
5. Depending upon the Web service operation that you used, provide an input value in the Input UI and execute the action.
Sometimes, an error appears while displaying the results. In such a case, restart the WS-AppServer service from System Resources Manager (). The retrieved results are displayed in the Output UI.
You can crosscheck the data by referring to the database tables.

The XForm is created for the Web service operation generated on the data source.

Attaching custom scripts to an XForm

To attach JavaScript files to an XForm to customize scripts:

1. [Open the XForm](#) in the XForms Designer.
2. Right-click the XForm and select **Properties**.
The Form dialog box opens.
3. Expand the **Javascript** pane, and type the URL of the JavaScript file.

To specify multiple JavaScript files for an XForm:

- Use .

The JavaScript files are attached to the XForm.

Downloading a file from a service

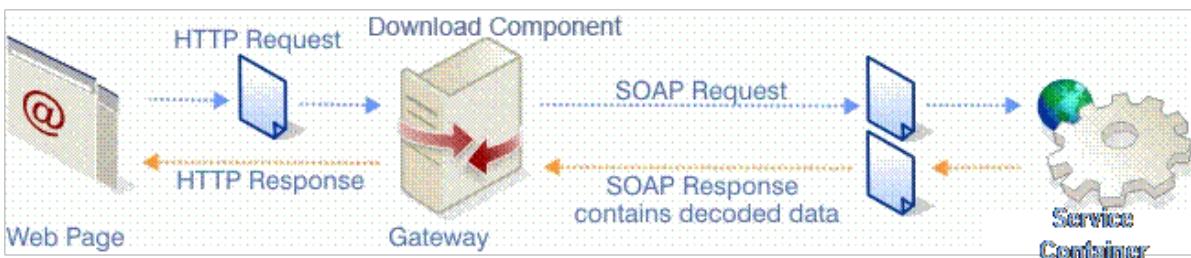
At times, you may have to download files from a service while it is running, through a Web browser. To support this process, a HTML page is required that is programmed to download the required file. AppWorks Platform provides you the required component to design such a page.

The component `download.htm` internally uses a template that is an HTML page. The component takes the content type and search path of the file that is to be downloaded. Additionally, it also takes the request SOAP message for the service from which the file will

be downloaded, and submits it to a Java class called `Download.wcp`. This Java class sends the request message to the AppWorks Platform Web Gateway, which in turn sends it to the appropriate service. The file is then successfully downloaded from the intended service.

You need to know which tag of the response SOAP message returned by the Gateway contains the file that is to be downloaded.

The following figure illustrates the way the download process works.



To download a file from a service:

1. Create a HTML page, fulfilling the following requirements:

- a. Add the script

```
<script type="text/javascript"
src='/cordys/wcp/application.js'></script>
```

to the page.

- b. Add the download component to the page as shown in the code:

```
<div cordysType="wcp.library.util.download" id='downloader' request =
sampleMessage.XMLDocument />
```

2. Create a Web Service Operation `GetResume` that will be sent to the service, and add it to the page as shown in the example.

```
<script id="sampleMessage" type="cordys/xml">
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>
<GetResume xmlns="http://schemas.cordys.com/DownloadDemo">
<EmployeeID>PARAMETER</EmployeeID>
</GetResume>
</SOAP:Body>
</SOAP:Envelope>
</script>
```

3. Set the following properties for the download component before you start downloading the file:

<code>contentType</code>	Specify the <code>contentType</code> of the file being downloaded. By default, this will be set to <code>application</code> or <code>octet-stream</code> . The following example sets the content type to a Microsoft Word file. <code>downloader.contentType = 'application/msword'</code>
--------------------------	--

searchPath	<p>Specify the exact node or tag of the content that is being downloaded in the response message that is returned from the service. In the following example, the content of the <Resume> tag, which is a word document, is being downloaded.</p> <pre>downloader.searchPath = '<Resume>';</pre>
------------	--

4. Add the `downloadFile` Web service operation to the page.

Developing the HTML page is complete.

5. Create a shortcut.

The URL for creating the shortcut is the URL for accessing the HTML page.

The HTML page to download a file from a service is created.

For more information on downloading a file from a service, see [Download in the AppWorks Platform API Guide](#).

Exporting data from an XGrid control to MS Excel sheet

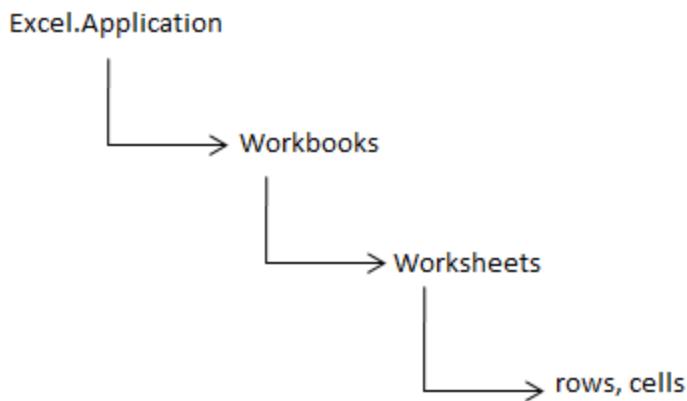
By using the XGrid control in an XForm, you can display data from any data model as a table. The tabular format makes it possible to represent data in various ways. While it is possible to perform simple manipulations such as sorting of data in an XGrid, if you need to perform complex data manipulations, you can export the data into an Excel sheet.

Before you begin:

- You must have the user interface (XForm) that contains an XGrid control with data in it.
- Ensure that appropriate security settings are set in the Internet Explorer browser to allow programming in Excel. For this, you must set the Initialize and script ActiveX controls not marked as safe option as Enable or Prompt.

The procedure to export data from an XGrid control to an Excel sheet follows. This procedure is applicable for the Internet Explorer browser only. It is not possible to enable Excel programming through JavaScript in other browsers.

To start programming in MS Excel, we must know the JavaScript objects that can be used. This refers to the hierarchy of excel objects and the APIs that are needed to write data to Excel. An Excel object can be represented as a basic hierarchical structure as shown.



The hierarchy consists of:

- **Application:** The top-most object that contains information about instances being executed and the various objects in excel.
- **Workbook:** An Excel document. Application has references to all workbooks.
- **Worksheet:** Sheet inside a workbook.
- **Rows, Cells:** Rows and cells present in a Worksheet that are used to manipulate data.

To start writing data to an excel sheet:

1. Create an object reference to **Excel.Application**, which gives access to Excel application.
 2. Add a Worksheet inside a Workbook.
 3. Add rows and cells into the Worksheet.
- For more information about MS Excel and how to perform the above actions, see the Microsoft Excel documentation.

While exporting data from the XGrid control to the Excel, the following APIs are used:

- <xgrid>.getColumn(<index>): Used to retrieve a column from the XGrid control. For each column, the header can be retrieved using thegetHeader()method, and<header>.getDescription()can be used to retrieve the header text.
- <xgrid>.getNumberOfRows(): Used to retrieve the number of rows in a header.
- <xgrid>.getCell(): Used to retrieve a cell (for given column and row) from the XGrid. The values in a cell can be retrieved using thegetValue()method.

4. Create a User Interface (XForm) that contains an XGrid control with data rendered in it. For information about the APIs that can be used to populate data from a data model in an XGrid control, see XGrid.
5. Add a Button control to the XForm and rename it appropriately. On clicking this button, users will be able to export data to Excel.
6. Add the following code to for the click event of the button control.

```

// opens Excel, and adds a worksheet inside
var excel = new ActiveXObject("Excel.Application");
excel.visible = true;
var sheet = excel.Workbooks.Add(1).Worksheets(1);

//displays the header
var numberofColumns = 0;
var column = table.getColumn(numberofColumns++);
while (column)
{
    sheet.cells(1, numberofColumns) = column.getHeader().getDescription();
    column = table.getColumn(numberofColumns++);
}

//displays the data
var numberOfRows = table.getNumberOfRows();
for (var I = 1; I < numberOfRows+1; I++)
{
    for (var j = 1; j < numberofColumns-1; j++)
    {
        sheet.cells(I+1, j+1) = table.getCell(j, I).getValue();
    }
}
application.inform("Data Exported to Excel Successfully");

```

Here, table denotes the ID of the XGRID control from which data must be exported into Excel.

If you have not set appropriate security settings as mentioned in the Pre-requisites section, you will get script errors. To avoid it, ensure the settings are correct. You can add a try-catch statement to view appropriate messages instead of a script error.

7. Save the user interface (XForm) and preview it.
The XGrid in the XForm displays and is pre-filled with data.
8. When you click the button, an Excel sheet opens and data is exported to it.
9. Save the excel sheet and use as required.

Handling globalization aspects in web application development

In a global scenario, application users across various countries speak different languages and are accustomed to local conventions for formatting and display of information. To be able to serve users better, you may need to globalize your applications for various countries.

Globalization includes the internationalization and translation of an application.

AppWorks Platform has the infrastructure necessary to support internationalization. You can also internationalize the AppWorks Platform infrastructure, including the Explorer and the Wizards. It is possible to translate the Explorer, its web components, and the web pages generated using the Web Generator to different languages such as Chinese and Hebrew.

AppWorks Platform is UTF8 enabled, that is it supports Multi-byte characters. The web pages generated by the Web Generator are also UTF8 enabled. Apart from this, AppWorks Platform provides the Web Components - translator and globalization, which help in internationalization.

The translator component enables the translation of AppWorks Platform web pages to other languages. It also enables the translation of the menu tree in the explorer to other languages. This component also enables localization of styles to suit requirements.

The globalization component enables the conversion of different units of measurement to the required format. Using this component, units such as date and time can be localized for internationalization. Globalization is also available as a Java class for enabling this conversion for back end applications.

It is also possible to have the LDAP entries and Metadata Explorer entries in Chinese and Hebrew.

To translate the explorer menu:

1. Create a dictionary file for the translation and store it under /cordys/wcp/admin/explorer.translation. This dictionary contains all possible terms that can be translated and their equivalents in other languages (such as Chinese and Hebrew). By default, AppWorks Platform provides a dummy file in this location. This dummy file has to be updated with the required content. As an example, see Demo code for Translator library, to see the usage of the dictionary by the translator component.
2. In the explorer.htm page, the translation file is already attached to the translator component. When the dictionary is updated with XML content, the page is refreshed with the change in language (similar to the regional settings of the Control Panel). The explorer is translated according to the translation text specified in the dictionary.
3. To provide a custom dictionary for the daemon page, or to provide translation for the new tree items added to the explorer tree, place the following line of code in the application page being opened.

```
\\" here "Menu" is the id of the application
var menuApplication= system.windows["Menu"];
\\menuApplication may not be running always, therefore ensure you
\\get the proper application before attaching the dictionary.
if (menuApplication)
{
    menuApplication.translator.appendDictionary(sURL);
}
```

Where, sURL denotes the URL of the translator document.

The Unified Feedback Objects (UFO) framework does not support translation.

4. Translate the AppWorks Platform wizards.

To do so, prepare a dictionary file as mentioned in Step 1. When the dictionary is ready, modify the wizard content needs as follows:

- a. Attach the translator component to the main wizard page.
- b. When the translator is loaded, the "onactive" event fires, which calls the function handler. The wizard properties such as caption and description are translatable by translating the text in the handler, as follows.

```
function translateElements()
{
    var itemCaptions = wizard.wizardData.selectNodes("./step/caption");
    for (var caption = itemCaptions.nextNode(); caption; caption =
itemCaptions.nextNode())
        caption.text = translator.translate(caption.text);
}
```

The code translates all the elements such as caption and description of the Wizard page to the corresponding language.

5. Translate Web pages.

To do so, use the Web Generator to generate web pages that are automatically UTF8 enabled, that is translatable. For already generated web pages,

- a. Include the translator component in the HTML page:

```
<div cordysType = "wcp.library.util.Translator" ></div>
```

For a translator component, an ID need not be specified. If not specified, the default identifier will be **translator**.

- b. For each of the tokens in the page that needs to be translated, include the translatable property whose value can be set to true as in the following <LABEL> tag.

```
<LABEL translatable="true">This is a text</LABEL>
```

- c. If the text that is to be translated is too long, it can be referenced in the dictionary file using a label ID. This is shown below for a menuitem inside the context menu.

```
<div menuitem translatable=true label_id="lbl1">This is a long text, which uses
a label_id for translation</div>
```

6. Localize date and time. Use the globalization component to localize date and time to suit the requirement.

To do so, add the following code in the HTML page:

```
<div "wcp.library.util.Globalization" id=globalID></div>
```

Where globalID denotes the ID of the component attached to the page.

The `getLocaleDate()` method of the Globalization component can be used to get the local equivalent of a given UTC (Universal Time Coordinated) date. It takes the date object as an input. For details, see Globalization in the *AppWorks Platform API Guide*.

The web application is globalized.

Reordering rows in a table

AppWorks Platform XForms provides the Table control to display data in a tabular form. The Table control is a data-bound control that can render data from an XML. The data displayed in the Table control depends upon the Web service operation used to retrieve it. For information about the Table control, see the Table topic in the *API Reference Guide*.

To modify the display order of rows at run time:

1. Open the [XForm](#) containing the Table control in the XForms Designer.
2. Add two navigation buttons (for example, Move Up and Move Down) to the XForm.
3. Right-click the Move Up button, and select **Properties**.
Alternatively, double-click the button.
The Button dialog box opens.
4. Click to expand the **Events** pane.
The Events pane displays the Click and Data Bind events.
5. Click  for the **Click** event.
The Script Editor opens.
6. Add the following code for the Click event of the Move Up button in the Script Editor.

```
function moveUp() { //get the current row index var index = DownloadsTable.getIndex(); var rows = DownloadsTable.getRows(); //if the row is the top row in table, return if (index == 1) return; var currentRow = rows[index-1]; //move the row position currentRow.parentNode.insertBefore(currentRow, currentRow.previousSibling); //set correct index DownloadsTable.setIndex(index-1); }
```

7. Similarly, open the Script Editor for the Move Down button using its property sheet, and add the following code for the On Click event.

```
function moveDown() { //get the current row index var index = DownloadsTable.getIndex(); var rows = DownloadsTable.getRows(); //if the row is the bottom row in table, return if (index == rows.length) return; var currentRow = rows[index-1]; //move the row position currentRow.parentNode.insertBefore(currentRow, currentRow.nextSibling.nextSibling); //set correct index DownloadsTable.setIndex(index+1); }
```

This action configures the Move Up and Move Down buttons with the table such that when you select a record and click the Move Up or Move Down button, the record moves up or down the table, respectively.

The procedure describes how to reorder the data displayed in the table. The reordering of data does not happen in conjunction with the pagination feature, and so does not affect the pagination of the table.

Uploading a file to a service

At times, you may have to upload files to a service while it is running, through a Web browser. To support this process, a HTML page is required that is programmed to upload the required file. AppWorks Platform provides you the required component to design such a page.

The `componentupload.htm` internally uses a template, which is an HTML page. The component takes the file name and the contents of the page. Additionally, it also takes the request SOAP message for the service to which the file will be uploaded, and submits it to a Java class called `Upload.wcp`. This Java class replaces the keywords `Upload:FileName1` and `Upload:FileContent1` in the request SOAP Message with the actual file name and the file content, and sends the request message to the AppWorks Platform Gateway. AppWorks Platform Web Gateway then sends it to the appropriate service. The file is successfully uploaded to the intended service.

Caution: Upload Gateway stores the uploaded files in a shared directory and passes the directory name to the service container to process the request. The shared directory is referred by the `com.eibus.web.tools.upload.UploadWritePath wcp.property`. It is possible that a hacker can continuously send upload requests where no service container is available to handle them. In such a case, the disk space will be full in no time. To avoid such a scenario, it is strongly recommended that you specify a size-limit on the shared directory.

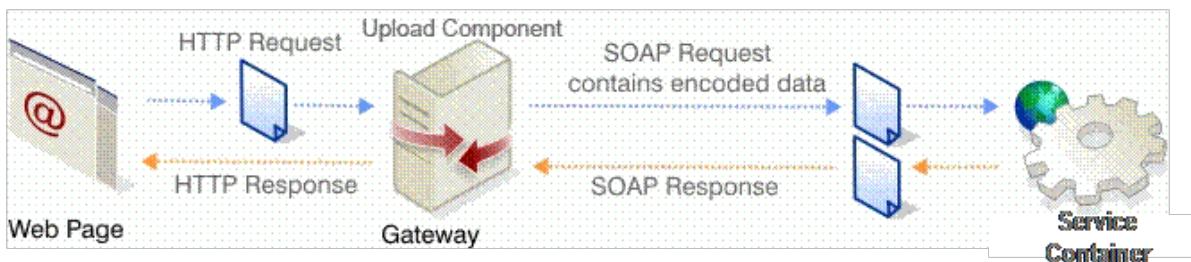
There are certain situations where you have to upload large files. In such situations, instead of uploading a file directly you can specify the path of the file in the `Upload:FilePath` property in the SOAP request. When this property is used, the uploaded file is stored at a temporary location given in the property `com.eibus.web.tools.upload.UploadWritePath` present in the `wcp.properties` file and the SOAP Request is sent to the SOAP service with the location of the temporary file.

The user in whose context the Webserver runs should have write permissions on the specified directory. Uploading large files leads to high memory consumption. The hardware configuration of your computer determines if the file that you are uploading is large. Certain computers can accommodate a file of size 50 MB but some computers may not be able to handle a file of size 4 MB too. Therefore, you can use this property depending on the hardware configuration of your computer.

The SOAP service can read the file from the temporary location.

The SOAP response from the gateway contains details of the file to be uploaded. You must know the exact element within the SOAP response that contains the details.

The following figure illustrates the way the upload process works.



To upload a file to a service:

1. Create an HTML page, fulfilling the following requirements:
 - a. Add the script `<script type="text/javascript" src='/cordys/wcp/application.js'></script>` to the page.
 - b. Add the upload component to the page as shown in the code:

```
<div cordysType="wcp.library.util.Upload" id="uploader" encode="true" showStatus="true" request = requestString.XMLDocument onupload = "onUploadHandler ()"/>
```
2. Create a SOAP message that will be sent to the service, and add it to the page as shown in the example:

```
<script id="requestString" type="cordys/xml">
  <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Body>
      <UpdateXMLObject xmlns="http://schemas.cordys.com/1.0/xmlstore">
        <tuple key="Upload:FileName1">
          <new>Upload:FileContent1</new>
        </tuple>
      </UpdateXMLObject>
    </SOAP:Body>
  </SOAP:Envelope>
</script>
```

In the SOAP Message, the keywords `Upload:FileName 1` and `Upload:FileContent 1` are included in places where the file name and file content are present. Also, the keyword `Upload:FileContent` should not be an attribute of an element in the SOAP message. If more than one file needs to be uploaded, the file count property of the component can be set to the number of files required. In such a case, the SOAP message should also be appropriately updated to include the keywords `Upload:FileName<n>` and `Upload:FileContent<n>` (where '*n*' represents the file count) to include all the files.

3. Add the browse Web service operation to the page.
This Web service operation provides the functionality for browsing the files. It can be called as many times as the number of files to be uploaded, using file numbers as parameters.

4. Add the uploadFile Web service operation to the page.
Developing the HTML page is complete.
5. Create a shortcut.
For more information on creating a shortcut, see the Creating a Shortcut using URL topic in the *AppWorks Platform Overview*.
The URL for creating the shortcut is the URL for accessing the HTML page.

The HTML page to upload a file from a service is created.

For more information on uploading a file to a service, see the Upload (AJAX Toolkit API) topic in the *AppWorks Platform API Reference*.

Using Inbox as a composite control

The Inbox composite control is used to display the Inbox in an XForm. This control enables you to display the tasks that are retrieved either from specific targets, sources, or instances of a source.

Before you begin:

- Ensure that the Inbox composite control that you want to refer is available in run time.
- Also, check if you have adequate permissions to access it and create the reference.

To use Inbox as a composite control:

1. In the Workspace Documents (Explorer), open <solution>, right-click <project> or <folder> and select **Add Runtime Reference > Other**.
The New Runtime Reference dialog box opens, displaying all the runtime documents.
2. Select **Composite Control**.
The Untitled Composite Control - Composite Control wizard appears.
3. Expand **Notification** folder, select **InboxView** and click **Next**.
The next screen of the window appears with the details of the Composite Control run-time document.
4. Provide additional details in **Additional Instructions** to install the run-time reference, and click **Finish**.
The Inbox composite control is displayed in the specified location.
5. Drag the Inbox composite control from the Workspace Documents window to an XForm.
The composite control is displayed on the XForm.
6. Double-click the Inbox control to set its properties.
A Properties sheet is displayed on the right.
7. Set the properties of the Inbox composite control in the Properties sheet as required.
The properties associated with the Inbox control and their descriptions follow.

ID	Provide the string that identifies the control on an XForm. If not specified, a unique ID is automatically generated.
----	---

Target	Target - Based on the target type selected, provide the relevant target that should be displayed in the Inbox control. If the target type is <ul style="list-style-type: none"> ■ role - Provide the DN of the role ■ team - Provide the ID of the Organizational Unit. ■ worklist - Provide the ID of the work list ■ user - Provide the DN of the user.
Target type	Target type can be a <ul style="list-style-type: none"> ■ role ■ team ■ worklist ■ user

Uploading a file using the Upload control

Many of your Web applications may require users to upload files to provide them as attachments, submit them for processing, or simply to save them to a server. Providing this upload functionality programmatically is a time consuming process. AppWorks Platform XForms provides an Uploadcontrol that can be easily added to an XForm for enabling the upload functionality at run time.

The Upload control enables you to upload a file to a server or another application connector. While adding a Upload control to an XForm, you need to provide the model to be used and a reference XPath to which data is to be uploaded. When a file is uploaded, the Upload control sends a SOAP request to the server or application connector, to which the file content must be uploaded.

The Upload control comprises an input field, a label, and a browse button that enables you to navigate to the file to be uploaded. The default label of the control is Attachment.

As an example, consider a Web page with upload functionality, which is used to upload and save employee photographs to the North Wind Employees table. The employee photographs need to be saved under the Photo column in the Employees table. Use the GetEmployeesObjects Web service, which is generated from the Employeesmodel.

To upload a file using the Upload control:

1. [Create an XForm](#), and drag the GetEmployeesObjects Web service from the **Insert** tab to it.
The Select Web Service Operation dialog box opens.
2. Select **Generate UI for Input message** or **Generate UI for Output message** if required, and click **OK**. In this case, select **Generate UI for Output message**.
Depending on the option selected, a user interface is automatically generated for the GetEmployeesObject Web service. If you do not select any of these options, no user interface is generated.

3. Drag the **Upload** control from the **Toolbox** tab to the **Designer Area**.
4. Double-click the control to display its property sheet. Alternatively, right-click and select Properties.
In the example, you can provide `photoUpload` as its ID. The Upload dialog box opens. The Model field displays models associated with the XForm.
5. Select the Model to be associated with the Upload control. In this case, select `EmployeesModel`.
6. In the Reference field, type the XPath where the data is to be uploaded. In this case, type `tns:Photo`, where `tns` refers to the namespace and `Photo` refers to the node in the update request.
7. Click  to save the changes made to the XForm.
8. At runtime, you can click **Browse** to navigate to, select an employee photograph, and click  to save the changes.

Using WS-AppServer custom class as an alternative to object template

Object Template has been deprecated in BOP 4.1. Hence, you cannot create a new Object Template from document creation wizard. However, if there are object templates used in the applications build prior to BOP 4.1, you can create and use WS-AppServer custom classes replicating the object template schema.

Similar to an Object Template, you can do the following with WS-AppServer custom class:

- Generate schema and schema fragments.
- Provide unique fields (analogous to the Special Attribute feature of Object Template)
- Generate Web services, use these Web services to fetch the instances matching the search criteria.
- Model rule(s), decision table(s), and data transformation(s).

This page illustrates the procedure and guidelines that are used to create a custom class and remodel the object template schema to WS-AppServer custom class.

Consider the following example that requires data transformation from SapOrder to BaanOrder schema. The objective of the Transformation application is to map common elements of these two structures.

SapOrder instance

```
<sap:SapOrder xmlns:sap="http://schemas.cordys.com/SapApp">
  <sap:OrderId/>
  <sap:OrderDate/>
  <sap:OrderedBy/>
  <sap:CustomerId/>
```

```
<sap:CustomerAddress/>
<sap:ShipmentDate/>
</sap:SapOrder>
```

BaanOrder instance

```
<baan:BaanOrder xmlns:baan="http://schemas.cordys.com/BaanApp">
    <baan:OrderIdentifier/>
    <baan:CustomerIdentifier/>
    <baan:OrderDate/>
    <baan:ShipmentDate/>
    <baan:OrderDetails/>
    <baan:ShipmentAddress/>
</baan:BaanOrder>
```

Data transformation can be created using the schema fragments that are generated by either of the following documents:

- Object Templates
- WS-AppServer Custom Class

This section contains the procedure on using the WS-AppServer custom class instead of an Object Template.

Using WS-AppServer custom class in the Transformation application

Prerequisites

Assumptions:

- User has Developer and Administrator role.
- User has a workspace and a project or has an existing CWS project.

To use WS-AppServer custom class in the Transformation application:

1. [Create a new WS-AppServer Package](#), for instance, SapApp has the namespace='http://schemas.cordys.com/SapApp'. Ensure that the namespace of this package is same as the namespace of the object template.
2. On the WS-AppServer Package toolbar, click . The Create Class from Object Layout dialog box opens.
3. Use the following object layout for the SapOrder schema.
You can use the Layout Tree tab to build the layout. If you are familiar with the WS-AppServer custom class layouts, you can directly use the layout. In this instance, all the elements are of type *string*. If required, you can use a relevant type.

SapOrder pure custom class layout

```
<SapOrder>
    <OrderId unique="true">string</OrderId>
    <OrderDate>string</OrderDate>
    <OrderedBy>string</OrderedBy>
    <CustomerId>string</CustomerId>
    <CustomerAddress>string</CustomerAddress>
    <ShipmentDate>string</ShipmentDate>
</SapOrder>
```

4. Click **OK**.
A custom class with the relevant attributes and methods is generated.
5. On the WS-AppServer Package toolbar, click on **Generate WebServiceInterface**.
A Web service generation wizard is displayed.
6. Select the custom class and follow instructions on the wizard.
At the end, a Web service Interface along with the Schema for SapOrder is generated.
7. Repeat the Steps 1 to 5 for creating BaanOrder. Provide the following details in the relevant fields:
 - a. WS-AppServer package name = BaanApp
 - b. WS-AppServer package namespace='http://schemas.cordys.com/BaanApp'
 - c. Custom class object layout

BaanOrder pure custom class layout

```
<BaanOrder>
    <OrderIdentifier unique="true">string</OrderIdentifier>
    <CustomerIdentifier>string</CustomerIdentifier>
    <OrderDate>string</OrderDate>
    <ShipmentDate>string</ShipmentDate>
    <OrderDetails>string</OrderDetails>
    <ShipmentAddress>string</ShipmentAddress>
</BaanOrder>
```

8. The schema for SapOrder and BaanOrder custom classes is generated along with the corresponding schema fragments.
9. [Create a new Data Transformation model](#), for instance, Sap_To_Baan_Transformation.
10. Select SapOrder schema fragment as the Source and BaanOrder schema fragment as the target in the transformation modeler.
11. Define the mappings.
12. Generate a Web service on Sap_To_Baan_Transformation with SapOrder schema fragment as the input and use it in your application.
For more information on generating a Web service on data transformation models, see [Generating a Web service on Data Transformation Model](#).

Guidelines to remodel Object Templates in the existing applications to WS-AppServer custom class

To have no break in the applications that used object templates, it is recommended to remodel the object templates to WS-AppServer custom classes. This section provides the guidelines to be followed while remodeling the object template into a WS-AppServer custom class.

To remodel Object Templates:

1. Create a WS-AppServer custom class that is same as the structure of the Object Template.
 - If an Object Template had any special attributes, ensure that those attributes are qualified as unique attributes while modeling a custom class from object layout.
2. If an Object Template is marked as Persistent, based on the object template structure, you can do any of the following:
 - a. If Object Template is a simple structure, that is, a structure that does not have any nested elements, you can create the database table representing this structure (ORM) and generate WS-AppServer class(es) from the database. In this case, you need not write any additional logic in the generated class to persist the instance in application database; it is taken care by WS-AppServer framework itself.
 - b. Use a Pure Custom class. You can use this regardless of simple or nested structure. However, if you use this option, you must write additional business logic to persist the instance data in the application database.
3. If the application is using Web services generated on object template, while generating Web services for custom class, you must ensure that name of the Web service and the names of the parameters match; that is, the custom class method and parameter names must match the parameter names of Web service generated on an object template.
4. Because the WS-AppServer framework interprets the Web service generated on a custom class, you must delink the namespace from CoBOC Service Group and attach the Web services generated on custom class to WS-AppServer Service Group.
5. If an object template had rules, decision table, or data transformations associated to it, then you have to update or remodel the respective rules, decision table or data transformation instances to use the schema fragments generated for WS-AppServer custom class and re-publish it.

Extending a connection in a WS-AppServer application

WS-AppServer exposes the IConnectionCustomizer interface and provides control of the Connection Object to enable usage of the extended properties supported by the respective database drivers. You can customize the SQL Connection Object of the respective database

driver so that its enhanced features can be used during a transaction. Applications need to implement this interface and use it during a transaction. For instance, Oracle supports opening a proxy session on top of an existing SQL connection. You can use the proxy session in AppWorks Platform by implementing the **IConnectionCustomizer** interface so that a proxy user can be used to perform transactions on a database at runtime.

IConnectionCustomizer interface

The **IConnectionCustomizer** interface is described as:

```
public interface IConnectionCustomizer { /** * This method is used to set additional properties to a standard connection * This connection can be any supported standard connection (For example, java.sql.Connection) * This method will be called internally when a request is executing using WS-AppServer * In such cases, this method will be called only once in a transaction * @param connection - a standard connection object * @return standard connection object after setting the properties * @throws ConnectionException */ public Object setExtendedConnectionProperties (Object connection) throws ConnectionException; \\ /** * This method is used to unset additional properties set on a standard connection * This method will be called internally when a request is executing using WS-AppServer * This method will be called only once in a transaction * Never close the standard connection here * @param connection - a standard connection object * @return standard connection object after unsetting the properties * @throws ConnectionException */ public Object unSetExtendedConnectionProperties (Object connection) throws ConnectionException; }
```

The following methods are present in this interface:

- **setExtendedConnectionProperties:** The WS-AppServer framework calls this method during a transaction before executing the actual request on a database. You need to customize the SQL Connection Object for the respective driver and set specific properties supported by the driver. The new connection returned by the application code that implements the **IConnectionCustomizer** interface will be used to perform the operation on the database.
- **unSetExtendedConnectionProperties:** The WS-AppServer framework calls this method once the operation is committed. You also need to remove the extended properties that were set.

To extend a connection:

1. Implement the **IConnectionCustomizer** interface.
2. To use the respective extended properties for the current transaction, pass the implemented class object to the WS-AppServer framework by using the `setConnectionCustomizer(IConnectionCustomizer connectionCustomizer)` API present on the `BusObjectManager` (BOM).
The WS-AppServer framework calls the `setExtendedConnectionProperties` method. The framework uses the returned connection during the transaction. After the transaction is committed, the WS-AppServer framework calls the `unSetExtendedConnectionProperties` method.

Example

In this example, there is table called CLOBTABLE in the database and objects are retrieved using a proxy user ("PROXY_TEST1"). A proxy session is opened in the `setExtendedConnectionProperties` method and is closed in the `unSetExtendedConnectionProperties` method.

```
public static BusObjectIterator<com.oracleproxylat.CLOBTABLE> getClobtableObjects
(String fromCLOB_ID, String toCLOB_ID) { //MyConnectionCustomizer is the class
implementing IConnectionCustomizer interface MyConnectionCustomizer
connectionCustomizer = new MyConnectionCustomizer(BSF.getObjectManager());
connectionCustomizer.setUseExtendedConnection(true); BSF.getObjectManager
().setConnectionCustomizer(connectionCustomizer); String queryText = "select * from
\"CLOBTABLE\" where \"CLOB_ID\" between :fromCLOB_ID and :toCLOB_ID"; QueryObject
query = new QueryObject(queryText); query.addParameter("fromCLOB_ID",
"CLOBTABLE.CLOB_ID", QueryObject.PARAM_STRING, fromCLOB_ID); query.addParameter
("toCLOB_ID", "CLOBTABLE.CLOB_ID", QueryObject.PARAM_STRING, toCLOB_ID);
query.setResultClass(CLOBTABLE.class); return query.getObjects(); } ;
```

When the `getClobtableObjects` method is executed, the `MyConnectionCustomizer` object is set on the `BusObjectManager` so that the framework will call the `setExtendedConnectionProperties` method and pass the `Connection Object` before executing the query on the database. Here, a proxy session is opened so that the WS-AppServer framework can make use of the opened proxy session. Once the query is executed and the transaction is complete, the framework will call the `unSetExtendedConnectionProperties` method. Here, the proxy session is closed and the original connection is returned so that the framework can put the connection back into the connection pool.

See the [database script](#) used and the [MyConnectionCustomizer](#) class on dev.ot.com, which implements the `IConnectionCustomizer` interface.

Note

- When a proxy session is opened, the Oracle driver closes the statement, prepared statement, and result set objects created prior to opening the proxy session. Therefore, the `reset()` API present on the `BusObjectManager` needs to be called to reset the cache before opening the proxy session. This ensures that the WS-AppServer does not end up using the objects that are already closed.
- You also need to reset the cache after closing the proxy session as objects created in between opening and closing of proxy sessions are also closed by the driver. Therefore, use the `reset()` API after closing the proxy session.
- If you do not use the `reset()` API in the above two scenarios, you will encounter closed statement errors in the logs.

Proxy user script

This topic presents the sample proxy user script used in the example to extend a connection in a WS-AppServer application.

```

//Create two Users PROXY_TEST1 and PROXY_TEST2 and grant PROXY_TEST1 to connect as
proxy for PROXY_TEST2

CREATE USER PROXY_TEST1 identified by PROXY_TEST1;
GRANT CONNECT, RESOURCE TO PROXY_TEST1;

CREATE USER PROXY_TEST2 identified by PROXY_TEST2;
GRANT CONNECT, RESOURCE TO PROXY_TEST2;

ALTER USER PROXY_TEST1 grant connect through PROXY_TEST2;

//Create table CLOBJECTABLE in PROXY_TEST2 schema
CREATE TABLE CLOBJECTABLE
(
  CLOB_ID      VARCHAR2(10 BYTE),
  CLOB_VALUE   CLOB
);

ALTER TABLE CLOBJECTABLE ADD (
  PRIMARY KEY
(CLOB_ID));

//Insert Data
Insert into CLOBJECTABLE
  (CLOB_ID, CLOB_VALUE)
Values
  ('1', '<Root><Test>Hello1</Test></Root>');

Insert into CLOBJECTABLE
  (CLOB_ID, CLOB_VALUE)
Values
  ('2', '<Root><Test>Hello2</Test></Root>');

//Grant Permissions on CLOBJECTABLE to proxy user.
GRANT SELECT, INSERT, DELETE, UPDATE on CLOBJECTABLE to PROXY_TEST1;

```

Developing applications with HTML5 SDK

Use the following topics to develop applications with AppWorks Platform HTML5 SDK:

- [Accessing the REST Web service](#)
- [Authenticating users with HTML5 SDK](#)
- [Building a Custom Approval page](#)
- [Building read-only mobile applications](#)
- [Building transactional mobile applications](#)
- [Displaying a KPI](#)
- [Displaying a geolocation on a mobile device](#)

- Displaying graphs on a mobile device
- Formatting a page per user preferences
- Implementing pagination
- Integrating with Google Maps using HTML5 SDK
- Integrating with WS-AppServer
- Retrieving the data
- Translating a page into a specific language
- Validating a page using jQuery validation plug-in
- Viewing Inbox tasks
- Working with the generic approval page

Accessing the REST Web service

You can retrieve the data from a REST Web service using `cordys.ajax` plug-in available in the AppWorks Platform HTML5 SDK. For more information on getting started, refer to [Getting Started with the SDK](#).

The following example Geo IP Locator uses the REST Web service for retrieving the details such as location and address of the specified IP address or host name. The location will be displayed in a Google Map. You can provide the IP address or Host Name and click the Show Location button on the page to view its location.

Example - Geo IP locator

[expand source](#)

The `initializeMap` function specified in the `done/callback` handler is invoked with the latitude and longitude values. This function uses the Google API for displaying the map and the Marker API for pointing the location. For more information on these APIs, see [Google Maps Javascript API V3 Reference](#).

This example is similar to the `restsample.htm` available in the `demo` folder of the AppWorks Platform HTML5 SDK. For more information on other Demo pages provided with the SDK, see [Sample Pages](#).

Authenticating users with HTML5 SDK

You can authenticate the user using AppWorks Platform HTML5 SDK by integrating with `cordys.authentication` plug-in. This plug-in displays the login page for authenticating the user. For information on getting started, see [Getting Started with the SDK](#).

The `Cordys.authentication` plug-in (or any method using it) handles the authentication based on the following:

- Web Server (Windows Authentication or Basic Authentication)
- AppWorks Platform SSO (Single Sign-On) implementation with cookies

Web server

If the user is not yet authenticated and the authentication type is WebServer, the user is prompted for the credentials.

AppWorks Platform SSO

If the cookies are not set before the authentication, the user is prompted for the credentials. When the login button is clicked, the cookies are set and the original page is displayed.

The following login pages are available:

- **Mobile Login:** The mobile login page is based on jQuery and jQuery Mobile and it works on all the web-kit browsers where jQuery and jQuery Mobile are available.
- **Standard Login:** If any other UI library is used instead of jQuery and jQuery Mobile, you need to create another login page with different HTML (the javascript must remain unchanged) and specify the URL as described in the Customization section. Otherwise, it displays the standard login page.

Building a Custom Approval page

You can build your own Custom Task Details page or Custom Approval page using the AppWorks Platform HTML5 SDK.

To build your own Custom Task Details page or Custom Approval page using the AppWorks Platform HTML5 SDK:

1. In the mobile landing page, select any task in the Tasks List page.
The corresponding details display in the Task Details page.
2. Customize the Task Details page to your requirements.

A sample `customapproval.htm` page (which is not a stand-alone page) is included in the AppWorks Platform HTML5 SDK for reference. For more details on Task Details or Task Approval pages, see [Sample Pages](#).

To use the Web page:

1. View the Web page as a task by providing its URL while creating an External User Interface.
For more information on creating an external user interface, see [Creating a User Interface Using a Web Page URL](#).
2. While creating an External User Interface, provide the Input Schema and Output Schema definitions. Add the required elements as an input or output to this Web page.
 - a. The required elements may be directly added as inputs in the XML Schema. Using the defined element, retrieve the required information from this Web page.

- b. In the code block `customapproval.htm`, OrderID is the input passed from the BPM and it is used to retrieve all the Order Details. The Sample Page for Reference section is used to describe the usage of the input elements.
- c. Access the input elements from the task details of the selected task as follows:


```
<task-details object>.TaskData.ApplicationData
```
- d. Include the required fields in the Output Schema definition, which must be used in the BPM or the other AppWorks Platform components.

Sample Input Schema and Output Schema definitions can be as follows.

Input schema definition

```
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://schemas.cordys.com/" xmlns=""
  xmlns:tns="http://schemas.cordys.com/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <element name="ApproveTask_IP" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
      <sequence>
        <element name="OrderID" type="xs:string"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </sequence>
    </complexType>
  </element>
</xsd:schema>
```

Output schema definition

```
<schema attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://schemas.cordys.com/approve"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://schemas.cordys.com/approve">
  <element name="ApproveTask_OP" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
      <sequence>
        <element name="ShipVia" type="xs:string"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

To access the task details:

The `taskId` is passed as a URL parameter while viewing the Task Details page.

- Retrieve the task details from the custom approval page using `getTaskDetails` API provided by the `cordys.workflow` plug-in.

To perform an action in a task:

1. Click the Actions button to display the task actions that can be performed on the task and supported by the HTML5 SDK Mobile Landing Page.
2. Define your custom actions on this Web page and execute the required action using the APIs offered by the `cordys.workflow` plug-in.

For example, you can have custom actions such as Approve or Reject, which can be used to perform the required action and complete the task. The required details as mentioned in the Output Schema definition can be added as `taskData` while performing the required action. The following is a code snippet, which performs the Complete action on the task and passes the detail `ShipVia` to the BPM.

```
$.cordys.workflow.completeTask(taskDetailModel.Task()[0], {
    ApproveTask_OP: {
        '@xmlns': 'http://schemas.cordys.com/approve',
        ShipVia: $('#fldShipVia').val()
    }
}, {dataType:'xml'}).done( function() {
    window.history.back();
});
```

To enable or disable fields according to the task status:

You can enable or disable the action buttons or elements according the current status of the task. This might be required in two different scenarios as follows:

- While navigating from the task list page to the details page
- While performing

1. Click Actions to call the method `refreshTaskDetailModel` in the Mobile Landing Page (`index.htm`).
2. Define the required action to be performed on the change of the task status.

The following is a code snippet to disable the elements, when the task state is not CREATED, ASSIGNED, or INPROGRESS:

```
function refreshTaskDetailModel(task) {
    var taskState = task.State;
    if( taskState == "CREATED" || taskState == "ASSIGNED" || taskState == "INPROGRESS" )
    {
        // Perform desired action like enable or disable the task action button
        // depending on the status
        //$("#controlgroup a").removeClass("ui-disabled");
    } else {
        //$("#controlgroup a").addClass("ui-disabled");
    }
}
```

To customize the page:

- Display the links to show the attachments that are available on the task and the location, if available.

For more information on getting the attachments and location, see [approvetask.htm](#).

Sample page

The following code snippet is a sample task details page that retrieves the order details, select the shipping method, and confirm the shipment.

customapproval.htm

```
DOCTYPE html>
<html>
    <head>
        <title>Approve</title>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>
        <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
        <link rel="stylesheet"
href="/cordys/thirdparty/jquery/jquery.mobile.structure.min.css" type="text/css" />
        <script src="/cordys/thirdparty/jquery/jquery.js"></script>
        <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"></script>
        <script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"></script>
        <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>
    </head>
    <body>
        <div data-role="page" id="approvePage">
            <div data-role="content" data-theme="c">
                <ul data-role="listview" data-inset="true">
                    <li data-role="heading" data-theme="c" data-mini="true">
                        <div>
                            <a class="ui-link-inherit">
                                <h3 class="ui-li-heading">Order Details</h3> <!-- NOMBV
-->

```

```
        </a>
    </div>
</li>
<li data-mini="true" id="detailView" data-bind="with: selectedItem"
readonly>
    <div>
        <div>
            <div><label for="fldOrderID" class="ui-input-text">OrderID</label></div>
            <div><input id="fldOrderID" readonly class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset" data-bind="value:OrderID" style="font-weight:normal"/></div>
        </div>
        <div>
            <div><label for="fldOrderDate" class="ui-input-text">Order Date</label></div>
            <div><input id="fldOrderDate" readonly class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset" data-bind="value:OrderDate" style="font-weight:normal"/></div>
        </div>
        <div>
            <div><label for="fldRequiredDate" class="ui-input-text">Required Date</label></div>
            <div><input id="fldRequiredDate" readonly class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset" data-bind="value:RequiredDate" style="font-weight:normal"/></div>
        </div>
        <div>
            <div><label for="fldName" class="ui-input-text">Freight</label></div>
            <div><input id="fldName" readonly class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset" data-bind="value:Freight" style="font-weight:normal"/></div>
        </div>
        <div>
            <div><label for="fldName" class="ui-input-text">Shipping Address</label></div>
            <div><input id="fldName" readonly class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset" data-bind="value:ShipName() + ', ' +"
```

```
on value="06">June</option><option value="07">July</option><option
value="08">August</option><option value="09">September</option><option
value="10">October</option><option value="11">November</option><option
value="12">December</option>
                </select>

                <label for="fldShippingDateDay">Day</label>
                <select name="fldShippingDateDay"
id="fldShippingDateDay">
                <option>Day</option>
                <option value="01">1</option>
                <option value="02">2</option>
                <option value="03">3</option>
                <option value="04">4</option>
                <option value="05">5</option>
                <option value="06">6</option>
                <option value="07">7</option>
                <option value="08">8</option>
                <option value="09">9</option>
                <option value="10">10</option>
                <option value="11">11</option>
                <option value="12">12</option>
                <option value="13">13</option>
                <option value="14">14</option>
                <option value="15">15</option>
                <option value="16">16</option>
                <option value="17">17</option>
                <option value="18">18</option>
                <option value="19">19</option>
                <option value="20">20</option>
                <option value="21">21</option>
                <option value="22">22</option>
                <option value="23">23</option>
                <option value="24">24</option>
                <option value="25">25</option>
                <option value="26">26</option>
                <option value="27">27</option>
                <option value="28">28</option>
                <option value="29">29</option>
                <option value="30">30</option>
                <option value="31">31</option>
            </select>
            <label for="fldShippingDateYear">Year</label>
```

```

user
    var Assignee = (taskDetails.Assignee) ?
taskDetails.Assignee.text : taskDetails.Assignee ; //NOMBV
    var taskStatus = taskDetails.State;
    if (!claimDone && Assignee == "") {
        claimDone = true;
        $.cordys.workflow.claimTask(taskDetails, {
            success: function() {
                completeTask();
            }
        });
        return;
    }
    //Completing the task and sending the new data back to the
bpm
$.cordys.workflow.completeTask(taskDetails, {
    ApproveTask_OP: {
        '@xmlns': 'http://schemas.cordys.com/approve',
        ShipVia: $("#"fldShipVia").val()
    }
}, {dataType:'xml'}).done( function() {
    window.history.back();
});
}).fail(function(error, statusText, errorThrown) {
    alert("Update Order Failed. Please Try again.");
});

} else {
    alert("Enter all the required Shipment Details.");
}
}

// This method is used to refresh the model once any of the action
occured through the menu item. Gets called from index.htm
function refreshTaskDetailModel(task) {
    var taskState = task.State;
    if( taskState == "CREATED" || taskState == "ASSIGNED" || taskState
== "INPROGRESS" ) {
        $("#"controlgroup_a").removeClass("ui-disabled");
    } else{
        $("#"controlgroup_a").addClass("ui-disabled");
    }
}

function getFormattedShippingDate(){
    if($("#fldShippingDateYear").val() != "Year" &&
$("#fldShippingDateMonth").val() != "Month" && $("#fldShippingDateDay").val() !=
"Day") {
        var formattedDate = $("#fldShippingDateYear").val() + "-" +
$("#fldShippingDateMonth").val() + "-" + $("#fldShippingDateDay").val() +
"T00:00:00.00";
        return formattedDate;
    } else {
}
}

```

In this Web page, the OrderID is passed from the BPM that is used to retrieve and display the order details. You can select the shipping method sent back to the BPM for updating the details. The task is completed when the user selects the confirm shipment option.

This Web page may not work independently, as it has dependencies with the other AppWorks Platform Components.

Building read-only mobile applications

You can build Read-only mobile applications on AppWorks Platform Web services using the cordys.model plug-in in the AppWorks Platform HTML5 SDK. For information on getting started, see [Getting Started with the SDK](#).

This plug-in uses [KnockoutJS](#) for rendering, which can be used for building transactional Web pages as well. For more information, see [Building Transactional Mobile Applications](#). If you are new to KnockoutJS, one of the best places to start is the Knockout Tutorial. You can also use the other rendering libraries such as jsRender to build the read-only model. For more information on using jsRender as the rendering library, [Retrieving the data](#).

Both the examples described in this topic are read-only. The following examples have been described in this topic:

- **Example 1:** (Read-only list of employees) - Displays a list of the employees from the Northwind Database
- **Example 2:** (Read-only list of employees with details) - Displays the details of each employee on clicking the items in the list.

Example 1

This example displays the list of employees with their Photograph, First Name, Last Name, Address, and City. This list is not editable. This example is similar to `employeeslist.htm` in the demo folder in the AppWorks Platform HTML5 SDK. For more information on other Demo pages shipped with the SDK, see [Sample Pages](#).

Read-only list of employees

```
<html>
  <head>
    <title>Employees</title>
    <meta content="width=device-width, initial-scale=1" name="viewport"/>
    <link href="/cordys/thirdparty/jquery/jquery.mobile.min.css"
rel="stylesheet"/>
    <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"/>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
```

```

type="text/javascript"/>
    <script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"/>
    <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
    <script type="text/javascript"> $(function() { // Create a new model
empModel = new $.cordys.model({ objectName: "Employees", // Name of the Business
Object context : $("#employeeList")[0], // Where the data has to be bound read: { // Settings for the read method namespace: "http://schemas.cordys.com/NW", dataType:
"json", method: "GetEmployeesObjects", // Parameters for the method parameters: { fromEmployeeID: "0", toEmployeeID: "99" } } }); // Call the read method. This would
fire the method with the namespace and parameters as specified in the read settings
above. empModel.read(); });
</script>
</head>
<body>
    <div data-role="page" id="mainPage">
        <div data-role="header" data-theme="b">
            <h1>Employees</h1>
        </div>
        <div data-role="content" data-theme="b">
            <ul data-bind="foreach:Employees" data-inset="true"
                data-role="listview" data-theme="c" id="employeeList">
                <li>
                    <!-- ko if: typeof(Photo) !=="undefined" && Photo-->
                    <img class="ui-li-thumb ui-corner-tl" data-bind="attr: {src:
'data:image/bmp;base64,' + Photo.substring(104) }"/>
                    <!-- /ko -->
                    <!-- ko if: typeof(Photo) ==="undefined" || (!Photo)-->
                    
                    <!-- /ko -->
                    <h3 class="ui-li-heading">
                        <span data-bind="text:FirstName"/>
                        <span data-bind="text:LastName"/>
                    </h3>
                    <p class="ui-li-desc" data-bind="text:Address"/>
                    <p class="ui-li-desc">
                        <span data-bind="text:City"/>
                        <span data-bind="text:Country"/>
                    </p>
                </li>
            </ul>
        </div>
    </body>
</html>

```

Example 2

You can customize the Example 1 by adding a click event for the employees list. The details of the selected Employee are displayed in the detail view, which is not editable. The example here is similar to `employeeswithdetails.htm` in the demo folder in the AppWorks Platform HTML5 SDK.

Read-only list of employees with details

```
<!DOCTYPE html> <html> <head> <title>Employees</title> <meta name="viewport" content="width=device-width, initial-scale=1"/> <link rel="stylesheet" href="/cordys/thirdparty/jquery/jquery.mobile.min.css" /> <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"></script> <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"></script> <script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"></script> <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script> <script type="text/javascript"> var empModel; // Reference to the model $(function() { ko.bindingHandlers.photo = { update: function(element, valueAccessor) { var value = valueAccessor(); if (value) { element.src = 'data:image/bmp;base64,' + value.substring(104); } else { element.src = "/cordys/html5/demo/images/defaultphoto.jpg"; } } }; // Create a new model empModel = new $.cordys.model({ objectName: "Employees", // Name of the Business Object context : document.body, // Where the data has to be bound to read: { // Settings for the read method namespace: "http://schemas.cordys.com/NW", dataType: "json", method: "GetEmployeesObjects", // Parameters for the method parameters: { fromEmployeeID: "0", toEmployeeID: "99" } } }); // Call the read method. This would fire the method with the namespace and parameters as specified in the read settings above. empModel.read(); }); // Called on clicking an Employee function selectEmployee(emp) { return function(data) { // Let us set the currently clicked item as the selected item to show in the detail page empModel.selectedItem(data); return true; } } </script> </head> <body> <div data-role="page" id="mainPage"> <div data-role="header" data-theme="b"> <h1>Employees</h1> </div> <div data-role="content" data-theme="b"> <ul id="employeeList" data-role="listview" data-theme="c" data-inset="true" data-bind="foreach:Employees"> <li> <a href="#detailsPage" data-transition="pop" class="ui-link-inherit" data-bind="click:selectEmployee($data)"> <img class="ui-li-thumb ui-corner-tl" data-bind="photo:Photo"/> <h3 class="ui-li-heading"><span data-bind="text:FirstName"> <span data-bind="text:LastName"></h3> <p class="ui-li-desc" data-bind="text:Address"> <span data-bind="text:Country"></p> </a> </li> </ul> </div> </div> <div data-role="page" id="detailsPage"> <div data-role="header" data-theme="b"> <a href="#" data-role="button" data-icon="back" data-rel="back">Back</a> <h1>Employee Details</h1> </div> <div data-role="content" data-theme="c"> <ul data-role="listview" id="detailView"> <li> <div data-bind="with: selectedItem"> <a class="ui-link-inherit" href="#"> <img class="ui-li-thumb ui-corner-tl" data-bind="photo:Photo"/> <h2 class="ui-li-heading" data-bind="text:EmployeeID"> <h3 class="ui-li-heading"><span data-bind="text:TitleOfCourtesy"> <span data-bind="text:FirstName"> <span data-bind="text:LastName"></h3> <p class="ui-li-desc" data-bind="text:Title"> </a> </div> </li> <li data-role="fieldcontain"> <div data-bind="with: selectedItem" id="employeeDiv"> <div> <label for="fldTitleOfCourtesy" class="select">Title of Courtesy</label> <select id="fldTitleOfCourtesy" data-bind="value:TitleOfCourtesy" data-native-menu="true" data-mini="true" data-theme="c" data-inline="true"> <option value="Mr.">Mr.</option> <option value="Mrs.">Mrs.</option> <option value="Ms.">Ms.</option> <option value="Dr.">Dr.</option> </select> </div> <div> <label for="fldFirstName" class="ui-input-text">First Name</label> <input type="text" id="fldFirstName" data-bind="value:FirstName" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldLastPhone" class="ui-input-text">Last Name</label> <input type="text" id="fldLastName" data-bind="value:LastName" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> </div> </li> </ul> </div> </div>
```

```

class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldTitle" class="ui-input-text">Title</label> <input type="text" id="fldTitle" data-bind="value:Title" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldPhone" class="ui-input-text">Phone</label> <input type="text" id="fldPhone" data-bind="value:HomePhone" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldAddress" class="ui-input-text">Address</label> <input type="text" id="fldAddress" data-bind="value:Address" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldCity" class="ui-input-text">City</label> <input type="text" id="fldCity" data-bind="value:City" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldCountry" class="ui-input-text">Country</label> <input type="text" id="fldCountry" data-bind="value:Country" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldNotes" class="ui-input-text">Notes</label> <textarea id="fldNotes" data-bind="value:Notes" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"></textarea> </div> </div> </ul> </div> </body> </html>

```

Building transactional mobile applications

You can build transactional mobile applications to perform the Create, Read, Update, and Delete (CRUD) operations using the `cordys.model` plug-in available in the AppWorks Platform HTML5 SDK. The `cordys.model` plug-in uses the KnockoutJS library to handle the transactions. For information on getting started, see [Getting Started with the SDK](#).

Using the Employees code block , you can build an Employee Details Web page with the CRUD operations based on the Employees table of the Northwind Database .

The following methods are used to perform CRUD operations as per the application requirement:

- You can add a new object to the model by invoking the `addBusinessObject` method. These objects will be inserted in the create or synchronize call.
- You must invoke the update method as all the modifications made to the view will be updated in the model.
- You can mark the objects to be deleted by invoking the `removeBusinessObject` method.

The synchronize operation has to be invoked to update the back-end for any changes (create, update, or delete). All the changes made in the view will be updated in the model and any changes made to the model will be updated in the view.

Each business object in the model is a Knockout Observable. This handles the change propagation from the view to the model and the model to the view.

Example

In this example, the data is displayed in the model using HTML. Invoke `ko.applyBindings` to bind your HTML with the model.

Employees

```
<!DOCTYPE html> <html> <head> <title>Employees</title> <meta name="viewport" content="width=device-width, initial-scale=1"/> <link rel="stylesheet" href="/cordys/thirdparty/jquery/jquery.mobile.min.css" /> <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"></script> <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"></script> <script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"></script> <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script> <script type="text/javascript"> var empModel; // Reference to the model var isViewMode; // True if Details Page is in view mode (Details page can be in view or edit mode). // Create a new model on page ready $(function() { empModel = new $.cordys.model({ context: document.body, objectName: "Employees", // Name of the Business Object // Common settings for all methods - read, create, update, delete, synchronize defaults: { namespace: "http://schemas.cordys.com/NW", dataType: "json" }, // Settings for the update/synchronize method update: { method: "UpdateEmployees" }, // Settings for the read method read: { method: "GetEmployeesObjects", parameters: { fromEmployeeID: "0", toEmployeeID: "99" } } }); // Call the read method. This would fire the method with the namespace and parameters as specified in the read settings above. empModel.read(); }); // Called on clicking an Employee function selectEmployee(emp) { return function(data) { // Let us set the currently clicked item as the selected item to show in the detail page empModel.selectedItem(data); // Set the Details Page to View Mode isViewMode = true; return true; } } // Called on confirmation of the Delete Dialog function deleteSelectedEmployee() { // Remove the Business Object empModel.removeBusinessObject(empModel.selectedItem()); // Synchronize the changes with the backend empModel.synchronize(); return true; } // Called on clicking the Add Button function addEmployee(){ // Create a new Business Object var newEmployee = empModel.addBusinessObject({Address: "",BirthDate: "",City: "",Country: "",Extension: "",FirstName: "",HireDate: "",HomePhone: "",LastName: "",Notes: "",Photo: "",PhotoPath: "",PostalCode: "",Region: "",ReportsTo: "",Title: "",TitleOfCourtesy: ""}); // Set the newly added Business Object as the currently selected item empModel.selectedItem(newEmployee); isViewMode = false; return true; } </script> </head> <body> <div data-role="page" id="mainPage"> <div data-role="header" data-theme="b"> <a style="display:none"></a> <h1>Employees</h1> <a id="btnAdd" data-role="button" data-icon="add" href="#detailsPage" data-bind="click:addEmployee">Add</a> </div> <div data-role="content" data-theme="b"> <ul id="employeeList" data-role="listview" data-split-icon="delete" data-split-theme="d" data-theme="c" data-inset="true" data-bind="foreach:Employees"> <li> <a href="#detailsPage" data-transition="pop" class="ui-link-inherit" data-bind="click:selectEmployee($data)"> <img class="ui-li-thumb ui-corner-tl" data-bind="attr: {src: (Photo && Photo()) ? 'data:image/bmp;base64,' + Photo().substring(104) : '/cordys/html5/demo/images/defaultphoto.jpg' }"/> <h3 class="ui-li-heading"><span data-bind="text:FirstName">&ampnbsp</span> <span data-bind="text:LastName"></h3> <p class="ui-li-desc" data-bind="text:Address"></p> <p class="ui-li-desc"><span data-bind="text:City">&ampnbsp</span> <span data-bind="text:Country"></p> </a> <a href="#deleteConfirmation" data-rel="dialog" data-transition="slideup" data-bind="click:selectEmployee($data)">Delete</a> </li> </ul> </div> </div>
```

```

role="page" id="deleteConfirmation"> <div data-role="header" data-theme="e">
<h1>Delete Employee?</h1> </div> <div data-role="content" data-theme="d" data-
bind="with: selectedItem"> <h4>Are you sure you want to delete employee<span data-
bind="text: ' ' + FirstName() + ' ' + LastName() + ''>?</h4> <a data-
role="button" data-inline="true" data-rel="back" data-theme="b" data-
bind="click:deleteSelectedEmployee">Yes</a> <a data-role="button" data-inline="true"
data-rel="back" data-bind="click:revertChanges">No</a> </div> </div> <div data-
role="page" id="detailsPage"> <div data-role="header" data-theme="b"> <a
href="#mainPage" data-role="button" data-icon="back" data-rel="back" data-
bind="click:revertChanges">Back</a> <h1>Employee Details</h1> <a data-role="button"
id="btnEdit">Edit</a> </div> <div data-role="content" data-theme="c"> <ul data-
role="listview" id="detailView"> <li> <div data-bind="with: selectedItem"> <a
class="ui-link-inherit"> <img class="ui-li-thumb ui-corner-tl" data-bind="attr:
{src: (Photo && Photo()) ? 'data:image/bmp;base64,' + Photo().substring(104) :
'cordys/html5/demo/images/defaultphoto.jpg' }"/> <!-- ko if: typeof
(EmployeeID)!=="undefined"--> <h2 class="ui-li-heading" data-
bind="text:EmployeeID"></h2> <!-- /ko --> <h3 class="ui-li-heading"><span data-
bind="text:TitleOfCourtesy">&ampnbsp<span data-bind="text:FirstName">&ampnbsp<span data-
bind="text:LastName"></h3> <p class="ui-li-desc"><span data-bind="text:Title"></p>
</a> </div> </li> <li data-role="fieldcontain"> <div data-bind="with: selectedItem"
id="employeeDiv"> <div> <label for="fldTitleOfCourtesy" class="select">Title of
Courtesy</label> <select id="fldTitleOfCourtesy" data-bind="value:TitleOfCourtesy"
data-native-menu="true" data-mini="true" data-theme="c" data-inline="true"> <option
value="Mr.">Mr.</option> <option value="Mrs.">Mrs.</option> <option
value="Ms.">Ms.</option> <option value="Dr.">Dr.</option> </select> </div> <div>
<label for="fldFirstName" class="ui-input-text">First Name</label> <input
type="text" id="fldFirstName" data-bind="value:FirstName" class="ui-input-text ui-
body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldLastName"
class="ui-input-text">Last Name</label> <input type="text" id="fldLastName" data-
bind="value:LastName" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div> <label for="fldTitle" class="ui-input-text">Title</label>
<input type="text" id="fldTitle" data-bind="value:Title" class="ui-input-text ui-
body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldPhone"
class="ui-input-text">Phone</label> <input type="text" id="fldPhone" data-
bind="value:HomePhone" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div> <label for="fldAddress" class="ui-input-text">Address</label>
<input type="text" id="fldAddress" data-bind="value:Address" class="ui-input-text
ui-body-b ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldCity"
class="ui-input-text">City</label> <input type="text" id="fldCity" data-
bind="value:City" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/>
</div> <div> <label for="fldCountry" class="ui-input-text">Country</label> <input
type="text" id="fldCountry" data-bind="value:Country" class="ui-input-text ui-body-b
ui-corner-all ui-shadow-inset"/> </div> <div> <label for="fldNotes" class="ui-
input-text">Notes</label> <textarea id="fldNotes" data-bind="value:Notes" class="ui-
input-text ui-body-b ui-corner-all ui-shadow-inset"></textarea> </div> </div> </li>
</ul> </div> <script type="text/javascript"> // Add Click handler to the Edit
Button on Page Ready $(function(){ $("#btnEdit").bind("click", function () { if
(isEditMode) { // We are switching to Edit Mode. Let us change the button's text to
Done and enable all the fields $(".ui-btn-text", this).text("Done"); $("#detailView
:input").removeAttr('disabled'); $("select", "#detailView").removeAttr('disabled');
$("#fldTitleOfCourtesy").select().focus(); } else { // We are switching to View
Mode. Let us change the button's text to Edit and disable all the fields $(".ui-btn-

```



This example is similar to the `employeeswithupdate.htm` Web page in the demo folder in the AppWorks Platform HTML5 SDK. For more information on other Demo pages in the AppWorks Platform HTML5 SDK, see [Sample Pages](#).

To work with the Employees page:

1. To add an employee:
 - a. Click **Add** and provide the required details in the Employee Details page.
 - b. Click **Done** to finish or click **Back** to cancel.
2. To edit the employee details:
 - a. Select the Employee you want to edit.
The Details page is displayed, where all the fields are disabled.
 - b. The Details page is displayed in the following two modes:
 - **View mode:** You can only view the employee details.
 - **Edit mode:** Click **Edit** to edit the employee details.
 - c. Click **Done** to finish or click **Back** to cancel.
3. To delete the employee details, do the following:
 - Click  to delete an employee.
The record is deleted after a confirmation.

Displaying a KPI

This topic describes displaying a KPI on a mobile device using the HTML5 SDK by integrating with the Fusion Charts.

The following code in the example retrieves the Employee details for the given Employee ID from the Northwind Database. You can invoke the KPI Web service using `$.cordys.ajax()` plug-in to retrieve the information related to the business measure and can display an Angular Gauge with it.

The method name is defined as the KPI Web service name in the code.

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>KPI</title>
<meta name="viewport" content="width=device-width, initial-scale=1"/>
    <link rel="stylesheet" href="/cordys/thirdparty/jquery/jquery.mobile.min.css" />
    <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"></script>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"></script>

    <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>
    <script type="text/javascript" src="/cordys/wcp/flash/fusion/fusioncharts.js"></script>
        <script type="text/javascript" src="/cordys/wcp/flash/fusion/FusionCharts.jqueryplugin.js"></script>
            <script src="/cordys/html5/knockout/knockout-2.1.0.js" type="text/javascript"></script>

<script type="text/javascript">
    $(function () {
        // get the employees
        $.cordys.ajax({
            method: "KPI3",
            namespace: "http://schemas.cordys.com/KPI3Webservice",
            parameters: {
                "cache": "false",
                "ns0:NewBM": {
                    "@xmlns:ns0":
                    "http://schemas.cordys.com/NewBMWebservice",
                    "ns1:GetEmployeesObject": {
                        "@xmlns:ns1": "http://schemas.cordys.com/NW",
                        "ns1:EmployeeID": "4"
                    }
                }
            },
            dataType: 'json', // the xml result will be converted into js
            objects
            success: function (data) {
                setChartData(data);
            }
        });
        function setChartData(data) {
            var colorArray = [];
            var lowerlimit;
            var upperlimit;
            //Create color array and assign limit values
            for (var i = 0; i < data.metricdata.rangegroup.range.length;

```

```

i++)
{
    if(i==0)
    {
        lowerlimit = data.metricdata.rangegroup.range[i]
    ["@lowerlimit"];
        upperlimit = data.metricdata.rangegroup.range[i]
    ["@upperlimit"];
    }
    else
    {
        if(parseFloat(lowerlimit) > parseFloat
(data.metricdata.rangegroup.range[i]["@lowerlimit"]))
            lowerlimit = data.metricdata.rangegroup.range[i]
    ["@lowerlimit"];
        if(parseFloat(upperlimit) < parseFloat
(data.metricdata.rangegroup.range[i]["@upperlimit"]))
            upperlimit = data.metricdata.rangegroup.range[i]
    ["@upperlimit"];
    }
    colorArray.push({
        'minvalue': data.metricdata.rangegroup.range[i]
    ["@lowerlimit"],
        'maxvalue': data.metricdata.rangegroup.range[i]
    ["@upperlimit"],
        'color': data.metricdata.rangegroup.range[i]["@color"]
    });
}
// Create complete chart data with schema required for Fusion
Chart
var chartData = {
    "chart": {
        "lowerlimit": lowerlimit,
        "upperlimit": upperlimit,
        "lowerlimitdisplay": lowerlimit,
        "upperlimitdisplay": upperlimit,
        "gaugestartangle": "180",
        "gaugeendangle": "0",
        "palette": "1",
        "numbersuffix": data.metricdata.rangegroup
    ["@unitofmeasure"],
        "tickvaluedistance": "20",
        "showvalue": "1"
    },
    "colorrange": {
        "color": colorArray
    },
    "dials": {
        "dial": [
            {
                "value": data.metricdata.processedvalues.value,
                "rearExtension": "2"
            }
        ]
    }
}

```

The callback function, `setChartData()`, is used to retrieve the business measure value, the lower limit value, and the upper limit value from the KPI Web service along with the color range details. After the required values are retrieved from the KPI Web service, you can create the json object for rendering the KPI in the Angular Gauge.

Business Activity Monitor (BAM) provides the option to monitor and capture critical information pertaining to a business process or an activity using the KPIs.

Displaying a geolocation on a mobile device

You can retrieve the position of the user on a mobile device using the HTML5 SDK Geolocation API, and display the position by integrating with Google Maps.

The following example retrieves and displays the position of a user on a mobile device using the Geolocation API. The user can subscribe to or unsubscribe from the location changes.

The following steps are involved in this example:

1. Including the script files
2. Defining the events
3. Retrieving the position of the user
4. Subscribing to the position changes
5. Unsubscribing from the position changes
6. Marking the location on Google Map

To include the script files:

- Include the following script files as displayed in the Scripts code block:
 - `$.cordys.mobile` plug-in to use the Geolocation APIs.
 - Google Maps for displaying the position of the user on a Google Map.

Scripts

```
<script src="/cordys/html5/plugins/cordys.mobile.js"></script>
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=true"></script>
```

To define the events:

1. Retrieve the position of a user:
 - Attach a click event for **My current location** to retrieve and display the position of the user in the Google Map.

2. Subscribe to the position changes:

- Attach a click event for **Mark my location** to subscribe to the position changes of the user.

3. Unsubscribe from the position changes:

- Attach a click event for **Stop marking my location** to unsubscribe from the position changes of the user.

To retrieve the position of the user:

- Use `$.cordys.mobile.geolocation.getCurrentPosition` in the `findPosition()` method for retrieving the current position of the user as displayed in the following code block.

```
function findPosition() {
    try {
        $.cordys.mobile.geolocation.getCurrentPosition(function
(position) {
            markGooglemap(position);
        }, function(parameters) {
            alert("failed: " + parameters.message);
        }, {
            maximumAge: 3000, timeout: 5000, enableHighAccuracy: true
        });
    }catch (e) {
        alert("error catched: " + e);
    }
}
```

To retrieve the position of the user:

- Use `$.cordys.mobile.geolocation.getCurrentPosition` in the `findPosition()` method for retrieving the current position of the user as displayed in the following code block.

Find position

```
function findPosition() {
    try {
        $.cordys.mobile.geolocation.getCurrentPosition(function
(position) {
            markGooglemap(position);
        }, function(parameters) {
            alert("failed: " + parameters.message);
        }, {
            maximumAge: 3000, timeout: 5000, enableHighAccuracy: true
        });
    }catch (e) {
```

```

        alert("error catched: " + e);
    }
}

```

To subscribe to the position changes:

- Use `$.cordys.mobile.geolocation.subscribeWatchPosition` in the `watchPosition()` method for subscribing to the position changes of the user as displayed in the following code block.

```

function watchPosition() {
    try {
        $.cordys.mobile.geolocation.subscribeWatchPosition(function(position) {
            markGooglemap(position);
        }, function(parameters) {
            alert("failed: " + parameters.message);
        }, {
            maximumAge: 3000, timeout: 5000, enableHighAccuracy: true
        });
    }catch (e) {
        alert("error catched: " + e);
    }
}

```

To unsubscribe from the position changes:

- Use `$.cordys.mobile.geolocation.unsubscribeWatchPosition` in the `stopWatchPosition()` method for unsubscribing from the position changes of the user as displayed in the following code block.

Stop watching position

```

function stopWatchPosition() {
    try {
        $.cordys.mobile.geolocation.unsubscribeWatchPosition();
        alert("Watching is stopped.");
    }catch (e) {
        alert("error catched: " + e);
    }
}

```

To mark the location on Google Map:

- Use the Google Maps API in the `markGoogleMap()` method for displaying the current position of the user in the Google Map as displayed in the following code block:

Mark location

```

function markGooglemap(position) {
    if(position.coords) {
        var coords = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
        var options = {
            zoom: 15,
            center: coords,
            mapTypeControl: false,
            navigationControlOptions: {
                style: google.maps.NavigationControlStyle.SMALL
            },
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        var map = new google.maps.Map(document.getElementById
("googleMap"), options);
        var marker = new google.maps.Marker({
            position: coords,
            map: map,
            title:"You are here!"
        });
    }else
        alert("Not able to retrieve your current location.");
}

```

See the following code block for more information on this example.

Geolocation

```

<!DOCTYPE html>
<html>
<head>
    <title>Find your location</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
    <link rel="stylesheet"
href="/cordys/thirdparty/jquery/mobile.structure.min.css" type="text/css" />
    <script src="/cordys/thirdparty/jquery/jquery.js"></script>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"></script>

    <script src="/cordys/html5/cordys.html5sdk.js"></script>
    <script src="/cordys/html5/plugins/cordys.mobile.js"></script>
    <script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=true"></script>
</head>

```

```

<body>
    <div data-role="page" id="positionPage">
        <div data-role="header" data-theme="d">
            <h1>Find your location</h1>
        </div>
        <div data-role="content" data-theme="d">
            <button id="btnpos" type="submit" data-theme="b">My current
location</button>
            <button id="btncancelwatchPosition" type="submit" data-theme="b">Mark my
location</button>
            <button id="btncancelwatchPosition" type="submit" data-theme="b">Stop
marking my location</button>
            <div id="googleMap" style="width:100%;height:250px;"></div>
        </div>
        <script>
            $("#positionPage").on("pageinit", function() {
                $("#btnpos").bind("click", findPosition);
                $("#btncancelwatchPosition").bind("click", cancelwatchPosition);
                $("#btncancelwatchPosition").bind("click", stopWatchPosition);
            });
            function findPosition() {
                try {
                    $.cordys.mobile.geolocation.getCurrentPosition(function
(position) {
                        markGooglemap(position);
                    }, function(parameters) {
                        alert("failed: " + parameters.message);
                    }, {
                        maximumAge: 3000, timeout: 5000, enableHighAccuracy: true
                    });
                } catch (e) {
                    alert("error caught: " + e);
                }
            }
            function cancelwatchPosition() {
                try {
                    $.cordys.mobile.geolocation.unsubscribeWatchPosition(function
(position) {
                        markGooglemap(position);
                    }, function(parameters) {
                        alert("failed: " + parameters.message);
                    }, {
                        maximumAge: 3000, timeout: 5000, enableHighAccuracy: true
                    });
                } catch (e) {
                    alert("error caught: " + e);
                }
            }
            function stopWatchPosition() {
                try {

```

```

        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById
("googleMap"), options);
    var marker = new google.maps.Marker({
        position: coords,
        map: map,
        title:"You are here!"
    });
} else
    alert("Not able to retrieve your current location.");
}
</script>
</div>
</body>
</html>

```

Displaying graphs on a mobile device

You can display graphs on a mobile device using the HTML5 SDK by integrating with charting toolkits such as Fusion Charts or Google Visualization.

To display graphs on a mobile device, do one of the following:

- To view the data in a graph, use the `$.cordys.ajax` plug-in to retrieve the data and then bind the data to the graph.
- To display the same data and allow modification, use the `$.cordys.model` plug-in and bind it with the data and the graph.

The following examples illustrate the usage of Google Visualization and Fusion Charts respectively to display the data.

Example 1

The following code snippet retrieves all the customers from the Northwind Database and displays a pie chart of the number of customers by their country.

Displaying graphs using Google Visualization

```

<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Customers By Region</title>
        <meta content="width=device-width, initial-scale=1" name="viewport"/>
        <link href="/cordys/thirdparty/jquery/jquery.mobile.min.css"

```

```

rel="stylesheet"/>
<script src="/cordys/thirdparty/jquery/jquery.js"/>
<script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"/>
<script src="/cordys/wcp/flash/fusion/fusioncharts.js"
type="text/javascript">
<script
    src="/cordys/wcp/flash/fusion/FusionCharts.jqueryplugin.js"
type="text/javascript"/>
<script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"/>
<script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
<script type="text/javascript">
$(function() {
    // Create a new model
    customerModel = new $.cordys.model({
        objectName: "Customers", // Name of the Business Object
        context: $("#customersList")[0], // Where to bind the data to
        isReadOnly:false,
        read: {
            // Settings for the read method
            async:false,
            namespace: "http://schemas.cordys.com/NW",
            dataType: "json",
            method: "GetCustomersObjects",
            // Parameters for the method
            parameters: {
                fromCustomerID: "aaaaaaaa",
                toCustomerID: "zzzzzzzz"
            }
        }
    });
}

// Call the read method. This fires the method with the namespace
and parameters as specified in the read settings above.
customerModel.read().done( function(){
    var customersByCountry = {};
    var customers = customerModel.Customers();

    // Create a collection of the customers by country
    for (var customerKey in customers)
    {
        var customer = customers[customerKey];
        var country = customer.Country();
        if (! customersByCountry[country]){
            customersByCountry[country] = [customer];
        }
        else{
            customersByCountry[country].push(customer);
        }
    }
}

```

```

Array.push({'label':country, 'value':customersByCountry[country].length, link:"j-
setCurrentCountry-"+ country});
}

// Create complete chart data with schema required for Fusion
Chart
var chartData = {
    "data" : dataArray
}

// Create the Chart
$("#chartContainer").insertFusionCharts({type: "Pie2D",
dataSource: chartData, dataFormat: "json", width: "100%", height: "400px", id:
"myChartID", renderer:'javascript'});
});

// Create an Observable for the selected country so that KO re-renders when
a different country is selected
var selectedCountry = ko.observable();
// Called on clicking a Country in the Chart. Sets the current country
function setCurrentCountry(value){
    selectedCountry(value);
    return true;
}
</script>
</head>
<body>
<div data-role="page" id="mainPage">
    <!-- header -->
    <div data-role="header" data-theme="b">
        <h1>Customers By Country</h1>
    </div>
    <!-- content area -->
    <div data-role="content" data-theme="b">
        <div id="chartContainer" style="height:40%"/>
        <ul data-bind="foreach:Customers" data-inset="true"
            data-role="listview" data-theme="c" id="customersList">
            <!-- ko if: selectedCountry() !=="undefined" && selectedCountry
() === Country()-->
            <li>
                <h3 class="ui-li-heading" data-bind="text:CompanyName"/>
                <p class="ui-li-desc">
                    <span data-bind="text:City"/>
                    <span data-bind="text:Country"/>
                </p>
                <p class="ui-li-desc" data-bind="text:Phone"/>
                <p class="ui-li-desc" data-bind="text:Fax"/>
                <p class="ui-li-desc" data-bind="text:ContactName"/>
            </li>
            <!-- /ko -->
        </ul>
    </div>

```

Example 2

The following code snippet retrieves all the products from the Northwind Database and displays a graph with the number of available units of each product in a bar chart.

Displaying graphs using Fusion Charts

```
<html>
  <head>
    <title>Products</title>
    <meta content="width=device-width, initial-scale=1" name="viewport"/>
    <link href="/cordys/thirdparty/jquery/jquery.mobile.min.css"
rel="stylesheet"/>
    <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"/>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
type="text/javascript">
    <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
    <script src="http://www.google.com/jsapi" type="text/javascript"/>
    <script type="text/javascript">
      google.load('visualization', '1', {packages: ['corechart']});
```

```
      var productsModel;
      function drawProductChart() {
        productsModel = new $.cordys.model({
          objectName: "Products",
          read: {
            namespace: "http://schemas.cordys.com/NW",
            dataType: "json",
            method: "GetProductsObjects",
            parameters: {
              fromProductID: "0",
              toProductID: "99999"
            }
          }
        });
        productsModel.read().done( function(products) {
          var data = new google.visualization.DataTable();
          data.addColumn('string', 'Product');
          data.addColumn('number', 'In Stock');
          $.each(products, function(p) {
            data.addRow([this.ProductName, parseInt
(this.UnitsInStock)]);
          });
          new google.visualization.BarChart
(document.getElementById('visualization')).draw(data,
          {title:"Products in stock",
           width:"100%", height:"100%",
            chartArea: {width:"70%",

```

```

        left:"50%",height:"90%"},

        vAxis: {title: "Product", textStyle:
{fontSize:10}},fontSize:10,
        hAxis: {title: "In Stock"},legend: {position:"left"}}
    );
});

}

google.setOnLoadCallback(drawProductChart);
</script>
</head>
<body>
<div data-role="page" id="myPage">
<div data-role="header" data-theme="b">
<h1>Products</h1>
</div>
<div data-role="content" data-theme="d">
<!-- Content comes here -->
<div id="visualization" style="width: 100%; height: 2000px"/>
</div>
</div>
</body>
</html>

```

Note

- Example 2 uses the HTML version of the Fusion Charts. If you want to display the Flash version, remove the renderer:"javascript" option while invoking the insertFusionCharts() API.
- You can click any country to display the customers from that country in a list.
- The supported version of the fusion chart is 3.2.4. See the [fusion chart documentation](#) for more information on the various chart types and formats.

Note

- Research Android browser version support for Google Visualization and Fusion Charts. Other browsers for the Android mobiles, such as Mozilla Firefox and Google Chrome support the charts irrespective of the Android version.
- The pages used in the examples are also available in the demo folder in the project sources.

Formatting a page per user preferences

You can format the pages you create according to your preferences. The procedure is explained using an example. This example is based on Orders and Order Details tables from the Northwind Database. It uses KnockoutJS for rendering, which can be used for building transactional Web pages as well. See [Building transactional mobile applications](#) for more

information. If you are new to [KnockoutJS](#), it is recommended to start with the [Knockout Tutorial](#).

This page displays all the orders in a list view. You can click on any order to view its details and view the order items by clicking the Orders Items button, which is displayed in the order details page.

The following fields are formatted according to the user preferences using the `cordys.formatting` plugin.

Field	Type
Discount	Decimal
Freight	Currency
Order Date	Date (MM/DD/YYYY HH:mm:ss)
Quantity	Number
Required Date	Date Time (YYYY/MM/DD HH:mm:ss)
Shipped Date	Date (MM/DD/YYYY)

The following steps are involved in this example:

1. Defining models
2. Adding the `data-bind` attribute
3. Reading models
4. Working with Orders page

To define the models:

- Define orders and order details model that are used for reading along with their corresponding order items.

Defining Order and Order Details Model

```
ordersModel = new $.cordys.model({
    objectName: "Orders", // Name of the Business Object
    context : $("#ordersListPage")[0], // Where the data has to be bound
    to
    read: {
        // Settings for the read method
        namespace: "http://schemas.cordys.com/NW",
        dataType: "json",
        method: "GetOrdersObjects"
    },
    // Settings for the update/synchronize method
    update: {
        method: "UpdateOrders",
    }
})
```

```

        namespace: "http://schemas.cordys.com/NW",
        dataType: "json"
    }
});

orderDetailsModel = new $.cordys.model({
    objectName: "Order_Details", // Name of the Business Object
    context : $("#orderDetailsList")[0], // Where the data has to be
bound to
    read: {
        // Settings for the read method
        namespace: "http://schemas.cordys.com/NW",
        dataType: "json",
        method: "GetOrder_DetailsObjects"
    }
});

```

To add the data-bind attribute:

Add the data-bind attribute with the corresponding prefix to format the field according to the user preferences as shown below. For date fields, you must provide the date pattern in the data-datepicker attribute.

- 1. Add the decimal format:** Provide the decimal format for the Discount field as follows.

```
<input id="fldDiscount_List" data-bind="decimal:Discount" type="text" />
```

- 2. Add the currency format:** Provide the currency format for the Freight and Unit Price fields as follows.

```
<input data-bind="currency:Freight" id="fldFreight" type="text"/>
```

- 3. Add the date format:** Provide the date format with the pattern for the Shipped Date field as follows.

```
<input data-bind="date:ShippedDate" data-datepicker="MM/DD/YYYY"
id="fldShippedDate"/>
```

- 4. Add the date-time format:** Provide the date-time format with the pattern for the Order Date field as follows.

```
<input data-bind="datetime:OrderDate" data-datepicker="MM/DD/YYYY HH:mm:ss"
id="fldOrderDate"/>
```

- 5. Add the date-time format:** Provide the date-time format with the pattern for the Required Date field.

```
<input data-bind="datetime:RequiredDate" data-datepicker="YYYY/MM/DD HH:mm:ss"
id="fldRequiredDate"/>
```

- 6. Add the number format:** Provide the number format for the Quantity field.

```
<input data-bind="number:Quantity" id="fldQuantity" type="text"/>
```

To read the orders:

- Display the orders list by reading the Orders from the Northwind table.

```
ordersModel.read({ "parameters" : { cursor: { "@numRows" : 9, "@position" : 0 },
fromOrderID : 0, toOrderID :99999 } })
```

To work with the orders page:

- To view the order details, click any Order to view the order details.
- To edit the order details, click the required Order, make the necessary changes, and click **Save**.
- To view order items, click **Order Items** to view the corresponding orders item of the selected order.

Example - Orders with Format

```
<!DOCTYPE html> <html> <head> <title>Orders with Format</title> <meta
name="viewport" content="width=device-width, initial-scale=1"/> <link
rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css" type="text/css" />
<link rel="stylesheet"
href="/cordys/thirdparty/jquery/jquery.mobile.structure.min.css" type="text/css" />
</head> <body> <div data-role="page" id="ordersListPage"> <div data-role="header"
data-theme="b"> <h1>Orders</h1> </div> <div data-role="content" data-theme="b"> <ul
id="orderList" data-role="listview" data-theme="c" data-inset="true" data-
bind="foreach:{ data:Orders }"> <li> <a href="#orderdetailsPage" data-
transition="pop" class="ui-link-inherit" data-bind="click:selectOrder($data)"> <h3
class="ui-li-heading"><span data-bind="text:OrderID"></h3> <p class="ui-li-desc"
data-bind="text:ShipName"></p> <p class="ui-li-desc" data-
bind="datetime:OrderDate"></p> <p class="ui-li-desc"><span data-
bind="text:ShipAddress"> </a> </li> </ul> </div> <div data-role="page"
id="orderdetailsPage"> <div data-role="header" data-theme="b"> <a
href="#ordersListPage" data-role="button" data-icon="back" data-rel="back"
onclick="revertOrderChanges()">Back</a> <h1>Order Details</h1> <a data-role="button"
id="btnEdit">Save</a> </div> <div data-role="content" data-theme="c"> <ul data-
role="listview" id="detailView"> <li data-role="fieldcontain"> <div id="orderDiv">
<div> <label for="fldOrderID" class="ui-input-text">Order ID</label> <input
type="text" id="fldOrderID" data-bind="value:OrderID" class="ui-input-text ui-body-b
ui-corner-all ui-shadow-inset"/> <a href="#orderDetailsList" data-inline='true'
data-icon="star" data-bind="click:lauchOrderDetails" data-role="button">Order
Items</a> </div> <div> <label for="fldCustomerID" class="ui-input-
text">Customer</label> <input type="text" id="fldCustomerID" data-
bind="value:CustomerID" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div> <label for="fldOrderDate" class="ui-input-
text">OrderDate</label> <input type="text" id="fldOrderDate" data-
bind="datetime:OrderDate" data-datepattern = "MM/DD/YYYY HH:mm:ss" placeholder =

```

```

"MM/DD/YYYY HH:mm:ss" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div > <label for="fldShippedDate" class="ui-input-text"> Shipped
Date</label> <input type="text" id="fldShippedDate" data-bind="date:ShippedDate"
data-datepattern = "MM/DD/YYYY" placeholder = "MM/DD/YYYY" class="ui-input-text ui-
body-b ui-corner-all ui-shadow-inset"/> </div> <div > <label for="fldRequireDate"
class="ui-input-text"> Require Date</label> <input type="text" id="fldRequireDate"
data-bind="datetime:RequiredDate" data-datepattern = "YYYY/MM/DD HH:mm:ss"
placeholder = "YYYY/MM/DD HH:mm:ss" class="ui-input-text ui-body-b ui-corner-all ui-
shadow-inset"/> </div> <div > <label for="fldShipName" class="ui-input-
text">ShipName</label> <input type="text" id="fldShipName" data-
bind="value:ShipName" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <label for="fldShipAddress" class="ui-input-
text">ShipAddress</label> <input type="text" id="fldShipAddress" data-
bind="value:ShipAddress" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <label for="fldFreight" class="ui-input-text">Freight</label>
<input type="text" id="fldFreight" data-bind="currency:Freight" class="ui-input-text
ui-body-b ui-corner-all ui-shadow-inset"/> </div> </div> </ul> </div> </div>
<div data-role="page" id="orderDetailsList"> <div data-role="header" data-theme="b">
<a href="#orderdetailsPage" data-role="button" data-icon="back" data-rel="back"
onclick="revertOrderDetailsChanges()">Back</a> <h1>Order Items</h1> </div> <div
data-role="content" data-theme="c"> <ul data-role="listview"
id="orderDetailsListview" data-theme="c" data-inset="true" data-bind="foreach:Order_
Details"> <li data-role="fieldcontain"> <div > <label for="fldProductID_List"
class="ui-input-text">Product ID</label> <input type="text" id="fldProductID_List"
data-bind="value:ProductID" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div > <label for="fldUnitPrice_List" class="ui-input-
text">Unit
Price</label> <input type="text" id="fldUnitPrice_List" data-
bind="currency:UnitPrice" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div > <label for="fldQuantity_List" class="ui-input-
text">Quantity</label> <input type="text" id="fldQuantity_List" data-
bind="number:Quantity" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div> <div > <label for="fldDiscount_List" class="ui-input-
text">Discount</label> <input type="text" id="fldDiscount_List" data-
bind="decimal:Discount" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/> </div><br/> </li> </ul> </div> </div> <script
src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"></script> <script
src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
type="text/javascript"></script> <script
src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"></script>
<script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>
<script src="/cordys/html5/cordys.html5sdk.util.js" type="text/javascript"></script>
<script type="text/javascript"> var empModel, ordersModel, // Reference to the model
selectedOrder = ko.observable(), orderDetailsModel; $(function() { ordersModel = new
$.cordys.model({ objectName: "Orders", // Name of the Business Object context :
$("#ordersListPage")[0], // Where the data has to be bound to read: { // Settings
for the read method namespace: "http://schemas.cordys.com/NW", dataType: "json",
method: "GetOrdersObjects" }, // Settings for the update/synchronize method update:
{ method: "UpdateOrders", namespace: "http://schemas.cordys.com/NW", dataType:
"json" } }); orderDetailsModel = new $.cordys.model({ objectName: "Order_Details",
// Name of the Business Object context : $("#orderDetailsList")[0], // Where the
data has to be bound to read: { // Settings for the read method namespace:
"http://schemas.cordys.com/NW", dataType: "json", method: "GetOrder_DetailsObjects"
});

```

Implementing pagination

You can implement the pagination over Web services using the AppWorks Platform cursor mechanism and the `cordys.model` plug-in available in the AppWorks Platform HTML5 SDK. See [Getting Started with the SDK](#) for more information.

Pagination can be done in the mobile interfaces in one of the following ways:

- Example 1: Pagination with buttons
- Example 2: Pagination with endless scrolling list

Using an endless list is preferred to pagination buttons, because it is the most common approach in the mobile apps.

Example 1

This example is built on the Employees table from the Northwind Database. The app displays a set of three employees per page. The next and previous buttons are displayed, which invoke the `model.getNextPage()` and `model.getPreviousPage()` methods respectively. You can subscribe to the collection of the business objects using the `model.read()` API. You can invoke the methods `model.hasNext()` and `model.hasPrevious()` to disable or enable   whenever the collection of business objects is modified.

Pagination with buttons

```
<html>
    <head>
        <title>Employees</title>
        <meta content="width=device-width, initial-scale=1" name="viewport"/>
        <link href="/cordys/thirdparty/jquery/jquery.mobile.min.css" rel="stylesheet"/>
        <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"/>
        <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"/>
        <script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"/>
        <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
        <script type="text/javascript"> $(function() { // Create a new model
            empModel = new $.cordys.model({ objectName: "Employees", // Name of the Business
                Object context : document.body, // Where the data is bound to read: { // Settings
                    for the read method namespace: "http://schemas.cordys.com/NW", dataType: "json",
                    method: "GetEmployeesObjects", // Parameters for the method parameters: {
                        fromEmployeeID: "0", toEmployeeID: "99", cursor:{ '@numRows':4 } } } });
                // Call the read method. This would fire the method with the namespace and parameters as
                specified in the read settings above. empModel.read(); // Subscribe to any change in
```

```

the collection of Employees to enable/disable the next, previous buttons
empModel.Employees.subscribe(function(){ // disable the previous button if there are
no previous records in the cursor $("#btnPrevious").toggleClass("ui-disabled", !
empModel.hasPrevious()); // disable the next button if we are at the end of the
cursor $("#btnNext").toggleClass("ui-disabled", ! empModel.hasNext()); });
}); // Called on clicking refresh button. Refreshes the page function refresh() {
empModel.clear(); empModel.read(); } // Called on clicking the Next button. Gets the
next set of records function getNext() { empModel.getNextPage(); } // Called on
clicking the previous button. Gets the previous set of records function getPrevious
() { empModel.getPreviousPage(); } </script>
</head>
<body>
<div data-role="page" id="mainPage">
    <div data-role="header" data-theme="b">
        <a data-bind="click:refresh" data-icon="refresh">Refresh</a>
        <h1>Employees</h1>
        <div class="ui-btn-right" data-role="controlgroup" data-
type="horizontal">
            <a class="ui-disabled" data-bind="click:getPrevious"
                data-icon="arrow-l" data-iconpos="notext"
                data-role="button" id="btnPrevious">Previous</a>
            <a class="ui-disabled" data-bind="click:getNext"
                data-icon="arrow-r" data-iconpos="notext"
                data-role="button" id="btnNext">Next</a>
        </div>
    </div>
    <div data-role="content" data-theme="b">
        <ul data-bind="foreach:Employees" data-inset="true"
            data-role="listview" data-theme="c" id="employeeList">
            <li>
                <!-- ko if: typeof(Photo) !=="undefined" && Photo-->
                <img class="ui-li-thumb ui-corner-tl" data-bind="attr: {src:
'data:image/bmp;base64,' + Photo.substring(104)}"/>
                <!-- /ko -->
                <!-- ko if: typeof(Photo) ==="undefined" || (!Photo)-->
                
                <!-- /ko -->
                <h3 class="ui-li-heading">
                    <span data-bind="text:FirstName"/>
                    <span data-bind="text:LastName"/>
                </h3>
                <p class="ui-li-desc" data-bind="text:Address"/>
                <p class="ui-li-desc">
                    <span data-bind="text:City"/>
                    <span data-bind="text:Country"/>
                </p>
            </li>
        </ul>
    </div>
</div>
</body>
</html>

```

Example 1 is similar to `employeeslistwithnavigation.htm` available in the demo folder in the AppWorks Platform HTML5 SDK. For more information on the other demo pages in the AppWorks Platform HTML5 SDK, see [Sample Pages](#).

Example 2

This is built on the Orders table from the Northwind Database. In an endless scrolling list, the next set of records is retrieved whenever the scroll appears below the rendered page and the result is appended to the existing list.

In this Web page, the next set of results are displayed by invoking `model.getNextPage()` on the scroll or touchmove event of the list content. You can subscribe to the collection of the business objects using `model.read()`.

Pagination with endless scrolling list

```
<!DOCTYPE html>
<html>
    <head>
        <title>Orders</title>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>
        <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
        <link rel="stylesheet" href="/cordys/thirdparty/jquery/mobile.structure.min.css" type="text/css" />
        <script src="/cordys/thirdparty/jquery/jquery.js"></script>
        <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"></script>
        <script src="/cordys/thirdparty/jsrender/jsrender.js"
type="text/javascript"></script>
        <script src="/cordys/html5/cordys.html5sdk.js"
type="text/javascript"></script>
        <style type="text/css">
        </style>
    </head>
    <script id="ordersTemplate" type="text/x-jsrender">
        <li data-icon="false">
            <a href="#" data-transition="pop" class="ui-link-inherit">
                <div >
                    <h3 class="ui-li-heading">Order {{:OrderID}}</h3>
                    <p class="ui-li-desc">Customer :{{:CustomerID}}</p>
                    <p class="ui-li-desc">Country :{{:ShipCountry}}</p>
                </div>
            </a>
        </li>
    </script>
    <script type="text/javascript">
        var orders = [],readNextOrders = false;
        var orderModel;
```

```

$(function() {
    // Create a new model
    orderModel = new $.cordys.model({
        objectName: "Orders", // Name of the Business Object
        context : document.body, // Where the data is bound to
        read: {
            // Settings for the read method
            namespace: "http://schemas.cordys.com/NW",
            dataType: "json",
            method: "GetOrdersObjects",
            // Parameters for the method
            parameters: {
                cursor:{
                    '@numRows':9
                },
                fromOrderID : "10248",
                toOrderID : "10287"
            }
        }
    });

    // Call the read method. Method is fired with namespace and parameters as
    // specified above.
    orderModel.clear();
    orderModel.read().done(function(){
        readNextOrders = true; // only after read method getnextpage should fire
        orders = orders.concat(orderModel.Orders);
        var html = $("#ordersTemplate").render(orders);
        $('#orderList')
            .html(html)
            .listview("refresh");
    });
});

//Specific to IE
if (navigator.appName.indexOf("Microsoft Internet") != -1)
    $(window).scroll(function(){ fireScroll()});
//getTouchEventName()
$(document).on(getTouchEventName(),function(){ fireScroll()});

function fireScroll() {
    var activePage = $.mobile.activePage ;
    if (activePage && activePage.attr("id") == "ordersPage") {
        // check whether scroll is near the bottom of the page
        if ($(window).scrollTop() + $(window).height() > (activePage.height() -
150)) {
            // readNextOrders is false initially to avoid the firing of
            // getNextPage before read method
            if(readNextOrders && orderModel.hasNext()){
                readNextOrders = false;
                orderModel.getNextPage().done( function(){

```

Example 2 is similar to `orderslistwithswipe.htm` available in the demo folder of the AppWorks Platform HTML5 SDK. For more information on the other demo pages in the AppWorks Platform HTML5 SDK, see [Sample Pages](#).

Integrating with Google Maps using HTML5 SDK

You can integrate your application with Google Maps using the Cordys HTML5 SDK. Refer to Getting Started with the SDK for information on getting started with the SDK.

This example displays the location of an employee on a mobile device using Google Maps. It lists all the employees in the Northwind Database and upon selecting a particular employee, a details page is displayed with the address and location of the employee plotted on Google Map.

A Geocoder API is used here to convert an address into a `LatLng` object, which is used to create a map. Although only the Address is used as a parameter, you can also pass other parameters to specify a better search. Refer to Google API documentation for more information. The search done in the previous step results in an array of objects or locations that are close to the address mentioned. To simplify, only the first result is considered in this example and is displayed on the Map. However, you may use more than one parameter. In addition to this, you must explicitly use the Marker API to add the mark pointing to the place selected.

See [Google Maps Javascript API V3 Reference](#) for more information on these APIs.

The example provided here is similar to `employeeswithlocation.htm` that is available under the demo folder in the AppWorks Platform HTML5 SDK. See [Sample Pages](#), for more information on other Demo pages shipped with the SDK.

employeeswithlocation.htm

```
<!DOCTYPE html> <html> <head> <title>Employees</title> <meta name="viewport" content="width=device-width, initial-scale=1"/> <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css" type="text/css" /> <link rel="stylesheet" href="/cordys/thirdparty/jquery/jquery.mobile.structure.min.css" type="text/css" /> <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"></script> <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"></script> <script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"></script> <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script> <script type="text/javascript" src="https://maps.google.com/maps/api/js?sensor=false"></script> <script
```

```

type="text/javascript"> var empModel; // Reference to the model $(function() {
ko.bindingHandlers.photo = { update: function(element, valueAccessor) { var value =
valueAccessor(); if (value) { element.src = 'data:image/bmp;base64,' +
value.substring(104); } else {
element.src="/cordys/html5/demo/images/defaultphoto.jpg"; } } };
$("#detailsPage").bind("pageshow",function(){ refreshPage(); //shows the employee's
location on the google map initializeMap(); }) // Create a new model empModel = new
$.cordys.model({ objectName: "Employees", // Name of the Business Object context :
document.body, // Where the data has to be bound to read: { // Settings for the read
method namespace: "http://schemas.cordys.com/NW", dataType: "json", method:
"GetEmployeesObjects", // Parameters for the method parameters: { fromEmployeeID:
"0", toEmployeeID: "99" } } }); // Call the read method. This would fire the method
with the namespace and parameters as specified in the read settings above.
empModel.read();}); // Called on clicking an Employee function selectEmployee(emp)
{ return function(data) { // Let us set the currently clicked item as the selected
item to show in the detail page empModel.selectedItem(data); return true; } }
function refreshPage() { var employeeListView = $('#detailView');
employeeListView.trigger("create"); employeeListView.listview("refresh"); }
</script> </head> <body> <div data-role="page" id="mainPage"> <div data-
role="header" data-theme="b"> <h1>Employees</h1> </div> <div data-role="content"
data-theme="b"> <ul id="employeeList" data-role="listview" data-theme="c" data-
inset="true" data-bind="foreach:Employees"> <li> <a href="#detailsPage" data-
transition="pop" class="ui-link-inherit" data-bind="click:selectEmployee($data)">
<img class="ui-li-thumb ui-corner-tl" data-bind="photo:Photo"/> <h3 class="ui-li-
heading"><span data-bind="text:FirstName"></span>&ampnbsp<span data-
bind="text:LastName"></span></h3> <p class="ui-li-desc" data-
bind="text:Address"></p> <p class="ui-li-desc"><span data-
bind="text:City"></span>&ampnbsp<span data-bind="text:Country"></span></p> </a> </li>
</ul> </div> <div data-role="page" id="detailsPage"> <div data-role="header"
data-theme="b"> <a href="#mainPage" data-role="button" data-icon="back" data-
rel="back">Back</a> <h1>Employee Location</h1> </div> <div data-role="content" data-
theme="c"> <ul data-role="listview" id="detailView" data-bind="with: selectedItem">
<li> <a class="ui-link-inherit"> <img class="ui-li-thumb ui-corner-tl" data-
bind="photo:Photo"/> <h2 class="ui-li-heading" data-bind="text:EmployeeID"></h2> <h3
class="ui-li-heading"><span data-bind="text:TitleOfCourtesy"></span>&ampnbsp<span
data-bind="text:FirstName"></span>&ampnbsp<span data-bind="text:LastName"></span></h3>
<p class="ui-li-desc"><span data-bind="text:Title"></p> </a> </div> </li> <li data-
role="fieldcontain"> <div id="employeeDiv"> <div> <label for="fldAddress"
class="ui-input-text">Address</label> <input readonly type="text" id="Address" data-
bind="value:Address + ', ' + City + ', ' + Country" class="ui-input-text ui-body-b
ui-corner-all ui-shadow-inset"/> </div> </div> <li data-role="fieldcontain">
<div> <div> <label for="map_canvas" class="ui-input-text">Location</label> </div>
<div> <div id="map_canvas" style="width:100%; height:500px"></div> </div> </div>
</li> </ul> </div> <script> function initializeMap() { var address =
$('#Address').val(); // a new google maps object is created to handle the request
and we pass the address to it var geoCoder = new google.maps.Geocoder(address) // a
new object for the request I called "request" , you can put there other parameters
to specify a better search (check google api doc for details) , // on this example
im going to add just the address var request = {address:address}; // I make the
request geoCoder.geocode(request, function(result, status){ // as a result i get two
parameters , result and status. // results is an array tha contenis objects with the
results founds for the search made it. // to simplify the example i take only the
first result "result[0]" but you can use more than one if you want // So , using the
}

```

Integrating with WS-AppServer

You can integrate your application with WS-AppServer using the `cordys.model.wsapps` plug-in available in the AppWorks Platform HTML5 SDK. This plug-in is extended from the `cordys.model` plug-in and it uses the KnockoutJS library to handle the transactions. For more information on getting started, see [Getting Started with the SDK](#).

This example displays Orders from the Northwind Database with the WS-Apps server integration.

The following steps are involved in this example:

1. Defining the business logic
2. Defining the WS-Apps models
3. Retrieving the Orders
4. Creating an Order
5. Managing Orders

To define the business logic:

1. Define the access modes by doing the following:
 - a. Hide the OrderID for the new orders.
 - b. Disable the OrderId for the existing orders.
 - c. Hide the Freight field if its value is less than ten.
 - d. Display the Freight for the new orders.
2. Enable the auto-initialization of new data by doing the following:
 - a. For Order Date, specify the present date in the case of a new order.
 - b. For Required Date, specify the date one week later than the order date of a new order.
3. Specify the validation constraints by doing the following:
 - a. Customer ID must be present in the customer table.
 - b. Order Date cannot be a past date.
4. Enable automatic fill of the selected control.
EmployeeID comprises the first and last name of the employee.

For more details on defining the business logic, see [Using WS-AppServer business logic in XForms](#).

To define WS-Apps models:

- Define orders WS-Apps model for retrieving, creating, and updating the orders as displayed in the following code block.

Defining WS-Apps model

```

orderModel = new $.cordys.model.wsapps({
    context: document.body,
    objectName: "Orders", // Name of the Business Object
    // Common settings for all methods - read, create, update, delete,
    synchronize
    defaults: {
        namespace: "http://schemas.cordys.com/Northwind",
        dataType: "json"
    },
    fields: [{
        name: "EmployeeID",
        getPossibleValues: true

    }, {
        name: "CustomerID"
    },
    {name: "OrderID" },
    {name: "RequiredDate" },
    {name: "OrderDate" },
    {name: "ShippedDate" },
    {name: "ShipVia" },
    {name: "Freight" },
    {name: "ShipName" },
    {name: "ShipAddress" },
    {name: "ShipCity" },
    {name: "ShipRegion" },
    {name: "ShipPostalCode" },
    {name: "ShipCountry" }],
    // Settings for the update/synchronize method
    update: {
        method: "UpdateOrders"
    },
    // Settings for the read method
    read: {
        method: "GetOrdersObjects",
        parameters: {
            "fromOrderID": "10248",
            "toOrderID": "10287"
        }
    },
    wsapps: {
        checkAccessControl: "1",
        initializeDefaults: "1"
    }
});

```

In the Defining WS-Apps model code block, all the fields are added inside the fields object. For EmployeeID field, getPossibleValues is set to true for reading all the possible options from the Employees table of Northwind database.

In the wsapps object, checkAccessControl and initializeDefaults are set to 1 for enabling the access control and for getting the default values for new orders respectively.

To retrieve orders:

1. Retrieve orders from the Orders table of Northwind database using the `orderModel.read()` method.
2. Display the orders using the following code blocks.

Displaying the OrderID

```
<div data-bind="visible: (typeof OrderID['@access'] === 'function') ? OrderID['@access']() != 'h' : OrderID['@access'] != 'h'">

    <label for="fldOrderID" class="ui-input-text">Order ID</label>

    <input type="text" id="fldOrderID" data-bind="value: (typeof OrderID == 'object' && OrderID != null) ? OrderID.text() : OrderID, enable: (typeof OrderID['@access'] === 'function') ? OrderID['@access']() != 'r' : OrderID['@access'] != 'r'" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/>

</div>
```

Displaying the Freight

```
<div data-bind="if: (typeof Freight == 'object' && Freight != null) ? Freight['@access']() != 'h' : true">

    <label for="fldFreight" class="ui-input-text">Freight</label>

    <input type="text" id="fldFreight" data-bind="value: (typeof Freight == 'object' && Freight != null) ? Freight.text : Freight" class="ui-input-text ui-body-b ui-corner-all ui-shadow-inset"/>

</div>
```

To create an order:

- To create a new order, click **Add**.
The following occurs:
 - OrderID is hidden.
 - For the Order Date, present date is specified.
 - For the Required Date, the date one week later than the order date is specified.
 - For EmployeeID, list of employees from the employees table is displayed.
 - A validation to ensure that the CustomerID exists in the customers table.

To manage orders:

1. To view the order details, click the order.
2. To edit the order, click **Edit** in the Order Details page, apply the changes, and click **Save**.
3. Refer to the following code block for more information on this example.

Orders with WS-Apps

```
<!DOCTYPE html>
<html>
  <head>
    <title>Orders</title>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
    <link rel="stylesheet"
href="/cordys/thirdparty/jquery/jquery.mobile.structure.min.css" type="text/css" />

    <script src="/cordys/thirdparty/jquery/jquery.js"
type="text/javascript"></script>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
type="text/javascript"></script>

    <script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"></script>
    <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>
    <script src="/cordys/html5/plugins/cordys.model.wsapps.js"
type="text/javascript"></script>

    <script type="text/javascript">
        var orderModel; // Reference to the model
        var isEditMode; // True if Details Page is in view mode (Details page can be in
view or edit mode).
    </script>
```

```
// Create a new model on page ready
$(function() {
    orderModel = new $.cordys.model.wsapps({
        context: document.body,
        objectName: "Orders", // Name of the Business Object
        // Common settings for all methods - read, create, update, delete,
        synchronize
        defaults: {
            namespace: "http://schemas.cordys.com/Northwind",
            dataType: "json"
        },
        fields: [
            {name: "EmployeeID",
             getPossibleValuesPerRecord: true,
             getPossibleValues: true
            },
            {name: "CustomerID"
            },
            {name: "OrderID" },
            {name: "RequiredDate" },
            {name: "OrderDate" },
            {name: "ShippedDate" },
            {name: "ShipVia" },
            {name: "Freight" },
            {name: "ShipName" },
            {name: "ShipAddress" },
            {name: "ShipCity" },
            {name: "ShipRegion" },
            {name: "ShipPostalCode" },
            {name: "ShipCountry" }],
        // Settings for the update/synchronize method
        update: {
            method: "UpdateOrders"
        },
        // Settings for the read method
        read: {
```

```
;  
    return true;  
}  
  
// Custom binding handler for loading the employee options and showing the  
selected employee  
ko.bindingHandlers.employeesList = {  
    update: function(element, valueAccessor, allBindingsAccessor) {  
        $(element).html(empSelectOptions);  
    }  
};  
  
// Called on clicking an Order  
function selectEmployee(order) {  
    return function(data) {  
        orderModel.selectedItem(data);  
        // Set the Details Page to View Mode  
        isViewMode = true;  
        return true;  
    }  
}  
  
function getOptions(possiblevalues){  
    if(possiblevalues instanceof Array)  
        return ko.observableArray(possiblevalues[0].values);  
    else  
        return ko.observableArray(possiblevalues.values);  
}  
  
function validateCustomer(){  
    orderModel.validate(orderModel.selectedItem()).done(function(response) {  
        var order = response.new.Orders,  
            errorType = order.CustomerID['@crType'],  
            errorMsg = order.CustomerID['@crMsg'] || undefined;  
  
        switch(errorType) {  
            case 'E' : alert("Error: " + errorMsg);  
                        break;  
            case 'N' : alert("Notification: " + errorMsg);  
                        break;  
            case 'I' : alert("Info: " + errorMsg);  
        }  
    })  
}
```

```

        <label for="fldRequiredDate" class="ui-input-text">Required
Date</label>
        <input type="text" id="fldRequiredDate" data-
bind="value:RequiredDate" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/>
        </div>
        <div >
            <label for="fldShippedDate" class="ui-input-
text">Shipped Date</label>
            <input type="text" id="fldShippedDate" data-
bind="value:ShippedDate" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/>
            </div>
            <div data-bind="if: (typeof Freight == 'object' && Freight
!= null) ? Freight['@access']() != 'h' : true">
                <label for="fldFreight" class="ui-input-
text">Freight</label>
                <input type="text" id="fldFreight" data-bind="value:
(typeof Freight == 'object' && Freight != null) ? Freight.text : Freight" class="ui-
input-text ui-body-b ui-corner-all ui-shadow-inset"/>
            </div>
            </div>
        </li>
    </ul>
</div>
<script type="text/javascript">
    // Add Click handler to the Edit Button on Page Ready
    $(function(){
        $("#btnEdit").bind("click", function () {
            if (isEditMode) {
                // We are switching to Edit Mode. Let us change the button's text to
Done and enable all the fields
                $(".ui-btn-text", this).text("Done");
                $("#detailView :input").removeAttr('disabled');
                $("select", "#detailView").removeAttr('disabled');
            }
            else {
                // Set the selected Employee ID in the Update request
                var selectedItem = orderModel.selectedItem(),
                    empType = typeof selectedItem.EmployeeID;
                if(empType !== 'function') {
                    (empType !== 'string') ? selectedItem.EmployeeID.text
($("#fldEmployeeID").val()) : selectedItem.EmployeeID = $("#fldEmployeeID").val();
                }
                // Set the qAccess attribute in the Update request
                orderModel.synchronize().done(function(response) {
                    alert("Order details have been saved.");
                    // Set the newly created Order Id
                    $('#fldOrderID').val(response.tuple.new.Orders.OrderID.text);
                    // We are switching to View Mode. Let us change the button's
                })
            }
        })
    })

```

Retrieving the data

You can retrieve the data using `cordys.ajax` plug-in in the AppWorks Platform HTML5 SDK. For information on getting started, see [Getting Started with the SDK](#).

The process of retrieving the data has been explained using an example in which all the employee details are retrieved from the Northwind Database and displayed as a list. In the code snippet Employees List below, the `cordys.ajax` plug-in is used to retrieve the information based on the method, namespace, and parameters. Here, the `GetEmployeesObjects` method retrieves all employee objects if the URL `http://schemas.cordys.com/NW/GetEmployeesObjects` is valid for a Web service in AppWorks Platform. Authentication is handled internally through the `cordys.authentication` plug-in.

The function specified in the success callback handler is invoked with the retrieved employee objects as data parameter. As the `dataType` is set to `json`, the XML that is returned from the server is automatically converted into javascript objects. The `$.map` function is used to create a new array with only the Employees objects, removing the tuple-old structure for an efficient rendering. The objects are passed to the rendering engine and the result HTML from rendering engine is added to the Document Object Model (DOM).

This example uses the standard libraries jQuery and jQuery Mobile, in order to make it work on all the Mobile devices. While using the jQuery library is mandatory, jQuery Mobile can be replaced by any other UI library. You can use the standard libraries from the location `/cordys/thirdparty/jquery`.

While using the jQuery library, use `$.ajax` to retrieve the data. The `$.cordys.ajax` is an extension of `$.ajax`, and is used for invoking the AppWorks Platform Web services. You can also include a rendering library to render the data, if required. In this example, jsRender is used for rendering the data, but you can also use other rendering libraries of your choice, such as KnockoutJS. For information on using KnockoutJS as the binding library, see [Building read-only mobile applications](#).

Example

Employees list

```
<html>
  <head>
    <title>Employees</title>
    <meta content="width=device-width, initial-scale=1" name="viewport"/>
    <link href="/cordys/html5/jquery/jquery.mobile.min.css" rel="stylesheet"/>
    <script src="/cordys/html5/jquery/jquery.js" type="text/javascript"/>
    <script src="/cordys/html5/jquery/jquery.mobile.js" type="text/javascript"/>
```

```

<script src="/cordys/html5/jquery/jsrender.js" type="text/javascript"/>
<script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
<style type="text/css"> img.photo { position: absolute; top:0px; right:0px;
bottom:0px; height:100%; } </style>
<script id="employeeTemplate" type="text/x-jsrender">
<li>
<div>
<h3 class="ui-li-heading">&{{:FirstName}} {{:LastName}}</h3>
<p class="ui-li-desc">&{{:Address}}</p>
<p class="ui-li-desc">&{{:City}} {{:Country}}</p>
</div>
</li>
</script>
<script type="text/javascript"> $(function() { // get the employees
$.cordys.ajax({ method: "GetEmployeesObjects", namespace:
"http://schemas.cordys.com/NW", parameters: { fromEmployeeID : "0", toEmployeeID :
"99" }, dataType: 'json', // the xml result will be converted into js objects
success: function(data) { // Create an array with only the Employees objects, skip
the tuple/old structure var employees = $.map($.makeArray(data.tuple), function
(tuple, index) { if (tuple.old.Employees.Photo) { // Remove first 104 characters
from the Photo, which is header information tuple.old.Employees.Photo =
tuple.old.Employees.Photo.substring(104); } return tuple.old.Employees; }); var html
= $("#employeeTemplate").render(employees); $('#employeeList') .html(html) .listview
("refresh"); } }); });
</script>
</head>
<body>
<div data-role="page" id="">
<div data-role="header" data-theme="b">
<h1>Employees</h1>
</div>
<div data-role="content" data-theme="b">

<ul data-inset="true" data-role="listview"
data-theme="c" id="employeeList"/>
</div>
</div>
</body>
</html>

```

Translating a page into a specific language

You can implement the translation of a Web page using `cordys.translation` plug-in available in the AppWorks Platform HTML5 SDK. See [Getting Started with the SDK](#) for information on getting started.

In this section, you will learn how to translate the Employees Web page using the `cordys.translation` plug-in available in the AppWorks Platform HTML5 SDK.

This example is built on the Employees table from the Northwind Database. This displays the employees list and the details based on the user preferences.

The following steps are involved in translating the Web page:

1. Creating a Java Script message bundle
2. Retrieving the message bundle
3. Translating the elements

To create a Java script message bundle:

A JavaScript message bundle is a collection of messages, which are available for referencing from any JavaScript file.

- See [Working with Javascript Message Bundles](#) for creating a JavaScript message bundle.

In the example, we have created a message bundle `employeewithtranslation` inside the `html5sdk` folder for translating the Employees Web page to Dutch language.

To retrieve the message bundle:

- Retrieve the `employeewithtranslation` JavaScript message bundle by specifying the `bundlePath` in the `$.cordys.translation.getBundle()` API.
This API considers the language code as an optional second parameter.
If no language is provided, the current language of the user is used by default.

Retrieve Message Bundle

```
$.cordys.translation.getBundle("html5sdk/employeewithtranslation").done(function
(mBundle) {
    // You can do the translation here
});
```

Translating the elements

To add the data-translatable attribute:

- To translate a label element using the `translate` method, add attribute `data-translatable = "true"` for all the elements that need to be translated as given in the following code block.

`data-translatable="true"`

```
<h1 data-translatable="true">Employees</h1>
```

To translate the elements:

- When the message bundles are retrieved, use the `translate()` method to translate all the elements that have the attribute `data-translatable = "true"`.

Translating the elements

```
$.cordys.translation.getBundle("html5sdk/employeewithtranslation").done(function
(mBundle) {
    mBundle.translate();
});
```

To translate messages:

- You can translate the messages of your Web page using the `getMessage()` method similar to the following code block.
The Employee Web page does not have any messages for translation. Hence, this is not required in this example.

Translate messages

```
$.cordys.translation.getBundle("html5sdk/employeewithtranslation").done(function
(mBundle) {
    var translatedMessage = mBundle.getMessage("The capital of {0} is {1}", "UK",
"London");
});
```

See Employees with Translation for more information on the code block.

Employees with Translation

```
<!DOCTYPE html>
<html>
  <head>
    <title>Employees</title>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
    <link rel="stylesheet"
href="/cordys/thirdparty/jquery/mobile.structure.min.css" type="text/css" />
    <script src="/cordys/thirdparty/jquery/jquery.js"
type="text/javascript"></script>
    <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
type="text/javascript"></script>

    <script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"></script>
    <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>

    <script type="text/javascript">
var mBundle,
    empModel; // Reference to the model
$(function() {
```

```
// Get the message bundle on ready event
$.cordys.translation.getBundle("html5sdk/employeewithtranslation").done
(function(messageBundle) {
    mBundle = messageBundle;
    // Translates all the labels which has data-translatable="true"
attribute
    mBundle.translate();
});

$("#detailsPage").bind("pageshow",function(){
    // Create and refresh the page again to avoid style issues
    $('#detailView').trigger("create").listview("refresh");

    // Translates all the labels which has data-translatable="true"
attribute on details page
    mBundle.translate();

    // JQuery Mobile will add two span for anchor tag so the translation
will affect the styling
    // Hence translating the button span text directly
    $("a#backbutton span span.ui-btn-text").text(function() {
        return mBundle.getMessage($(this).text());
    });
});

// Create a new model
empModel = new $.cordys.model({
    objectName: "Employees", // Name of the Business Object
    context : document.body, // Where the data has to be bound to
    read: {
        // Settings for the read method
        namespace: "http://schemas.cordys.com/NW",
        dataType: "json",
        method: "GetEmployeesObjects",
        // Parameters for the method
        parameters: {
            fromEmployeeID: "0",
            toEmployeeID: "99"
        }
    }
});

// Call the read method. This would fire the method with the namespace and
parameters as specified in the read settings above.
empModel.read();
```

```

<h3 class="ui-li-heading"><span data-bind="text:FirstName">&nbsp;</span>
    <span data-bind="text:LastName"></h3>
        <p class="ui-li-desc" data-bind="text:Address"></p>
            <p class="ui-li-desc"><span data-
bind="text:City">&nbsp;<span data-bind="text:Country"></p>
                </a>
            </li>
        </ul>
    </div>
</div>
<div data-role="page" id="detailsPage">
    <div data-role="header" data-theme="b">
        <a href="#mainPage" id="backbutton" data-role="button" data-icon="back"
data-rel="back">Back</a>
        <h1 data-translatable="true">Employee Details</h1>
    </div>
    <div data-role="content" data-theme="c">
        <ul data-role="listview" id="detailView" data-bind="with: selectedItem">
            <li>
                <div>
                    <a class="ui-link-inherit">
                        <h2 class="ui-li-heading" data-
bind="text:EmployeeID"></h2>
                        <h3 class="ui-li-heading"><span data-
bind="text:FirstName">&nbsp;<span data-bind="text:LastName"></h3>
                            <p class="ui-li-desc"><span data-bind="text:Title"></p>
                        </a>
                    </div>
                </li>
                <li data-role="fieldcontain">
                    <div id="employeeDiv">
                        <div>
                            <label for="fldTitleOfCourtesy" data-translatable="true"
class="select">Title of Courtesy</label>
                            <select id="fldTitleOfCourtesy" data-
bind="value:TitleOfCourtesy" data-native-menu="true" data-mini="true" data-theme="c"
data-inline="true">
                                <option value="Mr.">Mr.</option>
                                <option value="Mrs.">Mrs.</option>
                                <option value="Ms.">Ms.</option>
                                <option value="Dr.">Dr.</option>
                            </select>
                        </div>
                        <div>
                            <label for="fldFirstName" data-translatable="true"
class="ui-input-text">First Name</label>
                            <input type="text" id="fldFirstName" data-
bind="value:FirstName" class="ui-input-text ui-body-b ui-corner-all ui-shadow-
inset"/>
                        </div>

```



This example is similar to `employeeswithtranslation.htm` in the demo folder in the AppWorks Platform HTML5 SDK. See the [Sample Pages](#) for more information on other demo pages available with the SDK.

Validating a page using jQuery validation plug-in

This example describes the procedure to make a field mandatory such as input box or select box, and specify the validation rule such as minimum or maximum range. It is built on the Employees table from the Northwind Database and displays the employees form with the field validations. If all the mandatory fields are provided and the specified rules are met for the field, the employee data is saved in the Employees table.

The following steps are involved in this example:

1. Including jQuery validation plug-in
2. Adding the required attribute
3. Adding other validation attributes
4. Adding the validate method
5. Invoking the Validate method

The jQuery validation plug-in is used to validate fields and consists of many validation customization options. Refer to the jQuery website for more information.

To include the jQuery validation plug-in:

- Add the script `jquery.validate.js` in your Web page using the following code block.

```
<script
  src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.1/jquery.validate.js"
  type="text/javascript"/>
```

To add other validation attributes:

- Add the required attribute to make a field mandatory as follows.

Required elements

```
<input type="text" name="fldFirstName" data-bind="value:FirstName" id="fldFirstName"
required />
```

To add other validation attributes:

- Add other validation attributes for a field as shown in the following table.

Attribute	Description	Example
creditcard	Validates the credit card number	<input id="field" name="field" creditcard="true" />
date	Validates the date	<input id="field" name="field" date="true" />
dateISO	Validates if the provided date is an ISO date	<input id="field" name="field" dateISO="true" />
digits	Validates if the input consists of digits only	<input id="field" name="field" digits="true" />
email	Validates the email address	<input id="field" name="field" email="true" />
equalTo	Validates if two inputs have the same value by comparing them	<input id="password" name="password" /> <input id="password_again" name="password_again" equalTo="#password"/>
max	Maximum value allowed for the element	<input id="field" name="field" max="23" />
maxlength	Maximum length allowed for the element	<input id="field" name="field" maxlength="15" />
min	Minimum value required for the element	<input id="field" name="field" min="13" />
minlength	Minimum length allowed for the element	<input id="field" name="field" minlength="3" />
number	Validates if the provided number is a decimal number	<input id="field" name="field" number="true" />
required	Makes the element mandatory	<input id="field" name="field" required />
url	Validates the URL	<input id="field" name="field" url="true" />

In this example, the `minlength` and `maxlength` attributes are specified for the First Name field as follows.

Required elements

```
<input type="text" name="fldFirstName" data-bind="value:FirstName" id="fldFirstName"
required minlength="3" maxlength="15"/>
```

To add the validate record:

- Add the validate method to validate the fields based on the specified attributes.
In the following code block, `employeeForm` is the ID of the form.

Validate method

```
$("#employeeForm").validate();
```

Some options provided by the validate method are as follows:

- **Rules:** These are the key or value pairs for defining custom rules. Key is the name of an element or a group of check boxes or radio buttons. Value is an object consisting of the rule, parameter pairs, or a plain string. The following example specifies name and email elements as required using the shortcut for a single rule; a valid email address using another object literal.

Example

```
$(".selector").validate({
    rules: {
        // simple rule, converted to {required:true}
        name: "required",
        // compound rule
        email: {
            required: true,
            email: true
        }
    }
});
```

- **Error Placement:** Define the placement of the created error labels.

The following example uses a table layout for the form, placing error messages in the next cell after the input.

Example

```
$("#myform").validate({
    errorPlacement: function(error, element) {
        error.appendTo( element.parent("td").next("td") );
    }
});
```

The callback function consists of error and element parameters. The error is the error label to be inserted in the DOM and element is the validated input for relative positioning.

See the [jQuery](#) website for more information on the options available for the validate method.

To invoke the validate method:

- Invoke the validate method using the valid method.
This method returns the Boolean value true if all the conditions are satisfied; else, it returns false.

Valid method

```
$("#employeeForm").valid();
```

In this example, the valid method is invoked on the Save action. If all the mandatory fields are provided and the specified rule is met for the first name, the employee details are saved in the Employees table. The following code block displays the Employee Details.

Employee details

```
<!DOCTYPE html>
<html>
    <head>
        <title>Employees</title>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>
        <link rel="stylesheet" href="/cordys/thirdparty/jquery/cordys.min.css"
type="text/css" />
        <link rel="stylesheet"
href="/cordys/thirdparty/jquery/jquery.mobile.structure.min.css" type="text/css" />
        <style type="text/css">
            label.error {
                color: red;
                margin-top: 0.5em;
                width: 100%;
                float: right;
            }
        </style>
        <script src="/cordys/thirdparty/jquery/jquery.js"
type="text/javascript"></script>
        <script
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.11.1/jquery.validate.js"
type="text/javascript"></script>
        <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js"
type="text/javascript"></script>

        <script src="/cordys/thirdparty/knockout/knockout.js"
type="text/javascript"></script>
        <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"></script>

        <script type="text/javascript">
            var empModel; // Reference to the model
            // Create a new model on page ready
            $(function() {
                empModel = new $.cordys.model({
                    context: document.body,
                    objectName: "Employees", // Name of the Business Object
                    isReadOnly: false,
```

```

        // Common settings for all methods - read, create, update, delete,
synchronize
        defaults: {
            namespace: "http://schemas.cordys.com/NW",
            dataType: "json"
        }
    });

//Set the newly added Business Object as the currently selected item
var newEmployee = empModel.addBusinessObject({Address: "",Country:
"",FirstName: "",LastName: "",Notes: "",TitleOfCourtesy: ""});
empModel.selectedItem(newEmployee);

// Validate method for validating the form fields
$("#employeeForm").validate();

$("#btnSave").on("click", function() {
    // valid() will call the validate() method for validating the
required fields and specified rules
    if($("#employeeForm").valid()){
        empModel.create({method:"Update"}).done(function() {
            showMessage("Employee details have been saved.");
        })
    }
})
);
</script>
</head>

<body>
    <div data-role="page" id="mainPage">
        <div data-role="header" data-theme="b">
            <h1>Employees</h1>
        </div>
        <div data-role="content" data-theme="c">
            <form id="employeeForm" method="get">
                <fieldset data-bind="with: selectedItem">
                    <div>
                        <label for="fldFirstName" class="ui-input-text">First
Name*</label>
                        <input type="text" name="fldFirstName" data-
bind="value:FirstName" id="fldFirstName" required minlength="3" maxlength="15" />
                    </div>
                    <div>
                        <label for="fldLastName" class="ui-input-text">Last
Name*</label>
                        <input type="text" name="fldLastName" data-
bind="value:LastName" id="fldLastName" required/>
                    </div>
                    <div>
                        <label for="fldAddress" class="ui-input-
text">Address*</label>

```

Viewing Inbox tasks

You can use the `cordys.workflow` plug-in in the AppWorks Platform HTML5 SDK to view the inbox tasks.

This example uses the following libraries to display the tasks sent to your Inbox on a mobile device:

- **jQuery, jQuery Mobile scripts, and jQuery Mobile css** - to provide the mobile specific UI
- **KnockoutJS** - for rendering the data
- **AppWorks Platform HTML5 SDK** - for the required AppWorks Platform component plug-ins

The `GetTasks` method is invoked by the `cordys.workflow` plug-in as illustrated in the code block `Inbox Tasks`. This method retrieves all the tasks (in the form of a list) of the current user asynchronously. KnockoutJS is used for rendering the `data-bind` attribute specified for each Task object. This data binding creates a list item displayed as a child of the `tasklist` element.

Example - Inbox tasks

```
<html>
    <head>
        <title>Tasklist</title>
        <meta content="width=device-width, initial-scale=1" name="viewport"/>
        <link href="/cordys/thirdparty/jquery/jquery.mobile.min.css" rel="stylesheet"/>
        <script src="/cordys/thirdparty/jquery/jquery.js" type="text/javascript"/>
        <script src="/cordys/thirdparty/jquery/jquery.mobile.min.js" type="text/javascript"/>
        <script src="/cordys/thirdparty/knockout/knockout.js" type="text/javascript"/>
        <script src="/cordys/html5/cordys.html5sdk.js" type="text/javascript"/>
        <script type="text/javascript"> $(function() { // Create a model containing tasks, that will show them inside 'taskList' var taskModel = new $.cordys.model({ objectName: "Task", context: document.getElementById("taskList") }); // Read the tasks and put them into the taskModel $.cordys.workflow.getTasks().done(function(tasks) { taskModel.Task(tasks); }); }); </script>
    </head>
    <body>
        <div data-role="page" id="mainPage">
            <div data-role="header" data-theme="b">
                <h1>Tasklist</h1>
            </div>
            <div data-role="content" data-theme="b">
```

```

<ul data-bind="foreach:Task" data-inset="true"
    data-role="listview" data-theme="c" id="taskList">
    <li>
        <div>
            <h3 class="ui-li-heading" data-
bind="text:Activity.text">Activity</h3>
            </div>
        </li>
    </ul>
</div>
</body>
</html>

```

Working with the generic approval page

You can use a Generic Approval Page as a task in your business process. Often, business processes contain one or more steps that involve sending a document or an item for approval.

The Generic Approval Page displays the business identifiers of the process and the options specified in the message map as buttons. It can be used for approving or rejecting a task and any other action specified as options. This approval page is included in the AppWorks Platform HTML5 SDK application; after installing the application, it can be used as a task in any process.

To use a generic approval page in your business process:

1. Create an external user interface referring the approval page.

The URL of the approval task page must be

/<instance>/html5/demo/approvetask.htm.

For more information on creating an external user interface, see [Creating a User Interface Using a Web Page URL](#).

Use the following input and output schema definitions:

Input schema

```

<xsd:schema elementFormDefault="qualified"
    targetNamespace="http://schemas.cordys.com/" xmlns=""
    xmlns:tns="http://schemas.cordys.com/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <element name="ApproveTask" xmlns="http://www.w3.org/2001/XMLSchema">
        <complexType>
            <sequence>
                <element name="header" type="xs:string"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
                <element name="options" type="xs:string"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
            </sequence>
    </element>
</xsd:schema>

```

```
</complexType>
</element>
</xsd:schema>
```

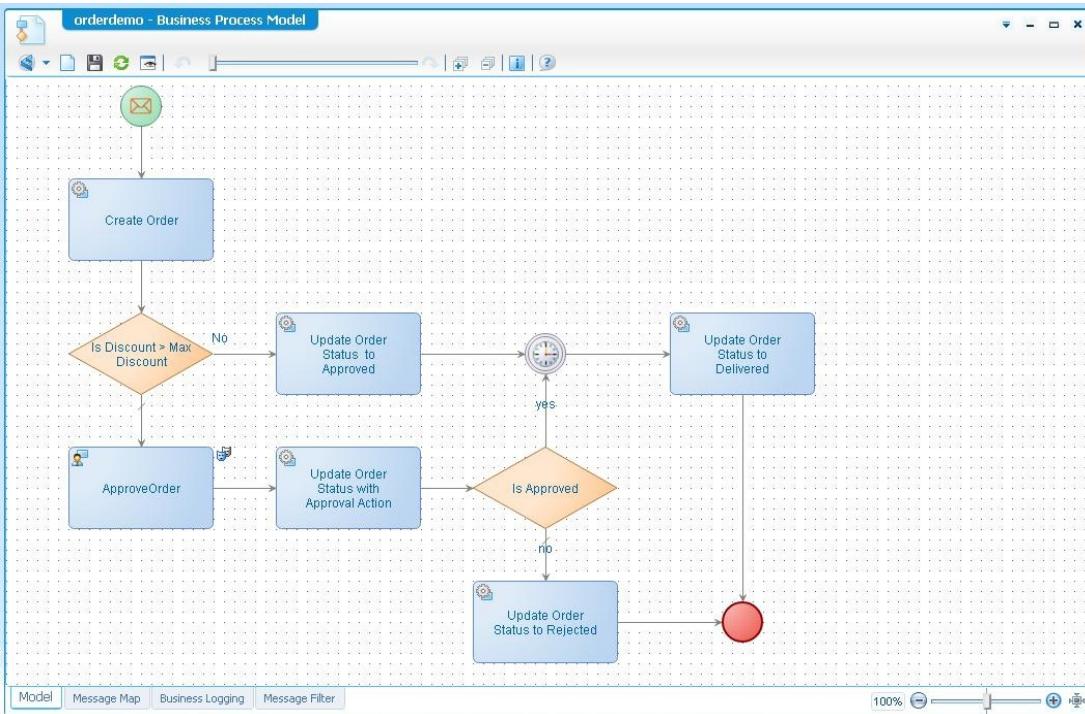
Output schema

```
<schema attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="http://schemas.cordys.com/approve"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://schemas.cordys.com/approve">
  <element name="ApproveTask_ApproveTaskDefaultDeliveryModel_OP"
  xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
      <sequence>
        <element name="response" type="xs:string"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <element name="comment" type="xs:string"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

2. Define the business identifiers for the business process.
These business identifiers will be displayed in the approval page.
3. Specify the header and options (approve or reject) in the message map.
4. You can check the response, which will be the value of one of the options, and the provided comment.

Example

The following example describes a sample order approval process.



1. The required business identifiers are defined in this example as Customer, Product, Quantity, Cost, and Discount.
For more information on creating a business identifier, see [Creating a business identifier](#).
2. Assign values to the business identifiers in the message map after the Start activity. For more information on message maps, refer to [Using Message Maps](#).
3. In the message map, specify the header and options in the Approval activity. The sample XML structure to be mapped for Options can be as follows.

```
<options>
    <option label="Approve" value="approve"/>
    <option label="Reject" value="reject"/>
</options>
```

4. Specify the message map with the response and comments after the Approval activity.
5. Execute the business process.
6. During run-time, the Task Approval Page is displayed as follows.

Customer	Cordys
Product	Web Cams
Quantity	250
Cost	600
Discount	25

If there are any attachments to the Task Approval page, they are displayed in the form of links as follows.

Cost	1500000
Customer	Cordys
Product	Web Cams

[report.txt](#)

The Generic Approval page is created and used in the business process.

Invoking AppWorks Platform REST APIs

Invoking AppWorks Platform REST APIs requires the following steps:

1. Download the REST Client
2. Configure the REST Client

3. Invoke the AppWorks Platform APIs using REST

To download the REST client:

1. Extract the **opentext-processsuite-<version>-windows.zip** file.
2. In the extracted folder, go to the `OpenTextCordysRESTClient` folder and access the `BPMService.war` file, which is the REST client.

To configure the REST client:

1. Deploy the REST Client file into AppWorks or a Web container.
2. Edit the `rest.properties` file available at `WEB-INF\classes` to configure the instance details:

<code>gatewayUrl</code>	Provide the instance gateway URL. Include the organization information in the URL that enables you to access the Case models and tasks. For example: <code>https://appworks-platform-server/home/system/com.eibus.web.soap.Gateway.wcp</code>
<code>WSDLGateway.wcp</code>	Provide the WSDL gateway URL. For example: <code>https://appworks-platform-server/home/system/com.eibus.web.tools.wsdl.WSDLGateway.wcp</code>

3. Restart the application server.

The REST client is configured.

To invoke the AppWorks Platform REST APIs:

1. Authenticate by using the Login API. (See the *AppWorks Platform API Guide*).
On successful authentication, a SAML artifact token is generated. This token is required while invoking the following REST APIs. See the *AppWorks Platform API Guide* for more details about the following APIs.

<code>Assign Task</code>	Assign a task to a user
<code>Claim Task</code>	Used by user to claim a task which is assigned to his team on the worklist
<code>Create Case</code>	Execute a case
<code>Delegate Task</code>	Delegates a task to another user
<code>Execute Process</code>	Create a new process instance or to trigger an existing process instance
<code>GetActivity Instances</code>	List all the activities of a case
<code>GetAllWorklists ForUser</code>	Lists all the worklists to which the user is assigned

GetCase Instances	List all the case instances
GetCase Variables	Retrieve case variables for a case instance
GetCaseInstance Details	Retrieve the case instance detail with state_instances and activity_instances details
GetTask Collection	Takes the task IDs and gives the task definitions for the requested tasks and also provides the detail attribute
GetTaskDetails	Retrieve the task detail (GetTask)
GetTasks ForRoles	Lists the tasks that are configured to the given roles
GetTasks ForUsers	Lists all the tasks available for the specific user based on the requested parameters
Plan Activities	Plan the activities manually
Send Message	Send a message to any process instance waiting on the Receive Message event.
Transfer Task	Transfer a task to a different target from what it is currently present
UpdateCase Variables	Update case variables for a case instance

2. Invoke the REST APIs as shown in the following Javascript example.

```
var request = new XMLHttpRequest();
request.open("POST", "http://appworks-platform-server:8080/BPMService/v1/login",
false );
request.setRequestHeader("Content-Type", "text/plain");
request.setRequestHeader("user_id", "somerestuser");
request.setRequestHeader("password", "someS3cretPwd");
request.send(null);
```

A response is generated. The response is based on the Content-Type provided in the HTTP Header.

You can start using the generated response.

Using ProcessInstances as a composite control

The ProcessInstances composite control is used to display the Process Instance Manager (PIM), which has a single view of all the instances of business processes. It monitors running processes and extracts important information pertaining to the process. This information can be the input or output messages of the instance, message map, or error messages that are displayed during the execution of an instance. During the course of time in a production environment, a user with a required authorization level may like to view the details of a process instance from a User Interface. For example, users might want to open PIM from a user interface based on instance ID or its current status.

Before you begin:

- Ensure that the ProcessInstances composite control is available at run time.
- You have adequate permissions to access it and create the reference.

To invoke the PIM from a user interface by adding custom filtering rules:

1. In the Workspace Documents (Explorer), open <solution>, right-click <project> or <folder> and select **Add Runtime Reference** > **Other**.
The New Runtime Reference dialog box opens, displaying all the run-time documents.
2. Select **Composite Control**.
The Untitled Composite Control - Composite Control wizard opens.
3. Select **Business Process Engine** > **ProcessInstance** and click **Next**.
The next screen of the window appears with the details of the Composite Control run-time document.
4. In Additional Instructions, provide additional details to install the run-time reference and click **Finish**.
The ProcessInstance composite control is displayed in the specified location.
5. Drag the **ProcessInstance** composite control from the Workspace Documents window to an XForm.
The composite control is displayed on the XForm.
6. Double-click the **ProcessInstance** control to set its properties.
A Properties sheet is displayed on the right side.
7. Set the properties of the **ProcessInstance** composite control in the Properties sheet as required.
The properties associated with the ProcessInstance control and their descriptions are as follows.

ID	Provide the string that identifies the control on an XForm. If not specified, a unique ID is automatically generated.
Process Name	Provide the fully qualified name of the process that must be monitored. For instance, a process name TestProcess is placed in the com\process\example directory, the process name must be provided as com\process\example\TestProcess.
Mode Space	Select the mode of space from the following options: <ul style="list-style-type: none"> ■ Organization - Display process instances from the organization space. ■ ISV - Display process instances from the shared space. ■ Both - Display process instances from the shared space and organization space.
Status	Select the status of process instances as a filter criteria from the following options:

	<ul style="list-style-type: none"> ■ Aborted ■ Suspended ■ Waiting ■ Running ■ Queued ■ Terminated ■ Replaced ■ Skipped ■ Complete ■ Debug ■ Debug ready ■ Obsolete
Filter XML	Enhances the list of custom filtering rules.

The ProcessInstance composite control that has the default representation based on the parameter you selected is added to the XForm.

The [message mapping](#) can be done between the ProcessInstance composite control and various composite controls on the XForm to pass parameters, based on which the rendered Web page is updated.

Using ProcessInstanceGraphicalView as a composite control

The ProcessInstanceGraphicalView composite control is used to display the graphical view of the process execution path.

Before you begin:

- Ensure that the ProcessInstanceGraphicalView composite control is available in the run time.
- Ensure that you have adequate permissions to access it and create the reference.

To display the graphical view of the process execution path:

1. In the Workspace Documents (Explorer), open <solution>, right-click <project> or <folder> and select **Add Runtime Reference** > **Other**.
The New Runtime Reference dialog box opens, displaying all the run-time documents.
2. Select **Composite Control**.
The Untitled Composite Control - Composite Control wizard opens.
3. Expand the **Business Process Engine** folder, select **ProcessInstanceGraphicalView** and click **Next**.
The next screen of the window appears with the details of the Composite Control run-time document.
4. In Additional Instructions, provide additional details to install the run-time reference and click **Finish**.
The ProcessInstanceGraphicalView composite control is displayed in the specified location.
5. Drag the **ProcessInstanceGraphicalView** composite control from the Workspace Documents window to an XForm.
The composite control is displayed on the XForm.
6. Double-click the **ProcessInstanceGraphicalView** control to set its properties.
A Properties sheet is displayed on the right.
7. Set the properties of the **ProcessInstanceGraphicalView** composite control in the Properties sheet as required.
The properties associated with the ProcessInstanceGraphicalView control and their descriptions are as follows.

ID	Provide the string that identifies the control on an XForm. If not specified, a unique ID is automatically generated.
Instance ID	Required. Provide the ID of the Process instance.
Runtime document ID	Optional and deprecated. Provide the ID of the CWS. Retrieve the Runtime Document ID using the GetProcessInstances Web service. The sample SOAP request and response follow in this topic.
Workspace ID	Optional and deprecated. Provide the ID of the CWS workspace ID. Retrieve the workspace ID using the GetProcessInstances Web service. The sample SOAP request and response follow in this topic.

The ProcessInstanceGraphicalView composite control, that has the default representation based on the parameter you selected, is added to the XForm.

The [message mapping](#) can be done between the ProcessInstanceGraphicalView composite control and various composite controls on the XForm to pass the parameters, based on when the rendered Web page is updated.

Sample SOAP request of GetProcessInstances

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Header xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <header xmlns="http://schemas.cordys.com/General/1.0/">
            <Logger xmlns="http://schemas.cordys.com/General/1.0/" />
        </header>
        <i18n:international xmlns:i18n="http://www.w3.org/2005/09/ws-i18n">
            <locale xmlns="http://www.w3.org/2005/09/ws-i18n">en-US</locale>
        </i18n:international>
    </SOAP:Header>
    <SOAP:Body>
        <GetProcessInstances
            xmlns="http://schemas.cordys.com/pim/queryinstancedata/1.0">
            <Query xmlns="http://schemas.cordys.com/cql/1.0">
                <Select>
                    <QueryableObject>PROCESS_INSTANCE</QueryableObject>
                    <Field>INSTANCE_ID</Field>
                    <Field>RUNTIME_DOCUMENT_ID</Field>
                    <Field>WORKSPACE_ID</Field>
                </Select>
                <Filters>
                    <EQ field="INSTANCE_ID">
                        <Value>001CC438-8F55-11E3-FA38-6693928596C6</Value>
                    </EQ>
                </Filters>
            </Query>
        </GetProcessInstances>
    </SOAP:Body>
</SOAP:Envelope>

```

SOAP response

```

<data>
    <GetProcessInstancesResponse
        xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <tuple
            xmlns="http://schemas.cordys.com/pim/queryinstancedata/1.0">
            <old>
                <PROCESS_INSTANCE>
                    <INSTANCE_ID>001CC438-8F55-11E3-FA38-6693928596C6</INSTANCE_ID>
                    <RUNTIME_DOCUMENT_ID>001CC438-8F55-11E3-FA38-57F94A9116C6</RUNTIME_DOCUMENT_ID>
                    <WORKSPACE_ID>001CC438-8F55-11E3-FA38-6557A555B6C6</WORKSPACE_ID>
                </PROCESS_INSTANCE>
            </old>
        </tuple>
    </GetProcessInstancesResponse>
</data>

```

Using ProcessInstanceActivities as a composite control

The ProcessInstanceActivities composite control is used to display the graphical view activities of a process in details.

Before you begin:

- Ensure that the ProcessInstanceActivities composite control is available in the run time.
- Ensure that you have the required permissions to access it and create reference.

To display the graphical view activities of a process:

1. In the Workspace Documents (Explorer), open <solution>, right-click <project> or <folder> and select **Add Runtime Reference** > **Other**.
The New Runtime Reference dialog box opens, displaying all the run-time documents.
2. Select **Composite Control**.
The Untitled Composite Control - Composite Control wizard opens.
3. Select **Business Process Engine** > **ProcessInstanceActivities** and click **Next**.
The next screen of the window appears with the details of the Composite Control run-time document.
4. In Additional Instructions, provide additional details to install the run-time reference and click **Finish**.
The ProcessInstanceActivities composite control is displayed in the specified location.
5. Drag the **ProcessInstanceActivities** composite control from the Workspace Documents window to an XForm.
The composite control is displayed on the XForm.
6. Double-click the **ProcessInstanceActivities** control to set its properties.
A Properties sheet is displayed on the right.
7. Set the properties of the **ProcessInstanceActivities** composite control in the Properties sheet as required.
The properties associated with the ProcessInstanceActivities control and their descriptions follow.

ID	Provide the string that identifies the control on an XForm. If not specified, a unique ID is automatically generated.
Instance ID	Provide the ID of the process instance.

The ProcessInstanceActivities composite control, that has the default representation based on the parameter you selected, is added to the XForm.

The message mapping can be done between the ProcessInstanceActivities composite control and various composite controls on the XForm to pass parameters, based on when the rendered Web page is updated.

Setting up continuous integration for AppWorks Platform applications

Continuous Integration is a software development practice where the team members integrate their work frequently, sometimes leading to multiple integration per day. Every integration is verified by an automated build (including testing) to detect integration errors. This approach leads to significantly reduced integration problems and allows the team to develop cohesive software more rapidly.

Continuous Integration automates the application development process to deliver high quality applications to production environments, which includes the following steps:

1. Building and packaging the application
2. Deploying the application on test or acceptance environments
3. Running automated regression tests on test or acceptance environments

Building an application

Building and packaging the application frequently, at fixed intervals or after every change, helps you constantly verify the consistency of the application. If an inconsistency is introduced, errors or warnings are raised during the build or package process. When there are no inconsistencies, the deliverable can be deployed on other environments.

There are different ways to facilitate multiple developers or multiple development teams to contribute to the development of a single application. See [Collaborative Workspace - Team Development Setups](#).

Before you begin:

- Ensure that the application sources are stored in the SCM repository where the development workspaces are configured. The workspace used to automatically build and package the application can then be configured to use the same SCM repository.
- Ensure that building and packaging the application is done on a separate build server.
- Ensure that you meet the prerequisites listed in [Using the CWS Command-line Tools](#) because the build master uses the CWS command-line tools to repeatedly execute a sequence of steps.

Configuring the build master job

After configuring the build server, configure the build master job to trigger the following sequence of steps at fixed intervals or when a change set is committed to the SCM repository that stores the application sources.

To configure the build master job:

1. Create a team development workspace using the SCM repository, which stores the CWS source models of the application.
Use the CWS command-line tool [CWS sub command - CreateSVNWorkspace](#).

2. Build the projects that together comprise the application using the CWS command-line tool [CWS sub command - validate](#).
3. Package the same projects using the CWS command-line tool [CWS sub command - package](#).
See [CWS sub command - package](#) for the location of the resulting packages.
4. Copy the packages to the location from where they are taken to be deployed on the target environments (perhaps to the target environments).
Use regular file tasks such as [Copy in Apache Ant](#).
5. Remove the team development workspace. Use the CWS command-line tool [CWS sub command - removeworkspace](#).

For every separate SCM repository, a dedicated build master job must be configured to perform the steps repeatedly.

The following tools are available to set up Continuous Integration for application development:

- [Jenkins](#)
- [AutoRABIT](#)

You can also use [Apache Ant](#) to perform regular file tasks.

Deploying an application

After building the application, you must deploy it on the target environment where manual or automated tests are performed to identify application issues. For example, if a change introduces a circular dependency between two or more packages, it can be identified. The sooner such issues are identified, the easier the root cause can be found and resolved.

Before you begin:

- Ensure that the application is developed using CWS and is packaged as a CAP that is accessible from the server where it needs to be deployed.

For Continuous integration, the package must be deployed silently. This can be done using a set of Ant tasks included in AppWorks Platform. These Ant tasks enable automating some of the common post-installation tasks such as deploying and undeploying packages, creating service groups, creating database configurations, and so on.

To deploy a package:

1. Create a build file, for example `build.xml`, and a properties file.
2. Based on application needs, add the necessary Ant tasks to:
 - a. Create the required database configurations.
 - b. Deploy the required packages
 - c. Create the required service groups and containers, and attach the Web services.
3. Install AppWorks Platform.
For more information about the silent installation of AppWorks Platform, see the

Installing AppWorks Platform in Command-line Mode section in the *AppWorks Platform Installation Guide*.

4. Install any third party components necessary for the application.
5. Run the deployment script created in Step 2.
6. Create the necessary users and assign the roles needed to access the application.

The system is ready for use or testing.

The Apache Ant tool is available for deploying packages.

Testing an application

After deploying the packages in the test environment, you must determine if any recent changes caused regression in the application by running regression tests on the latest version of the application. These regression tests must be automated as much as possible.

Before you begin:

- Ensure that the application is deployed on a regression test server.

AppWorks Platform applications can contain different types of application artifacts for which different testing tools are available.

Testing is of two types:

- **Front-end testing** - Testing the front-end of the application where the functionality available to the end users is tested including the client-side logic.
- **Back-end testing** - Testing the back-end of the application where the server-side logic of the application, which responds to the actions and events triggered in the application (either by end users or by the application itself), is tested.

Front-end testing

Front-end testing can be performed on the AppWorks Platform applications using the UIUnit framework, which is an extension of Selenium. The UIUnit framework is JUnit-based and can therefore be easily triggered from Continuous Integration tools such as Jenkins or AutoRABIT. A large number of AppWorks Platform utilities are available on top of which application-specific tests can be developed.

The test master job can only trigger the UIUnit tests or UIUnit test sets, because UIUnit tests are triggered as regular JUnit tests.

Back-end testing

Testing the back-end of AppWorks Platform involves the following two approaches:

- **JUnit** - Tests the application artifacts that are programmed in Java.
- **SoapUI** - Tests the application artifacts whose functionality is exposed through a Web service. These artifacts include:

- Business Web Services
- Business Rules
- Business Process Models
- Case Models

In addition, AppWorks Platform delivers the following default Web services to trigger instances of Business Process Models and Case Models:

- ExecuteProcess – Web service to trigger a process instance
- CreateCase – Web service to trigger a case instance

The following tools are available to test a package:

- [Jenkins](#)
- [AutoRABIT](#)
- [JUnit](#)
- [Selenium](#)
- [SoapUI](#)

Using Brava to view documents

Before you begin:

- You must install Brava and configure Brava with document store.
- Make sure that Brava is able to render documents stored in the repository configured in document store connector.
- Verify using the URL `http://<AppWorks Platform Machine Name>:8080/home/<organizationName>/app/documentservices/viewer.html?docId=<document ID>`

In the application HTML page, where Brava viewer needs to be embedded, use the following URL in an iframe.

```
http://<AppWorks Platform Machine Name>:<port number>/home/<organization name>/app/documentservices/viewer.html?docId={document Id}
```

Where:

AppWorks Platform Machine Name	AppWorks Platform Machine Name
Port Number	TomEE port number in which AppWorks Platform is configured
Organization	AppWorks Platform organization name
Document Id	Id of the document stored in the repository. This can be obtained by

GetDocument API

Example - Sample HTML page

```
<!DOCTYPE html>
<html>
<body>
<iframe src="http://demoserver:8080
/home/BravaDemo/app/documentservices/viewer.html? docId=005056C0-0008-11E5-FB24-
CCAEC30CBF6B ">
  <p>Your browser does not support iframes.</p>
</iframe>
</body>
</html>
```

Part IV

BUSINESS MODELING REFERENCES

Chapter 21

Business modeling references

Business modeling is the activity of representing both the current ('as is') and future ('to be') processes and structure(s) of an enterprise, so that the current process may be analyzed and improved. Business analysts and managers who are seeking to improve process efficiency and quality typically perform BPM. The process improvements identified by BPM may or may not require IT involvement, although that is a common driver for the need to model a business process, by creating a process master.

While creation of a business process model engulfs the scenario of an end to end application development in AppWorks Platform, there are other artifacts such as organization diagrams, the use of business calendars, user interface, Web services and others which the business process model consumes and helps you to realize your business goals successfully.

A major portion of performing the higher tasks of creating a business process model, using business calendars, and case management models is addressed in the [Business process models](#) section. Related information and supportive tasks are compiled in this section:

- [Process Modeling Reference](#)
- [Setting Process Execution Mode](#)
- [Modeling a Transaction](#)
- [Data Modeling Reference](#)

Easy way to draw constructs in design time

When designing end-to-end applications, choosing the right construct becomes imperative. This requires a knowledge of the constructs that can be associated to the current scenario. Also, looking for the required construct, selecting the right construct from the modeler Toolbox and dragging and dropping it on to the modelers itself may become a cumbersome activity that impact costs as well. The Business Analysts and users who are engaged in modeling processes or building end-to end applications often need to have a quick and easy access to required constructs so that they can model their processes. AppWorks Platform addresses this need by providing a view of all the possible constructs that can be used in association with the selected construct while designing a business process or a case model.

A sample view of the available options for various constructs

The following image displays a view of all the possible constructs that can be used immediately after Start Case.



The following image displays a view of all the possible constructs that can be used immediately after Activity construct.



To use these available options:

- Select a construct and then, click on one of the available options.

Business process modeling reference

Business process modeling in AppWorks Platform is done using the process modeling editor whose intuitive interface and the constructs therein are detailed for quick reference in the topics listed below. These topics not only provide an overview of the process modeling editor but also focus on the purpose and importance of each and every BPMN construct so that you may use the most appropriate BPMN construct that suits the needs of your business process model.

- [Constructs Used in a Business Process Model](#)
- [Business Process Modeler Interface](#)

Business processing interface

The references for business process modeling are as follows:

- [Activity Properties Interface](#)
- [Compensate Event Properties Interface](#)
- [Embedded Sub-Process Properties Interface](#)

- [Decision Properties Interface](#)
- [Delay Event Properties Interface](#)
- [End Event Properties Interface](#)
- [Exception Event Properties Interface](#)
- [For Each Properties Interface](#)
- [Select Process Instances Interface](#)
- [Receive Message Properties Interface](#)
- [Web service Definition Set Interface](#)
- [New Archive Policy Interface](#)
- [Saving Document Interface](#)
- [Print Preview Interface](#)
- [Send Message Event Properties Interface](#)
- [Start Event Properties Interface](#)
- [Independent Subprocess Properties Interface](#)
- [Case Activity Properties Interface](#)
- [Swimlane Properties Interface](#)
- [Time-out Properties Interface](#)
- [Transaction Properties Interface](#)
- [Until Properties Interface](#)
- [While Properties Interface](#)

Activity properties interface

General tab

Description	<p>Indicates a meaningful description of the current activity. You may add a meaningful description for the current activity for future reference.</p> <p>While typing the description, auto suggest help appears based on existing description of labels and fields. The activity description is translated to a preferred language if user configures the language settings. The description for the activities and models can be seen in PIM activity table and graphical view in the preferred language selected in the Welcome page.</p>
Font Size	<p>Indicates the font size of text description for the current activity in the business process model. Increase or decrease the font size of the text within an activity for visibility or easy readability. The default font size is 12.</p>
Priority	<p>This field is visible only when the activity is that of a User Interface or human activity.</p> <p>Priority indicates the importance of the User Interface activity or task</p>

	<p>on how it is executed. Select a priority when you have specific criteria to be met before an activity is executed.</p> <ul style="list-style-type: none"> ■ The execution priority of an activity can be Same as Process or Specific to Workflow. ■ When you select Specific to Workflow, the Static Value and Normal values appear by default. ■ Depending upon the importance of the User Interface activity, select the appropriate priority that can be Static Value or Read from Message. <p>The following options are displayed in a list for a Static Value:</p> <ul style="list-style-type: none"> ■ High ■ Normal ■ Low <p>If the exact parameters to indicate the priority are not known, specify the same from the Read from Message. When the priority is of type Read from Message, the PATH editor icon () appears at the end of this field. Click this icon to select a message.</p> <p>If no priority is assigned, the default priority of the task, is Same as Process.</p>
Message Type	<p>This field is visible only when the activity is that of a User Interface or human activity.</p> <p>Message is a piece of meaningful information whose intent could result in a task being performed, or for sharing purpose.</p> <ul style="list-style-type: none"> ■ Select the message type as Info or Task. <p>By default, the message type is Task.</p> <ul style="list-style-type: none"> ■ If the message type is Task you can send the task to all linked users (applicable only when the Assignee Type selected is Role) to execute that task while the business process goes into a 'waiting' state until the user acts upon the task. ■ If the assignee of the User Interface is a Role or User, then the task is considered to be completed only after all the linked users act upon it. Linked users are a group of users who are expected to act upon parts of a single task. However, if the assignee of the User Interface is a single user, the task is considered to be completed once the user acts upon it. However, if you feel that the User Interface contains only information to be shared and does not need to be acted upon, then you need to specify the message type as Info.
Execute Condition	Based upon the business requirement, you need to specify the condition for executing an activity. When you know the exact parameters to build

	<p>the condition, you may specify a Static Value and, if the exact parameters to perform the condition are not known, it is recommended to specify Read from Message where the condition should be picked up from a message that holds the condition. The activity is executed only if the result of set the condition is true, otherwise the activity is skipped. When the execute condition is of type Read from Message, the XPath editor icon () appears at the end of this field. You can build a condition by clicking this icon. If the execute condition is None, the execute condition does not apply at all and the activity is executed as a matter of fact.</p>
Execution User Type	<p>This is not applicable for an empty activity, a DecisionCase activity, or a Sub-Case Model.</p> <p>Execution User Type can be either a Process Instantiation User or a Current User. By default, the Execution User Type is set as Current User for any type of Activity with this property.</p> <ul style="list-style-type: none"> ■ The process instantiation user is the one who triggers the process. ■ The current user is the one who last acted on the process instance or the one who is currently working on a delivered task.
Message to Send	<p>This field is visible only when there is a Send Message activity.</p> <ul style="list-style-type: none"> ■ Click  and from the Selected Message dialog box, select the required message. <p>The Select a Message dialog box displays all process specific messages, Web service message and Independent Sub-process messages that are available for the current business process model.</p> <ul style="list-style-type: none"> ■ You can also Add Document or Add Runtime Reference from the Select a Message dialog box.
Input Message	<p>This field is visible only when there is a Receive Message activity or when the Trigger Type is Message for Start.</p>
Use current user context of Sub Process on callback	<p>When you want only the current user of the sub process to perform subsequent tasks of the main business process:</p> <ul style="list-style-type: none"> ■ Select Use current user context of Sub Process on callback. For example, consider that in a specific department, there are a set of tasks which need to be performed only after obtaining a Manager's approval. Also, consider that some of these tasks are part of sub processes. In such a scenario, once the Manager approves, all the tasks that are performed in this context need to have the Manager as the current user so that for future reference, it can be known as to who approved the tasks.

Application tab

The Application tab appears only when there is a User Interface or Web Service activity or a Decision Table in the business process model. All the fields on the Application tab appear in read-only mode.

Name	The name of the User Interface, Web Service or the Decision Table or the application that is used by the current activity.
Description	The description of the User Interface, Web Service, Decision Table or the application.
Application URL	This field displays the location of the User Interface. This field displays the location of the application having file extensions such as .htm, .jsp, and .caf. For example, for an application having a .caf file extension, where XForms1 is a User Interface, the application URL will be /cordys/Xform1.caf when the process is published.
UI Application Type	This field displays the type of the UI application that is attached to the User Interface, Web Service or the Decision Table. For example, <i>XForms</i> .

Workflow Model Tab

The Workflow tab appears only when there is a user interface in the business process model.

Delivery Model	<p>Specify the delivery format for the User Interface how it should appear when it reaches the AppWorks Platform Inbox. If a user interface does not have any delivery model specified for it, then business process cannot be completed.</p> <ul style="list-style-type: none"> ■ Click  to select an existing delivery model for the User Interface or External User Interface from the Select a Delivery Model dialog box. ■ To select a delivery model, you must first create a delivery model. <p>If a delivery model has Inbox Model and Email Model enabled, the Use Inbox Model and Use Email Model check boxes appear after you attach the delivery model to the task. When there is only Inbox or Email model enabled for the delivery model, only the corresponding check box, that is, only Use Inbox Model or Use Email Model check box appears. However, if a delivery model does not have Inbox model or Email model enabled, then neither of these corresponding check boxes appear.</p> <ul style="list-style-type: none"> ■ Select the appropriate check box that appears for the delivery model. ■ You can configure Inbox and or Email models for Team and Worklist
----------------	--

	<p>as well. A task may have both Inbox and Email models configured for it.</p>
Notification Subject	<p>Specify the notification subject so that the recipient of your task understands what the task contains or indicates.</p> <p>The value for this field is filled based on the description of the activity. But, if a predefined Inbox Model is selected, then the value is filled based on the subject of the selected Inbox Model.</p> <p>If you do not use the Inbox Model but use an email client as Inbox, then the task is sent to the role linked to the activity through email instead of the AppWorks Platform Inbox, and the subject of the email is the Inbox Model name or the defined Notification Subject.</p> <p>The Notification Subject can be of types - Static Value or Read from Message. Select the Read from Message option if you want to dynamically set a notification subject at runtime.</p> <ul style="list-style-type: none"> ▪ Select the Static Value option if you want to specify the subject for notification during design time.
Dispatch Algorithm	<p>A Dispatch Algorithm is the document which allows you to define your custom dispatch logic to identify the receiver of the task at runtime. Using this document, you can define your implementation logic to decide on the target. The target could be a user, role, team, or a worklist.</p> <p>Specify dispatch algorithm when there are multiple users to ensure that there is no confusion on who will perform the said task and this also helps to manage load balancing.</p>
Task Nature	<p>When multiple tasks need to be completed by users, it is recommended to specify the task nature to demarcate whether or not a user performing the first task should get the second task. Consider a scenario wherein, the first task is that of developing and the second task is testing. In such a case, the user who performs development should not be allowed to perform the testing task. Select 4Eye Nature or Rendezvous Nature property.</p> <ul style="list-style-type: none"> ▪ 4-Eye Principle: Based on the 4-Eye Principle, consider that there are two tasks - first task and second task in a business process model. In this context, the user who performs the first task can view the second task but cannot work on it. ▪ Rendezvous Principle: Based on the Rendezvous principle, consider that there are two tasks - first task and second task in a business process model. In this context, the user who performs the first task should get the second task. ▪ When the task nature is none, a user who performs the first task may or may not get the second task.

	<p>For example, consider that there are two tasks A and B. Both the tasks are assigned to a same worklist L and Users U1 and U2 are allowed to pull task from the worklist L. If the task B is in Rendezvous principle with A and if the User U1 picked the task A, then the task B also should be done by U1. If the task B is in 4 eye-principles with A and if the user U1 picked the task A, then the User U1 is not allowed to pick the task B.</p>
Initial Action	<p>Specify the initial action during the task delivery.</p> <p>The initial action can be either Create or Suspend.</p> <ul style="list-style-type: none"> ▪ The Create option is the default option and the task is delivered in the assigned state. ▪ When the Suspend option is selected, the task is delivered in the suspended state. ▪ You can provide the initial action dynamically by using the Read from Message option, but the accepted values are only CREATE or SUSPEND. ▪ If the values other than CREATE or SUSPEND are provided by using the Read from Message option, the task delivery fails and the process aborts.

Work Assignment tab

Inherit from Swimlane	<p>This field appears only when the activity is a part of a swimlane. When you select this option, you enable the activity to inherit the settings of the swimlane. For example, if a role is attached to a swimlane, all the activities within the lane inherit the role from the swimlane.</p>
Assignee Type	<p>Based on your business requirements, you may have a single user, multiple users, or teams working upon an activity or a set of activities. Therefore, it is important to select an assignee type to clearly indicate who must perform the current task.</p> <p>Select the assignee type as one of the following:</p> <p>Worklist, Role, User, or Team.</p> <p>This indicates that the task can be assigned to a worklist, role, user, or a team respectively.</p> <p>If the exact parameters to specify the assignee is not known, it is recommended to specify Read from Message, which allows you to assign a type that can be one of the following:</p> <ul style="list-style-type: none"> ▪ Static Value: For example, while selecting a Worklist type, a zoom button appears. Click  to view all the available worklists. The same is applicable to Role and Team types.

	<p>■ Read from Message: This field displays a text box for the assignee from where the value must be retrieved. Also, the XPath editor icon (🔍) appears at the end of this field. Click this icon to select a message.</p>
All linked users must execute the task	<p>When a task is delivered, the business process remains in 'Waiting' state until all the linked users complete the task; the execution of the business process will continue only after the task is completed by all the linked users.</p> <p>Select this option to enable all linked users to execute the task. If you do not select this option, the business process execution continues after any one of the linked users execute the task.</p> <p>This field appears only if the Assignee Type is Role or when Read from Message is specified for User.</p>
Assign to User	<p>This is an optional field, which appears if the message type is Task and when the Assignee Type of the task is Role, Team, or Worklist.</p> <p>This property specifies the user to whom the task must automatically be assigned after delivery.</p> <p>In run-time during the process instance execution, the XPath specified for this field must either have a valid user DN or can be left empty.</p> <p>Note:</p> <ol style="list-style-type: none"> 1. It is mandatory to have the activity execution user specified in the property Execution User Type from the same target as the specified Assignee Type. Otherwise, automatic task assignment fails and the activity aborts. 2. The activity execution aborts if an invalid user DN is provided, multiple user DNs are provided, or the user is not a part of the assignee target in the case where Role, Team, or Worklist is specified as the Assignee Type for this task. However, if the user DN pertains to a user having the Notification Admin role, even if the user is not a part of the target, the activity execution does not abort and the task is assigned to the specified user. 3. If no user DN is available in the message-map in the run-time at the specified XPath, the task is delivered to the Assignee Type without assigning the task to any user. This behavior is the same as the default behavior without this property. 4. The process executes on behalf of the Process Instantiation User or Current User that is set in the process definition. This user context is used for the delivery of a task. As the subsequent task assignment is taken up within the same user context, you must ensure that the Execution User is also part of the target to which the task is delivered.

E-Mail tab

The E-mail tab appears when the **Message Type** of the User Interface is **Info** or **Task**.

Read Recipients from Message	<p>This field appears only when the Message Type of the task is Info. Before setting the properties for the E-Mail tab ensure that you create the E-Mail Model. You need to select the recipients to whom you intend to send the info so that they are notified at the appropriate moment.</p> <ul style="list-style-type: none"> ▪ Click  associated with the To, Cc, and or >Bcc fields. The XPath Editor dialog box opens. ▪ Select the relevant elements that hold values of the recipients at runtime.
Sender details	<p>Provide the exact Email address of the sender so that the recipients know who sent the message or task to their AppWorks Platform Inbox. Provide the following details:</p> <ul style="list-style-type: none"> ▪ Sender Mail ID: Select required item from the list. Select None if you want the ID of the process instantiation user to be displayed or, select Static Value to type the sender's mail ID in the text field that appears or select Read from Message to select the sender's mail ID from the message map via XPath Editor. If the exact parameters to specify the sender Mail ID is not known while designing the model, it is recommended to specify Read from Message. ▪ Sender Display Name: Select required item from the list. Select None if you want the mail ID of the sender to be displayed as the display name of the sender or, select Static Value to type the display name of the sender in the text field that appears or select Read from Message to select the display name of the sender from the message map via XPath Editor. If the exact parameters to specify the display name of the sender is not known while designing the model, it is recommended to specify Read from Message.
Abort activity on e-mail delivery failure	<p>Select this option to ensure that the activity is aborted if the e-mail is not delivered to the recipient(s). If you do not select this option, then the process continues with the next activity even when the e-mail delivery fails.</p>

Duration tab

The Duration tab appears only when the **Message Type** of the User Interface is **Task**.

Use Business Calendar for Start and Due Time calculation	<p>You may have the need to use a business calendar when you want to specify your organizations' working and non- working days to handle tasks more efficiently. This ensures exact calculation of the time spent on a task taking into account the non-working hours and days.</p> <p>To associate your business process model with a business calendar:</p>
--	---

	<ol style="list-style-type: none"> 1. Select Use Business Calendar. 2. However, if you wish to continue with the default work-week i.e., 24* 7, you need not select this option. 3. Select a specific business calendar to link it to the activity. 4. Click Working with a Business Calendar to get more information. 5. Select this check box to attach a business calendar to the task. 6. When you select this check box a field appears from which you can select any one of the following: <ul style="list-style-type: none"> ▪ Same as that of process: When a business process model has a business calendar associated with it, all the activities within that process model are performed based on the definition of the working and non-working days within that business calendar. If this option is not selected, then the process level business calendar will not be applicable to the current process and you may have to select the activity specific option. However, an option to apply a different business calendar at the activity level is also available for convenience of ease, in view of the activity being performed from a different geographical location, and shift timings. In this context, when you apply a business calendar at the activity level and select Same as that of process, the working and non-working hours and days defined in the business calendar at the business process model overrule that of the business calendar at activity level for that task. ▪ Specific to Activity: Select Specific to Activity when you do not want the business calendar used at the process level to be applied to the current activity. In other words, you need to associate a business calendar that's different from the one used at the process level. ▪ Read from Message: Select this option when you want associate a business calendar at run time to the activity instead of applying it while designing the business process. In this context, you need to choose the required XPath expression. If the exact parameters to specify the business calendar is not known, it is recommended to specify it from Read from Message.
Start Time	Defining a start time helps the user(s) to know by when they need to start upon a task so that they may finish it as per schedule. Defining the start time helps to organize work better and to address the activity on hand in time. The tasks are delivered to AppWorks Platform Inbox and are available in the 'Calendared tasks' view. On expiry of the start time, the task is displayed in the tasks list. If the start time is not defined, there is a possibility for performing an activity ahead of schedule or behind schedule which may affect priority work.

	<p>Note: If you have not selected the Use Business Calendar option, only Static Duration (time is specified at design time) and Read Duration from Message (the condition that holds the time in the message is selected as an XPath expression - time is specified at runtime) appear in the list.</p> <p>If you selected the Use Business Calendar, the following options appear. Select a start time from the list :</p> <ul style="list-style-type: none">▪ Static Value in Business Days: When you know the exact time at which a task should be started or acted upon, you can specify a static value.▪ Read Business Day from Message: If the exact time parameters to specify the start time in days is not known, it is recommended to specify the same from the message that holds the condition as an XPath expression.▪ Static Value in Business Hours/Minutes: When you know the exact start time for the task in business hours and minutes, you can specify a static value.▪ Read Business Hours/Minutes from Message: If the exact time parameters to specify the start time in hours and minutes is not known, specify the same from the message that holds the condition as an XPath expression. <p>For the static start time, this field displays the following options:</p> <ul style="list-style-type: none">▪ Days▪ Hours▪ Minutes <p>Also, the XPath editor icon () appears at the end of this field when the start time is set to Read Business Day from Message or Read Business Hours/Minutes from Message. Click this icon to select a message element.</p>
Due Time	<p>Specifying a due time helps the user(s) to know by when they should complete the assigned task. Even when the due time exceeds, the task remains active and is highlighted.</p> <p>Note: If you have not selected Use Business Calendar for Start and Due Time calculation, only Static Duration (time is specified at design time) and Read Duration from Message (the condition that holds the time in the message is selected as an XPath expression - time is specified at runtime) appear in the list.</p> <p>If you selected the Use Business Calendar for Start and Due Time calculation, the following options appear. Select a due time from the list :</p> <ul style="list-style-type: none">▪ Static Value in Business Days: When you know the exact parameters to specify the Due time for the task, you may specify a

	<p>static value.</p> <ul style="list-style-type: none"> ■ Read Business Day from Message: If the exact parameters to specify the Due time in days is not known, it is recommended to specify the same from the message that holds the condition as an XPath expression. ■ Static Value in Business Hours/Minutes: When you know the exact parameters to specify the Due time for the task in hours and minutes, you may specify a static value. ■ Read Business Hours/Minutes from Message: If the exact parameters to specify the Due time in hours and minutes is not known, it is recommended to specify the same from the message that holds the condition as an XPath expression. <p>For the static due time, this field will display the following fields:</p> <ul style="list-style-type: none"> ■ Days ■ Hours ■ Minutes <p>Also, the XPath editor icon () appears at the end of this field when the due time is set to Read Business Day from Message or Read Business Hours/Minutes from Message. Click this icon to select a message.</p>
Due Time > Set Reminder for Duration	<p>It helps to set a reminder for the task so that the user(s) working upon it are reminded in time to complete the task before the set end time. When a reminder is set for the due time, the reminder appears in the user's AppWorks Platform Inbox as per the defined time.</p> <p>If you have not selected the Use Business Calendar check box, only Static Duration and Read Duration from Message appear in the list.</p> <p>If you selected the Use Business Calendar, the following options appear. Set reminders for due time which can be specified as:</p> <ul style="list-style-type: none"> ■ Static Value in Business Days: When you know the exact parameters to specify the Reminder for setting the Due time for the task, you may specify a Static Value. ■ Read Business Day from Message: If the exact parameters to specify the Reminder for setting the Due time for the task in days is not known, it is recommended to specify the same from the Read from Message. ■ Static Value in Business Hours/Minutes: When you know the exact parameters to specify the Reminder for setting the Due time for the task in hours and minutes, you may specify a Static Value. ■ Read Business Hours/Minutes from Message: If the exact parameters to specify the Reminder for setting the Due time for the task in hours and minutes is not known, it is recommended to specify the same from the Read from Message.

	<p>task in hours and minutes is not known, it is recommended to specify the same from the Read from Message.</p> <p>Also, the XPath editor icon ( appears at the end of this field when the due time is set to Read Business Day from Message or Read Business Hours/Minutes from Message. Click this icon to select a message.</p>
Allow Due Time Change in Inbox	<p>This option enables modification of the Due Date of a task in AppWorks Platform Inbox. When this check box is selected, the task recipient can modify the Due Date on the Modify System Attributes dialog box accessed through AppWorks Platform Inbox. Leave this check box clear (default behavior), to prevent the task recipient from modifying the due date.</p> <p>Caution: If you modify the Due Date of a task, the escalations and reminders associated with it will also be updated. Therefore, select the check box only if you are sure of allowing the Due Date to be changed.</p>
Due Time > Reminder Subject	<p>An appropriate meaningful subject for the reminder helps the user(s) performing the task to understand how much time is left to complete the said task before the end time. This helps them to prioritize their activities/daily routine and focus upon completing the task. Enter a reminder subject so that it is flashed when the reminder pops up.</p>
Include start day (Day on which Task is Delivered)	<p>This field appears only if you select the Use Business Calendar check box.</p> <ol style="list-style-type: none"> Select this check box if the start day needs to be included to calculate the due date of the task. From the Calculation Ends On list, select Start of Next Business Day if you want to include the start of next business day as the start day of the task or select End of Business Day if you want to include the end of current business day as the start day of the task.

Escalation tab

The Escalation tab appears only when the **Message Type** of the User Interface is **Task**.

Excalate On	<p>Set escalations for a task activity when it crosses the due time. Based on escalations defined, either notifications or task re-assignment should be made.</p> <p>This option will not be available for a workflow activity of type Info.</p>
Send Notification	<p>When a task is not completed within the prescribed time frame, appropriate users must be informed so that suitable action can be taken to ensure that the task is completed at the earliest possible time. To achieve this purpose, select this check box to send notification to any of the following entities:</p>

	<ul style="list-style-type: none"> ■ Manager of Worklist ■ Manager of Team ■ Manager of the User <p>User - Displays the list of available entities to whom the notification needs to be sent as selected as either Static Value or Read from Message.</p> <p>While sending the notification, if the selected target is found to be invalid, the information will be logged for reference in the logs with exact details.</p>
Reassign Task	<p>When a task is not completed within the prescribed time frame and is escalated to relevant users to inform about the delay in completing it, you may want to reassign the task to same or different user(s) or teams or a user having a specific role. To achieve this purpose, select this check box to reassign workflow tasks to any of the following entities:</p> <ul style="list-style-type: none"> ■ Worklist ■ Role ■ User <p>Team - Displays the list of available entities to whom the workflow task needs to be reassigned by selecting either Static Value or Read from Message.</p>

Monitoring tab

Configure Monitoring	<p>Monitoring at the activity level is done to keep track of activity level information at run time. If monitoring is disabled, the activity/task related information is not stored for future reference in PIM (Process Instance Manager). However, disabling monitoring improves the performance of process execution. For a long lived process, this option is selected by default. You may enable or disable monitoring for the task.</p>
Monitor Level	<p>By default, the options defined in the Default Activity Monitoring tab at the process level will be inherited by the newly created task. But these settings can be customized accordingly.</p> <ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, and user information. ■ Activity status, input and output messages: Records the input and output messages of the activity including its status. ■ Store complete activity information when activity aborts: Records activity status and input and output messages information only in case the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

	<ul style="list-style-type: none">■ Store complete activity information when activity aborts and basic information when activity completes: Records activity status and input and output messages information only in case the activity aborts. If the activity is successfully completed, only basic monitoring information such as activity or task status is recorded. <ol style="list-style-type: none">1. For an empty activity, the monitoring level will only display the Activity Status. The remaining monitoring options will not be available.2. For a decision table activity, the read-only view of the decision table will be available in the runtime only when Monitor Level is set to:<ol style="list-style-type: none">a. Activity statusb. Activity status, input and output messagesc. Store complete activity information when activity aborts (Only when activity completes successfully)d. Store complete activity information when activity aborts and basic information when activity completes (Only when activity completes successfully)
--	---

Recovery tab

The Recovery tab appears only when a Web service, Send Message, Receive Message, Decision Table, Case Model Activity, Independent Sub-process, Task is attached to an activity.

Store recovery data before the execution of this activity	When you enable the store recovery point for the activity, only the data pertaining to 'before the execution of the current activity' is stored and the process execution resumes from that point. Also, to store activity level data you must enable it at business process model level. Select the check box to store recovery data before the current activity is executed.
Store recovery data after the execution of this activity	When you enable the store recovery point for the activity, only the data pertaining to 'after the execution of current activity' is stored and the process execution resumes from that point. Also, to store activity level data you must enable it at business process model level. Select the check box to store recovery data after the current activity is executed. By default, this check box remains selected.

Webservice tab

The Webservice Options tab appears only when a Web service is attached to an activity.

Timeout in seconds	Specify the maximum time (in seconds) a Web service may take to complete. The business process sends request and waits for the
--------------------	--

	<p>response as per the time specified in this field.</p> <ul style="list-style-type: none"> ■ If the response is not received in time, the activity is aborted. ■ If this field is left empty, then the default time-out of the Process Engine (30 seconds) is used. <p>Use this option to set longer time frames for SOAP requests that take longer to complete. For example, when several tables have to be updated in the database, you may need time-out periods of more than 30 seconds. Based upon the business requirement, you need to specify the parameters for executing the Web service activity.</p> <p>To perform the Web service activity:</p> <ul style="list-style-type: none"> ■ Specify a Static Value, when you know the exact parameters. ■ Specify Read from Message, if the exact parameters are not known. The parameters are picked up from this value. <p>At runtime, the timeout in seconds is read from the XPath provided in the Timeout in seconds field. The Web service activity is executed (iterated) for each XML node in the Message Map that matches the XPath condition. This ensures that the business process does not have to be modified for every change.</p>
Perform other tasks simultaneously	<p>Select this option for Web service activities which take long time to respond. For example, calling an external Web service or updating multiple records in a database.</p> <p>If this check box is selected, the activities send the SOAP request asynchronously. In this case the process instance waits for the response and releases the resources which can then be used by other process instances. Also, response time-out caused in a synchronous call can be eliminated to prevent the activity to abort because of timeout.</p> <ul style="list-style-type: none"> ■ Select this option only for Web services which take long time, else the overhead is greater than the improved performance. ■ This option is not displayed for activities grouped as a transaction or for short lived process execution mode.
Use reliable messaging	<p>Reliable messaging uses queue to ensure that Web service calls are not lost due to unavailability of the Process Engine.</p> <p>For example, if the Process Engine is not running or not yet started, the Web service calls stay in the queue until the Process Engine is started. As a result, the Web service calls may take longer time for execution as services are in a waiting state for some service provider to read the message from the queue and process it. The actual process waits for the response or does not depend on the Output message expected selection.</p> <p>When during processing if it encounters any exception scenario the message is roll backed to the queue again and the activity status is</p>

	<p>aborted.</p> <ul style="list-style-type: none"> ■ If reliable messaging option is selected, the Perform other tasks simultaneously option is hidden because the web services called through reliable messages are asynchronous by default. ■ For short lived process if you select reliable messaging, the Output message expected option is hidden because short lived process does not wait for response. 																								
Output message expected	<p>Indicates that an output of the SOAP service is expected. This option is selected by default.</p> <p>The behavior pattern of this option along with other options is as follows:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: center; padding: 5px;">Option combinations</th> <th style="text-align: center; padding: 5px;">Behavior pattern</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input type="checkbox"/></td> <td style="text-align: center; padding: 5px;">The web service is executed in asynchronous manner and the activity waits until the response is received.</td> </tr> <tr> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input type="checkbox"/></td> <td style="text-align: center; padding: 5px;">The web service is executed in synchronous manner and the activity waits until the response is received.</td> </tr> <tr> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input type="checkbox"/></td> <td style="text-align: center; padding: 5px;">The web service is executed in asynchronous manner and the process proceeds with the next activity.</td> </tr> <tr> <td style="text-align: center; padding: 5px;"></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;">The process waits indefinitely until response comes for the service.</td> </tr> <tr> <td style="text-align: center; padding: 5px;"></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;"><input checked="" type="checkbox"/></td> <td style="text-align: center; padding: 5px;">The process does not wait after sending the request, it completes the activity and moves to the next activity.</td> </tr> </tbody> </table>	Option combinations			Behavior pattern	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The web service is executed in asynchronous manner and the activity waits until the response is received.	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The web service is executed in synchronous manner and the activity waits until the response is received.	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	The web service is executed in asynchronous manner and the process proceeds with the next activity.		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	The process waits indefinitely until response comes for the service.		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	The process does not wait after sending the request, it completes the activity and moves to the next activity.
Option combinations			Behavior pattern																						
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The web service is executed in asynchronous manner and the activity waits until the response is received.																						
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	The web service is executed in synchronous manner and the activity waits until the response is received.																						
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	The web service is executed in asynchronous manner and the process proceeds with the next activity.																						
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	The process waits indefinitely until response comes for the service.																						
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	The process does not wait after sending the request, it completes the activity and moves to the next activity.																						
Return standard error on SOAP fault	<p>If the reliable messaging option is selected and process execution mode is short lived then this option is hidden.</p> <p>If you want to be notified when a SOAP fault occurs in a specific structured message, select this option to map different types of SOAP faults returned by a Web service (activity) to the Process Engine's standard SOAP faults, that is, communication failures. These faults are treated as a single type of exception so that they can be handled in a common manner. If this option is cleared, any SOAP fault returned by</p>																								

the Web service is returned as it is.

Behavior pattern of various options

Legend

	Hidden (Not Applicable)
	Selected
	Not Selected

			Web service options		
Message Exchange Patter	Execution Behavior	Applicable Process Mode	Perform other tasks simultaneously	Use reliable messaging	Output message expected
Request - Response	The business process sends a request and waits for the response message or goes into timeout. The receiver processes the request and replies with a response message or a fault.	Long Lived Short Lived	 	 	
Request - Delayed Response	This pattern is similar to Request-Response pattern except timeout. After sending a request, business process indefinitely waits for the response message. During this period, the business process releases the resources which can then be used by other process instances. Request and Response messages are independent, and message correlation is used to map a response to the corresponding request.	Long Lived			
Fire-and-Forget	The business process sends a request and proceeds to the next activity without waiting for a response.	Long Lived Short Lived	 	 	
Request-Delayed Response (Reliable)	This is similar to the Request - Delayed Response pattern but messaging queues are used for asynchronous and reliable delivery of the message.	Long Lived		 	
Fire-and-Forget (Reliable)	This similar to the Fire-and-Forget pattern but messaging queues are used for asynchronous and reliable delivery of the message.	Long Lived Short Lived	 	 	

Retry tab

The Retry tab appears only for Web service activities. It allows for retrying the Web service activity in case of an error.

Retry Count	Retry count indicates the number of retry attempts made for the Web
-------------	---

	<p>service activity when it is aborted. The default value for Retry Count is zero. If you do not specify the retry count, an attempt to restart the Web service activity to execute it is not made.</p> <ul style="list-style-type: none"> ■ After specifying a retry count , if an error of type Communication Failure, SOAP Fault etc., occurs, the activity will be executed for the number of retry counts specified with the time delay value specified as retry interval. ■ If the activity is executed for a specified number of retries, the activity will abort with the same error as it aborts without any retry. For example, if retry count is given as 2, the activity will execute for 3 times if an error situation is not corrected during retry interval and finally the activity will abort with the error. ■ In case if an exception handler is attached to the activity, it will be executed only after the activity is executed for the given number of retry attempts.
Define retry condition	Select this check box to specify the Retry condition field.
Retry condition	When you specify the retry condition, the activity is retried only when the evaluation of the Boolean condition returns 'true'. This condition is evaluated before every attempt to retry the activity. The Retry condition is an optional parameter and can hold an XPath condition returning Boolean result.
Retry interval in hours	Specifying the retry interval in hours helps you to ready or streamline related activities so that when the next retry attempt happens the activity is successfully executed. Specify the time delay between the retry attempts which is entered in hours.
Retry interval in minutes	Specifying the retry interval in minutes helps you to ready or streamline related activities so that when the next retry attempt happens the activity is successfully executed. Specify the time delay between the retry attempts which is entered in minutes.
Retry interval in seconds	Specifying the retry interval in seconds helps you to ready or streamline related activities so that when the next retry attempt happens the activity is successfully executed. Specifies the time delay between the retry attempts which is entered in seconds.
	The minimum value for the retry interval is 10 seconds.

Debug tab

The Debug tab is used to send an instantiated process directly to the debug ready state in the **Process Instance Manager**.

Debug Process at Activity	While executing a specific activity, you may want to debug the business process to view errors, if any, and correct them so that all the activities
---------------------------	---

	<p>are executed as intended and the business process is successfully implemented. In such a situation, you may choose to debug or not to debug the business process when the current activity is executed. If you choose to debug the business process, then you need to specify whether you wish to perform the debug operation each time the current activity is executed, or based only on a specified condition. Specify these parameters based on your requirement or expectations of the business process. You have the following three options in the Debug Process at Activity list to choose:</p> <ul style="list-style-type: none"> ■ No: Select this option if you do not want to debug the business process when the current activity is being executed. ■ Always: Select this option if you want to debug the business process each time the current activity is executed. When you select this option, the Start to Debug list appears. You may select either Before Activity or After Activity from the Start to Debug list. This will indicate whether the debug operation should be performed before or after the current activity is executed. ■ Conditional: Select this option if you want the business process to be debugged based on a condition that you set. When you select this option, the Start to Debug list and Debug Condition fields appear. From the Start to Debug list, you may select either Before Activity or After Activity from the Start to Debug list. This will indicate whether the debug operation should be performed before or after the current activity is executed. Click  associated with the Debug Condition field to browse and select a debug condition from the message map that appears in the XPath Editor. The Debug Condition field appears only if you select Conditional from the list. Provide the relevant XPath expression in the Debug Condition field.
--	--

Estimated Duration tab

The Estimated Duration tab appears only when the **Message Type** of the User Interface is **Task**.

Estimated Lead Time	<p>The estimated lead time, or the time duration from assignment of a task to a certain user, until the time the task is completed by the user. You can define the elapsed time for each activity (that is a URL or Application) in the process. The Elapsed Time field has the following fields:</p> <ul style="list-style-type: none"> ■ Ideal- Specify the ideal time for completing the activity. ■ Expected- Specify the time in which the activity is expected to be completed. ■ Minimum- Specify the minimum time the activity should take to
---------------------	--

	<p>complete.</p> <ul style="list-style-type: none"> ■ Maximum- Specify the maximum time the activity should take to complete. Specify the above values and click the Calculate button to calculate the average lead time. These values are translated into KPI setting values for the Activity Cycle Time KPI.
Estimated Spend Time	<p>The spend time is the time duration from when the task is started, from the AppWorks Platform Inbox, until the time the task is completed by the user. You can define the actual time spent for each activity (that is a URL or Application) in the process. The Elapsed Time field has the following fields:</p> <ul style="list-style-type: none"> ■ Ideal- Specify the ideal time for completing the activity. ■ Expected- Specify the time in which the activity is expected to be completed. This value must not be less than the minimum value. ■ Minimum- Specify the minimum time the activity must take to complete. ■ Maximum- Specify the maximum time the activity must take to complete. This value must not be less than the expected or the ideal values. <p>Specify the above values and click the Calculate button to calculate the average spend time. These values are translated into KPI setting values for the Activity Cycle Time KPI.</p>

Attachments tab

The Attachments tab appears only when there is a User Interface in the business process model.

Assign an Attachment	Click  to create an attachment. Select an attachment definition from the list to assign it to an activity.
Read	Select Read to allow the user to download and view the attachment.
Write	Select Write to download the attachment and modify the contents, or add a new attachment to the task.
Delete	Select Delete to enable the user to view the attachment, modify the contents in the attachment, add another attachment, or delete the attachment too.

Links tab

The Links tab appears only when there is a User Interface in the business process model.

URL	When you have Website links which the user(s) should visit to perform the assigned task, specify the URL of the website attached to the task.
Description	A description of the website attached to the task.

KPI tab

The KPI tab appears at the Business process and Activity level.

Create a KPI	Click  to add a KPI to the BPM. A KPI wizard is displayed. See Creating a KPI on a Business Process Model topic for the detailed procedure.
Edit	To edit or view the KPI, double-click the KPI and make the relevant changes to it in the KPI editor that appears.
Delete	Select Delete for the KPI to be deleted and click  .

Annotation tab

Annotation	Additional notes or comments on the task, if any.
------------	---

Business process model properties interface

The *Properties - Business Process Model* helps in viewing and setting the properties of a business process model.

General tab

Contract	<p>If you want to specify your own implementation for a Web service action:</p> <ol style="list-style-type: none"> 1. Select Contract to create a Contract. The Binding Operation field appears. 2. Click  to select a binding operation from the Select a Binding Operation dialog box and click OK. 3. When you select a binding operation, a dummy business process model is created with Start, (dummy) Activity, and End events. The input message of the binding operation is added to the Start event and the output message of the binding operation is added to the End event. 4. It is recommended that you add the required implementation to the business process model to make it executable. When you execute the Web service, the business process model is triggered. <p>To create a Contract, you must first generate the AppWorks Platform Web service operations for external Web services.</p>
Description	Provide a description for the business process model. While typing the description, auto suggest help appears based on existing description of labels and fields. If you configure the language settings, the model description is translated to a preferred language . The description for the activities and models can be seen in the Process Instance Manager activity table and graphical view in the preferred language selected in user preferences. This is a mandatory field.

Enable Crash Recovery	<p>Data loss situations are common in all computers due to corruption, user errors, virus infections, or system failures. All result in loss of critical data. Crash recovery is the ability to restore a process instance from the point of execution failure without having to restart it. If a business process encounters an issue due to system or processor failure, you can restore the system using the crash recovery feature at runtime.</p> <p>For example, a business process is scheduled to run 400 iterations, but the Business Process Management Service Container crashes after 200 iterations. In such a case, if the Business Process Management Service Container is restarted, the entire process has to begin again (i.e., from the first iteration). However, if crash recovery is enabled, the business process restarts from the point at which the process was terminated (in this case, from iteration 201).</p> <p>Select this check box to enable crash recovery for the business process model. When you enable crash recovery at the business process model level, the crash recovery feature is automatically applied to all the activities with that business process even if it is not enabled at the activity level.</p> <p>When the crash recovery feature is enabled only at activity level, crash recovery is applicable only for that particular activity. However, when crash recovery is enabled at both business process model and activity levels, then the property set at the model level takes precedence.</p> <ul style="list-style-type: none">■ If you select Store Recovery Data, you can store recovery information of an activity that is useful if the process aborts at this activity. Ensure that you select Enable Crash Recovery on the General Enable Crash Recovery tab before you select Store Recovery Data.■ This option is only applicable to business processes of type Page Flow and Long Lived.
Execution Mode-Reference topic	To know which execution mode to select, see Process Execution Modes and to set the process execution mode, see The mode in which the business process model should be executed .
Execution Priority	<p>When you execute multiple business processes, the corresponding process instances are created. If you want to specify the execution priority for these process instances, select an execution priority for the process instances at the business process level from the Execution Priority list. However, if you do not specify the execution priority, the process instances are executed in FIFO (First-In, First-Out) order. It displays the process execution priority levels of the business process model in a list, which assumes importance at runtime.</p> <p>You can select one of the process execution priority levels: Highest,</p>

	<p>High, Normal, Low, and Lowest. The default priority level is Normal. At runtime, a business process model, which has a priority set to Highest takes precedence over other business process models that are set to lower priority levels. The logic of execution of the process instances according to priority is based on an algorithm, which is set through the Business Process Management service configuration.</p>
Namespace	<p>A namespace is a set of rules that determine how network resources are named and identified. This field displays the namespace defined for the business process model as a complete URL. The namespace is editable.</p> <p>The default namespace (http://schemas.cordys.com/default) is generated by the business process.</p> <ul style="list-style-type: none"> ■ To change the default namespace to a custom namespace, edit the default namespace. The prefix of a namespace is editable. ■ When the prefix of a namespace is changed, the change is reflected in all corresponding instances of the prefix in the message map or wherever it occurs. <p>You can change the default namespace only for namespaces that the business process has generated.</p>
Register Event Listener class FQN	<p>You can configure an event listener that implements certain logic appropriate for process events. The event listener methods are invoked by the process engine during the execution of a process instance at appropriate process events.</p> <p>For information on how to configure the event listener, see Working with Process Event Listeners.</p>
Use Business Calendar	<p>If you want to associate your business calendar having specific working and non-working days with the business process model:</p> <ul style="list-style-type: none"> ■ Select Use Business Calendar. The Business Calendar field appears. ■ Select an available business calendar by browsing the list from the Select a Business Calendar dialog box. <p>For more information on how to use a business calendar and its functions, see Business Calendar.</p>

Monitoring tab

Configure Monitoring on Process Level > Monitoring Level	Monitoring at the business process level keeps track of process level information at runtime. If monitoring is disabled, the process related information is not stored for future reference from the Process Instance Manager (PIM). However, disabling monitoring improves the performance of process execution. Enable or disable monitoring for the process.
--	--

	<p>For a long lived process, this option is selected by default.</p> <ul style="list-style-type: none">▪ Process status: Records basic process information (status, timestamp, user information, and so on).▪ Process status, input and output messages: Records basic process information (process status) and input and output messages of the business process.▪ Process status, input, output messages and message-map: Records basic process information (process status), input and output messages, and the message map of the business process.▪ Store complete process information only when process aborts: Records basic process information, such as process status, input and output messages, and the message map of a business process only when the process aborts.▪ If you select the Enable Crash Recovery check box on the General tab of the Business Process Model property sheet and at the Business Process Model service container level, it stores complete information of the process even if the process does not abort.▪ Store complete process information when exceptions are handled for aborted activities: Records complete process information, such as process status, input and output messages, and the message map of a business process only when an activity within the process is aborted and after the exception is handled to continue with process execution. If this option is used, then information (such as the issue encountered when the activity was aborted and how the exception was handled) is available to debug the issue.▪ Store complete process information when process aborts and basic information when process completes: Records complete process information (such as status, input and output messages, and the message map) when the process aborts; records basic information (such as process and activity statuses) when the process completes.▪ Default monitoring settings for long lived processes and page flow: Process status, input and output messages, and message-map for process monitoring.▪ Default monitoring settings for short lived processes: Process status for process monitoring.
Default Settings For All Activities > Configure Monitoring > Monitor	Monitoring at the activity level is done to keep track of activity level information at runtime. If monitoring is disabled, the activity related information is not stored for future reference

Level Default Settings For All Activities > Store Recovery Data	<p>from PIM. Also, to monitor activity level data, you must enable it at the business process model level. However, disabling monitoring improves the performance of process execution. Enable or disable monitoring for the activity.</p> <ul style="list-style-type: none"> ▪ Activity status: Records basic activity information (status, timestamp, and user information). For a short-lived process, if you select a single activity for monitoring, this field appears. ▪ Activity status, input and output messages: Records basic information and input and output message of the activity. ▪ Store complete activity information only when activity aborts: Records basic information such as input and output messages only when the activity aborts. ▪ Store complete activity information when activity aborts and basic information when activity completes: Records complete activity information (such as status, input and output messages, and the message map) when the activity aborts and basic activity information such as status when the activity completes. <p>Tip: If you want to configure default monitoring and crash recovery settings for all the activities in the business process, select the Default Settings for All Activities check box. The Configure Monitoring and Store Recovery Data check boxes are enabled.</p> <p>If you select Store Recovery Data, you can store recovery information of an activity that can be useful if the process aborts at this activity. Ensure that you select Enable Crash Recovery on the General Enable Crash Recovery tab before you select Store Recovery Data.</p> <p>For a long-lived process, when you select properties of a single activity, all the above three fields appear.</p> <ul style="list-style-type: none"> ▪ Default monitoring settings for long lived and page flow processes: Activity status, input and output messages. ▪ For short lived process, by default, no monitoring is enabled for a process and its activities. <p>If you want to set the store recovery point for all activities in the business process model, select this option. You can set the recovery point for each individual activity by setting this property on the Recovery tab. You may either enable or disable the store recovery point for all the activities at a time. However, if you also enable or disable the store recovery point at the individual</p>
--	--

	<p>activity property at the process level, only the activity level property prevails.</p> <p>When you enable the store recovery point for all the activities in the Default Settings for All Activities, only the data pertaining to before the execution of all activities is stored and the process execution resumes from that point. If Store recover data is enabled at the process level, then data pertaining to after execution of this activity is stored for recovery. Also, to store activity level data, you must enable it at the business process model level</p>
--	---

Business Identifiers tab

Publish Business Identifier Values using	<p>This option enables you to retrieve the business identifier values published through a Web service.</p> <p>To use this feature, you must provide the implementation to the PersistBusinessIdentifiersOperation operation as specified in the PersistBusinessIdentifiers WSDL.</p> <p>The Web service will be invoked after the execution of any activity, which updates the business identifier value. It enables you to persist the changes to the business identifiers in custom repositories. Such storage is useful if applications need to have specific queries over the business identifiers and their values. The process engine calls the Web service asynchronously. Any error in triggering the Web service raises an alert and the details are logged in the business process engine service container log file. Errors from the Web service do not abort the execution of the process instance.</p> <p>Following are the possible values and its related functionality:</p> <ul style="list-style-type: none">■ None: The engine does not trigger any request, that is, business identifier values are not published to the Web service.■ Non-Transactional No-Reply: The service container that implements the Web service (PersistBusinessIdentifiers) need not be configured using queues. If the implementor service container is stopped, then the engine only raises an alert and logs the error in the log file. If the call is successful, the engine does not wait for the response.■ Transactional No-Reply: To use this option, the service container that implements the Web service (PersistBusinessIdentifiers) must be configured using a queue (JMS). When the engine calls the Web service, the SOAP request is sent to the container's queue that was configured without waiting for the response.
--	---

Defer Publish of BI values to database until next recovery point	<p>When this property is enabled, the publishing of business identifier values to the PIM database is deferred until the next recovery point. This helps in reducing the number of database updates and improves performance.</p>
Business Identifiers	<p>Business (process) identifiers are specific attributes of the business process that are defined in the design time environment. The main purpose of business (process) identifiers is to use them for identification of instances of the process in the Process Instance Manager. For more information, see Creating Business Identifiers, Associating Business Identifiers to a Business Process Model, and Using Business Identifiers.</p> <ul style="list-style-type: none"> ■ Create a business identifier first before associating it with the business process model. Perform the following steps to associate a business process model with a business identifier: ■ Click . The Select a Business Identifier dialog box opens displaying a list of existing Business Identifiers. ■ Select required <Business Identifier> from the Select a Business Identifier dialog box. ■ To discard and select a different business identifier, click  to browse and select the required business identifier. ■ The Show in Runtime check box is selected by default. This helps you view the business identifier in PIM. However, if you do not want to view the business identifier in PIM, clear the Show in Runtime check box. ■ To delete a business identifier, select the required business identifier and click . ■ To move the selected row up or down, use  or  as required. <p>It is possible to select and simultaneously open only four Business Identifiers in PIM. This is to ensure that the performance of PIM is not impacted.</p>

Namespaces tab

Namespaces	<p>Displays prefixes and their corresponding namespaces that are used in the business process model in the Prefix and Namespace columns respectively.</p> <p>All prefixes are editable. Namespaces that are generated only by the business process model are also editable. When the prefix of a namespace is changed, the change is reflected in all corresponding instances of the prefix in the message map or wherever it occurs.</p>
------------	---

	However, you can change the default namespace only for the namespaces that the business process has generated
--	---

Links tab

URL	Type any URL that you want to make accessible for additional information when designing the business process.
Description	Type the description of the URL.

Attachments tab

When working with processes that involve human interaction, there is a need to share unstructured data (such as documents, images, and so on) across the tasks in the process by end-users working on that process. The attachment configuration enables the application developers to define and configure attachment at business process level and set appropriate authorizations at each activity level.

Based on these authorizations defined at each activity level, the end-users working on different tasks in Inbox, can access the attached documents during the process execution. The attachments can also be shared between various processes (BPM to BPM, BPM to Case) by relating assignments in the Message Map. For any BPM or Case sub-process associated to the BPM model, the related attachment definitions are also available for mapping. When such mapping is considered between attachment definitions, the application developers must ensure that the MIME types of the attachment definitions are compatible.

Attachment Location	<p>Optional. This is useful if Process Model designers want to store attachments in custom specified locations. A Process designer can choose one of the following options:</p> <ul style="list-style-type: none"> ■ Default: This is selected by default and attachments are stored in the default location. ■ Custom: Select this option to specify the custom location and select the location type from the select box. <ul style="list-style-type: none"> • Static: Specify the static location. - • Read from message: Specify the path from the process message map (Process instance properties or Process specific message). <p>By default, all the attachments are uploaded to a standard location on the document store if there is no custom location specified. This default location evaluates as per the following pattern: bpm/<Process Name>/<process Instance Id>/<AttachmentDefinitionName></p> <p>Example : bpm/HomeLoanApplication/5337b252-d28c-4607-a0df-59654c5ebf4d/HomeLoanDocuments</p>
Attachment Name	Name of the attachment definition.

Supported Document Types	Select the MIME type to which the attachments in runtime must comply. If a required document type is not available in the list, select Any file type .
--------------------------	---

Annotation tab

Annotation	Type any additional notes or comments on the business process model that you would like to use long after designing the process.
------------	--

Contract first development

When creating Web services, there are two development styles: Contract-Last (also known as Code-First approach) and Contract-First. When using a contract-last approach, you start with the code, and let the Web service contract (WSDL) be generated from that. This is ideal for small applications and for learning Web services.

When using contract-first, you first create the WSDL contract or document , supporting schema and then use code to implement the said contract. For composite applications, long-lasting Web services and service-oriented architecture (SOA), contract-first way of development has advantages that usually outweigh the ease of method-first thinking. Of these two techniques, contract-first is the recommended way to create a Web service as in this approach the main focus is given to the messages that come in and go out from the Web service. This is also the correct approach to use in the SOA. A contract-first approach results in better long term development, interoperability and maintenance.

When you abstract code into services, contract-first makes it easier to use these services as pieces and plug them into where ever they might be needed. By first building the service's contract, it provides the structure to define and make those pieces work together. This allows you to consider a service as a stand alone piece of functionality. A service contract includes both functional and non-functional requirements. The functional requirements include what the service does, the service interface and the means of invocation. The non-functional requirements include service level agreement, security and quality of service, transactional constraints, process and semantic definitions.

Benefits of Contract First Development (CFD)

You need contract-first to give the service its own scope of responsibility. A basic tenet of SOA is that a service is an autonomous unit of functionality. Also, it provides a definitive scope of what the service needs to provide.

- Increased visibility: A contract increases visibility and provides a neat package of functionality than can be used as a component in application infrastructure.
- Enables test-driven development: Because you get to know the functional requirements, you can develop test cases much before the code is even created, to determine if the services perform the required functionality within the constraints of the contract.
- Improved stability: Stability of the system is improved by versioning of contracts. Once a contract is established, it cannot be changed and it must be supported as long as consumers exist for it. However, new versions can be cloned and modified. By having both versions of the contract supported, new functionality can be built and made

available to early adopters while the old contract is still supporting other applications. This allows a company to be agile in reacting to business changes while not breaking existing applications.

Compensate event properties interface

General tab

Description	A description of the Compensate event.
-------------	--

Monitoring tab

Enable Monitoring	Enable or disable monitoring for the activity.
Monitor Level	Activity status: Records activity status, timestamp, user, etc.

Annotation tab

Annotation	Additional notes or comments on the Compensate event, if any.
------------	---

Embedded sub-process construct properties interface

The following table lists the tabs and fields of the embedded sub-process construct properties interface.

General tab

Description	Additional notes or comments on the Compensate event, if any.
Font Size	Decrease or increase the font size of the text within the embedded sub-process. The default font size is 12.
Execute Condition	Specify the actual condition or the message from where the condition should be picked up for execution. Select one of the following options: <ul style="list-style-type: none">■ None: This option is displayed by default. Select this option if you do not want to specify any execute condition.■ Static: Select Static option to specify the actual condition for execution. You can build a condition by clicking the XPath editor icon () appearing at the end of this field.■ Read from Message: Select Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. Click to build a condition. Note: <ul style="list-style-type: none">■ Ensure that the dynamic condition, which is taken from the Message Map does not start with a slash.

- If a message contains an XPath expression, then the XPath need not be in single quotes.
For example, if the message is /mycondition and it contains an expression as follows:
GetEmployeeOutput/GetEmployeeResponse/tuple/
old/Employees/City = "London", the XPath will be as follows:
GetEmployeeOutput/GetEmployeeResponse/tuple/
old/Employees/City = "London".
- In an Embedded Sub-process construct, the Execute Condition option can be used instead of the Decision construct. A small Decision icon appears on the left at the bottom of the construct to indicate that the embedded sub-process includes a condition to be executed.

Monitoring tab

Enable Monitoring	Enable/disable monitoring for the embedded sub-process.
Monitor Level	<ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, user, etc. When you set the monitor level at this parameter, only the activity status is monitored. This helps in getting the updated activity status and to spot errors should the activity abort. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. When you set the monitor level at this parameter, the activity status and the input and output messages of the message map is monitored. This helps in tracking and spotting any error occurring at this level. ■ Store complete activity information if activity aborts: Records activity status and input and output messages information only if the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

Annotation tab

Annotation	Additional notes or comments on Embedded Sub-process construct, if any.
------------	---

Decision construct properties interface

General tab

Description	A description of the decision.
Type	Select the decision type to use. The available types are: <ul style="list-style-type: none"> ■ Inclusive: The specified conditions are evaluated individually. All the conditions that evaluate to true are considered, and all or a

	<p>combinations of paths are executed.</p> <ul style="list-style-type: none"> ■ Exclusive: The specified conditions are evaluated in the order specified on the Conditions tab. The first condition that evaluates to true is considered and its path is executed. The remaining conditions are skipped. ■ Parallel: No conditions are evaluated. An incoming path is split into two or more parallel paths that are executed simultaneously. You can also use this decision type to combine multiple incoming paths. <p>The Parallel decision type is only available when designing business process models in AppWorks Platform.</p>
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12

Conditions tab

Activity	Displays the activity name. You can set the order in which these activities can be considered for a decision.
Name	Displays the name of the condition.
Condition Type	<p>Displays the condition type. The available condition types are:</p> <ul style="list-style-type: none"> ■ Static Value: Specify a static type of condition where you can manually type the XPath of the element defined in the Message Map. ■ Read from Message: Specify a condition by reading the condition from the Message Map. ■ Default: Set a default condition to ensure that the action leading from this condition is executed if the other specified conditions are False.
Condition	Displays the complete XPath of the selected condition type. You can customize your condition in the XPath Editor by clicking  .

Monitoring tab

Configure Monitoring	Enable or disable monitoring for the activity.
Monitor Level	Monitor level is Activity status by default and this is a non-editable field.

Annotation tab

Annotation	Additional notes or comments on the Decision, if any.
------------	---

Delay event properties interface

General tab

Description	A description of the activity.
-------------	--------------------------------

Execute Condition	<p>The Execute Condition can be of types: None, Static Value, or Read from Message.</p> <ul style="list-style-type: none"> ■ Select None to specify that there is no execution condition for the business process. ■ Select Static Value to specify the actual condition for execution. ■ Select Read from Message to read the condition from the Message Map. This ensures that the business process does not have to be modified for every change. Click  to select the process specific message from the XPath editor. <p>Ensure that the dynamic condition, which is taken from the Message Map, does not start with a slash.</p> <p>If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is /mycondition and it contains an expression:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple /old/Employees/City = "London", the XPath will be: GetEmployeeOutput/GetEmployeeResponse/tuple/ old/Employees/City = "London".</pre> <p>All XPath expressions are supported in Message Map 'Expression'.</p> <p>In a Delay Event, the Execute Condition option can be used instead of the Decision construct. You can view a small Decision () icon at the bottom left of the construct to indicate that the Delay Event includes a condition to be executed.</p>
-------------------	--

Duration tab

Use Business Calendar	<p>This field appears only when you select Use Business Calendar.</p> <ul style="list-style-type: none"> ■ Specific to Activity: Select a particular activity that appears in the Business Process dialog box for using the business calendar. ■ Same as that of Process: Selecting this option allows you to use the business calendar for the entire business process. ■ Read from Message: Selecting this option allows you to specify the XPath for reading the business calendar dynamically from the Message Map.
Duration	<p>You can specify the duration for a Delay Event that is related to the business days, hours or minutes. When you select the Use Business lendar option, the business days are calculated based upon the business days specified in the business calendar. Otherwise, the business days are calculated including holidays and non-working hours.</p> <p>In PIM, the duration for the delay and timeout activity may at times be less than the actual duration with a difference of 1 to 0.5 In seconds.</p> <p>The following options appear when the Use Business Calendar check box</p>

	<p>is left unselected:</p> <ul style="list-style-type: none"> ■ Static Duration: Enter Days, select Hours, Minutes, and Seconds from the respective lists. ■ Read Duration from Message: Click  and from the XPath Editor, select required message, validate, test, and click OK. The input for this field should be in XML Schema Duration Data Type. For example: PT5M, which stands for 5 minutes. <p>The following options appear in the Duration list only when you select Use Business Calendar :</p> <ul style="list-style-type: none"> ■ Static Value in Business Days: Enter the business days as a fixed value for the number of days in the Days field. ■ Read Business Days from Message: Click  and from the XPath Editor, select required message, validate, test, and click OK. ■ Static Value in Business Hours / Minutes: Select Hours and Minutes from the lists. ■ Read Business Hours / Minutes from Message: Click  and from the XPath Editor, select the required message, validate, test, and click OK.
Calculation ends on	This field appears only when you select the Use Business Calendar check box. <ul style="list-style-type: none"> ■ Start of Next Business Day: Sets the calculation to end on the start of next business day. ■ End of Business Day: Sets the calculation to end on the end of next business day.
Include start day	This field appears only when you select the Use Business Calendar check box. <ul style="list-style-type: none"> ■ Select this option if the start day needs to be included to determine the end of calculation of the due date.

Monitoring tab

Configure Monitoring	Enable/disable monitoring for the activity.
Monitor Level	Activity Status: This field records activity status, timestamp, and user.

KPI tab

Create a KPI	<ul style="list-style-type: none"> ■ Click  to add a KPI to the BPM. A KPI wizard is displayed. <p>Refer to Creating a KPI on a Business Process Model topic for the detailed procedure.</p>
--------------	--

Edit	To edit or view the KPI: 1. Double-click the KPI. The KPI editor opens. 2. Make the relevant changes.
Delete	Select to delete the KPI and click  .

Annotation tab

Annotation	Additional notes or comments on the Delay event, if any
------------	---

End event properties interface

General tab

Description	A description of the End event.
End Type	The various types of End events are Message, Error, Rollback and None. <ul style="list-style-type: none">■ If the End event is in a transaction then the End type Abort appears.■ If the End event has an embedded sub-process, the End type Terminate appears.
Output Message	Available if End Type is Message. <ul style="list-style-type: none">■ Drag an output message from Workspace Documents > <Solution> > <Project> tree on to the End event.

Monitoring tab

Configure Monitoring	Visible if End Type is Error, Message, or Rollback. Select to enable/disable monitoring for the End event activity. <ul style="list-style-type: none">■ Activity status: Records activity status, timestamp, user, and so on.■ Activity status, input and output messages: Records the input and output messages of the activity along with its status.■ Store complete activity information when activity aborts: Records activity status and input and output messages information only in case the activity process aborts. If the activity is successfully completed, no monitoring information is stored.
----------------------	--

Error details tab

Error Code	Visible if End Type is Error. Displays the error code and reads the value from the Message Map if the Read from message is selected. <ul style="list-style-type: none">■ Select the XPath from the Message Map. If this option is selected, the error code can be dynamically assigned at runtime. At runtime, the
------------	---

	<p>error code is read from the code that is defined in the Error Code field.</p> <ul style="list-style-type: none"> ■ If you select Static Value, enter your custom code for the error.
Error Message	<p>The message of the error.</p> <ul style="list-style-type: none"> ■ If you select Static Value, enter your custom message for the error. ■ If you select Read from message, select the XPath from the Message Map. If this option is selected, the error message can be dynamically assigned at runtime. At runtime, the error message is read from the message that is defined in the Error Message field.
Error Details	<p>The description of the error.</p> <ul style="list-style-type: none"> ■ If you select Read from message, select the XPath from the Message Map. If this option is selected, at runtime, the error detail is read from the error details that are defined in the Error Detail field. ■ If you select Static Value, enter your custom details for the error.

Annotation tab

Annotation	Additional notes or comments on the End event, if any.
------------	--

Catch exception event properties interface

General tab

Description	A description of the Exception event.
Error Code	A list of the different types of error codes that are used for exception handling.
Custom Error Code	Specify the Error Code if Custom Error is selected.

Monitoring tab

Enable Monitoring	Enable or disable monitoring for the activity.
Monitor Level	Activity status: Records activity status, timestamp, user, etc.

Annotation tab

Annotation	Additional notes or comments on the Exception event, if any.
------------	--

For each construct properties interface

General tab

Description	A description of the For Each loop.
Font Size	Allows the user to decrease or increase the font size of the text within the activity. The default font size is 12.

Select Condition	<p>Define the XPath of the condition if the selection condition is known during design-time. The For Each group is executed (iterated) for each XML node in the Message Map that matches the XPath condition.</p> <p>The Select Condition can be of types:</p> <ul style="list-style-type: none"> ▪ Static or Read from Message ▪ Select the Static option to specify the select condition in the Value field. <p>Select Read from Message to read the select condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change.</p> <p>Build a select condition by clicking  at the end of this field. Click this icon to select a condition.</p> <p>Ensure that the dynamic condition, which is taken from the Message Map does not start with a slash.</p>
Iterator Name	<p>Defines a name for the XPath condition for selection condition. Thus, while creating assignments in the Message Map, specify the Iterator Name instead of the XPath condition. An element for the For Each loop is created In the Source column of the Message Map, where:</p> <ul style="list-style-type: none"> ▪ The Description of the For Each group construct forms the root element. ▪ The Iterator Name forms the child element. <p>The Iterator Name property is available for constructs within a For Each loop. If you do not assign an Iterator Name, the default value given is <code>Iterator_<ID of For Each></code>. For example, <code>Iterator_o_39</code>.</p>
Execute Condition	<p>Enables you to specify the actual condition or the message from where the condition should be picked up for execution.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ▪ None: This option is displayed by default. Select this option if you do not want to specify any execute condition. ▪ Static: Select Static option to specify the actual condition for execution. You can build a condition by clicking the XPath editor icon  appearing at the end of this field. ▪ Read from Message: Select Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. Click  to build a condition. ▪ Ensure that the Read from Message condition from the Message Map does not start with a slash.

	<ul style="list-style-type: none"> ■ If a message contains an XPath expression, then the XPath need not be in single quotes. <p>For example, if the message is <code>/mycondition</code> and it contains an expression as follows:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple/old/ Employees/City = "London",</pre> <p>the XPath will be as follows:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple /old/Employees/City = "London".</pre> <p>In a For Each construct, the Execute Condition option can be used instead of the Decision construct. You can view a small Decision icon on the left at the bottom of the construct to indicate that the activity includes a condition to be executed.</p>
Data	<p>Select the input data required for each iterator. You may select any one of the following:</p> <ul style="list-style-type: none"> ■ Read from Message: Upon selecting this option, the input data is extracted from the message map. ■ Read file URL from Message: Select this option and click  to dynamically provide a file path from the message map (i.e., XML from web URL or file location) to get the streaming support for the For Each construct. <p>Note: This supports only forward navigation across matched elements. For example, read the following sample code.</p> <pre><bookstore> <book> <title lang="eng">Harry Potter</title> <price>29.99</price> </book> <book> <title lang="eng">Learning XML</title> <price>39.95</price> </book> </bookstore></pre> <p>In this case, forward navigation such as <code>/bookstore/book/title</code> (complete XPath) is supported but backward navigation such as <code>ancestor::book</code> or <code>descendant::book</code> is not supported.</p> <p>See Handling large XML documents in AppWorks Platform - Streaming XML for more information from a requirement and application standpoint.</p>

Monitoring tab

Configure Monitoring	Select this option to enable/disable monitoring for the activity.
Monitor Level	<ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, user. When

	<p>you set the monitor level at this parameter, only the activity status is monitored. This helps in getting the updated activity status and to spot errors should the activity abort.</p> <ul style="list-style-type: none"> ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. When you set the monitor level at this parameter, the activity status and the input and output messages of the message map is monitored. This helps in tracking and spotting any error occurring at this level. ■ Store complete activity information if activity aborts: Records activity status, input, and output messages only if the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.
--	--

Recovery tab

Store recovery data (A point of recovery from where the process will restart if it fails in middle of execution.)	Enable or disable recovery data for the activity. When you enable store recovery data at the activity level, the data is recovered only for the current activity. For more information on crash recovery, see Crash Recovery. If the recovery option is not enabled, lost data cannot be recovered when system and other failures occur or when a virus infects the system.
Recovery option	<p>The Recovery Option field is visible when crash recovery is enabled for the activity.</p> <ul style="list-style-type: none"> ■ At every iteration: The recovery data point is set at every iteration. This means that at every iteration of the current activity, data is stored for recovery. ■ At the end of all iterations: The recovery data point is set at the end of all iterations. This means that data is stored for recovery only after all the iterations of the current activity are completed.

Annotation tab

Annotation	Additional notes or comments on For Each construct, if any.
------------	---

Handling large XML documents in AppWorks Platform - Streaming XML content

Certain business scenarios involve processing large volumes of data as part of executing a business process instance. In such cases, instead of processing all data in a single, performance-intensive step, it is better to handle data, represented as an XML file, as a steady and continuous stream. This translates into a need to provide control over the manner in which data is processed by the Process Engine.

By this approach, you will have the following benefits.

Salary processing of railway employees

Let us assume a scenario of salary processing of employees in a government organization like railways. They perform this in a batch process. They will have the employee information in a legacy system and for salary processing, they will query the legacy system and create a batch file. These batch files are huge in size (In GB). In the business process model, for every employee, you have to get his/her work details, process the salary, and generate the pay slip.

Currently, you cannot stream the XML data from a file into the Business Process Management Service container. So, you have to load the file completely into the Business Process Management Service container memory which will eventually cause a out of memory problem. In order to avoid this, as a developer, you have to convert the employee records in the XML batch file into records in a table, generate the query method, and use these methods in the business process model. Developing, packing, and deploying batch processing models is a tedious and time-consuming approach.

Streaming support

1. Streaming support in XPath: XPath now provides a streaming API. Using this API, you can navigate through a collection of records without loading the entire XML into memory.
2. Streaming support in the For Each construct
 - In the For Each BPMN construct, you need to provide an XPATH expression that returns a collection of XML Nodes (Records) on which you iterate. This For Each construct is now extended to support streaming XPATH Expression.
 - The For Each construct works on the Normal mode and Streaming XPATH Mode.
 - When a streaming XPath mode is selected, you are required to provide the file URL via a message in message map to the For Each construct.
 - During the runtime, the engine resolves the URL from the message, creates an XPathReader with the URL, and streams the records from the file one by one for each iteration.
 - By default, after each interaction, you can delete the streamed record. However, if you want to maintain the record, you are provided with an option to do that.
 - In case of crash recovery, you can start streaming from the place you were when the crash happened.

Filter process instances interface

Pre-defined Filters	<p>When there are a huge number of process instances, you might want to filter them based on certain criteria for monitoring purpose. Select a filter from the list of predefined filters.</p> <ol style="list-style-type: none">1. Click  associated with the Choose Filter field. The Select Process Filter dialog box opens displaying a list of available filter criteria.2. Select a filter criteria and click OK. Alternatively, select User Default
---------------------	---

	<p>to set the default filter criteria.</p>
Select by Action in Progress	<p>Refers to the date and time of last action performed in a processes instance. You can use this option to display the process instances in which an action occurred with reference to a specific date and time.</p> <p>Action Date Time:</p> <ol style="list-style-type: none"> From the Selection Type list, select an option to set the selection criteria. The available options are Equal to, Between, Greater or Equal to, Greater than, Less or Equal to, Less than, and Like. Click  to display the Calendar control and select a date. Specify the time in the hh:mm format. <p>Action: Refers to the action status of the process instances.</p> <ul style="list-style-type: none"> Select a check box to select process instances by their action status. The options available are: <ul style="list-style-type: none"> All: Displays all process instances. Restart: Displays the restarted process instances. Suspend: Displays the process instances in which an activity was suspended. Reset: Displays the process instances that were reset. Edit MessageMap: Displays the process instances for which the message map was edited. Terminate: Displays the process instances in which an activity was terminated. Resume: Displays the process instances in which an activity was resumed. Skip: Displays the skipped process instances.
Select by Business Identifier	<p>You can also filter process instances based upon the business identifiers. When you select a business process, its associated business identifiers with column Show in PIM selected in business process properties in design time, are added to the business identifier selection table by default. The options available for filtering are:</p> <ul style="list-style-type: none"> Name: Displays name of the business identifier as specified during design time. Description: Displays description of the business identifier as specified during design time. Data Type: Displays the data type of the business identifier as specified during design time. Filter: Displays whether the filter is defined on the business identifier or not.

	<ul style="list-style-type: none">■ Selection Type: Displays the specified selection type. Value(s): Provide the specific value(s) by editing the entry. You can perform the following actions:<ul style="list-style-type: none">• To add a business identifier in the selection table, click . The Select a Business Identifier dialog box appears with existing list of business identifiers.• To edit a business identifier in the selection table, click . The Edit the Business Identifier dialog box opens.<ul style="list-style-type: none">• Select the Filter and enter the value and click OK.
Select by Business Process Name	<p>Note:</p> <ul style="list-style-type: none">■ Only four business identifiers will be considered for filtering the process instances.■ The sequence in which the columns are displayed in the Filter Process Instances screen, can be ordered using  and  buttons.■ If the business identifiers associated to business process model are modified, then only those business identifiers associated with the latest published or deployed business process model are shown in the Select by Business Identifier table. <p>When you want to filter process instances, you need to indicate the business process whose instances you want to filter. Use this field to filter process instances by selecting the business process to which they belong.</p> <ul style="list-style-type: none">■ Click  associated with the Business Process field. The Select a Process dialog box opens displaying all available business process models.■ Select the required business process model for which you want to view the process instances and click OK.■ To remove the attached process model, click .
Select by Instance Properties	If you want to filter process instances based on its specific properties, use Select by Instance Properties. However, not specifying process instances properties as filter criteria does not affect filtering. <ul style="list-style-type: none">■ Instance Type: Select the required criteria from the list and type the source from where the business process was triggered.■ Started by User: Select required criteria from the list and click  to select the user who instantiated the process, from the list that appears.■ Start Date Time: The start date and time. This is used to display processes that have been instantiated until this date. Select required criteria from the list and click  to select a date. Specify the time in the text box associated with this field.

-
- | | |
|--|--|
| | <ul style="list-style-type: none"> ■ Duration: Select the required criteria from the list and type the duration of the process. Specify the duration in days, hours, and minutes format. ■ Instance ID: The ID representing the process instance. You cannot use a combination of other selection criteria along with the instance ID. This is because, at any point of time, only a single matching record is displayed as a result of the search. ■ Process Status: To set the filter criteria based on the status of the process instance select the relevant option. |
|--|--|
-

Receive message properties interface

Annotation tab

Annotation	Additional notes or comments on the Intermediate Message event, if any.
------------	---

General tab

Description	A description of the Receive Message event.
-------------	---

Monitoring tab

Enable Monitoring	Enable or disable monitoring for the activity.
Input Message	<p>The input message should contain the name of the message and the message name prefix, where:</p> <ul style="list-style-type: none"> ■ name of the message refers to root tag of the message XML data which is send to the process instance by method <code>executeProcess</code> or from a sub-process instance. ■ message name prefix refers to the namespace displayed in the business process properties (tab namespaces). When sending the XML data to the process instance, the namespace of the root XML tag (message name) must have the same namespace, as defined in the business process, replicated under Process Specific Messages of the Message Map. Drag any message from the Workspace Documents > Solution > Project content tree on to the Receive Message event.
Message Mode	<p>Select a message mode to indicate whether Receive Event is Persistent or Transient.</p> <ul style="list-style-type: none"> ■ Persistent - Consume message even before Receive Message event is active: Select this option to ensure that all messages received while the Receive Message event is inactive are kept in queue. When the Receive Message event becomes active, it consumes the messages in queue and continues to execute the

	<p>process.</p> <ul style="list-style-type: none"> ■ Transient - Consume message only when Receive Message event is active: Select this option to ensure that the Receive Message event consumes only the messages received while it is active. Messages received before the Receive Message event is active are ignored. Persistent - Consume message even before Receive Message event is active: Select this option to ensure that all messages received while the Receive Message event is inactive are kept in queue. When the Receive Message event becomes active, it consumes the messages in queue and continues to execute the process. Transient - Consume message only when Receive Message event is active: Select this option to ensure that the Receive Message event consumes only the messages received while it is active. Messages received before the Receive Message event is active are ignored.
Monitor Level	<p>The status of the Receive Message event (while it is active) is 'waiting'.</p> <ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, and user. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. ■ Store complete activity information when activity aborts: Records activity status and input and output messages information only in case the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

Web service interface

The following table describes the fields on the Untitled Web Service - Web Service interface.

Select the source	The source for generating a Web service. When the source is selected, the Namespace field appears. Select Business Process Model as the source.
Name	Provide a name for the Web service that you want to create.
Description	A description of the Web service being generated. Provide important information about the Web service that you might need later.
Namespace	Provide a unique namespace for the Web service being generated.
Web Service Interface Name	Name of the selected business process on which the Web service is being generated. By default, the name of the Web service Interface remains the same as the name of the business process that was selected.

Web Service Operation Name	Provide a name of the Web Service Operation.
Activity Name	<p>Name of the activity in the business process model.</p> <p>Read-only field that, based on the business process model, displays the name of the activity that contains the message, which is used as input for the Web service. The default value is Start and the other possible value is ReceiveMessage.</p>
Monitoring	<p>A monitoring criterion helps you to track your business process and quickly respond to any anomalies.</p> <p>To specify monitoring, select monitoring criteria from the Process Monitoring list. Select any one of the following required monitoring criteria:</p> <ul style="list-style-type: none"> ■ As defined in the business process: Uses the default setting that is defined during modeling the business process. Note: If the setting has not been used while modeling, then the Enable Admin setting specified in the Business Process Management Service Container properties will be used; so if the Enable Admin option is selected, then the business process will be monitored. ■ On: Monitor the process. This setting will override what has been set during the modeling stage, or in the Business Process Management Service Container properties. ■ Off: Prevent the process from being monitored. This action will override the settings done during the modeling stage, or in the Business Process Management Service Container properties.
Execution Priority	<p>An execution priority for the business process helps you to execute important processes necessary for your business first and execute lesser priority ones later.</p> <p>Select required priority for a process. Processes with higher priority have precedence, during execution, over those having lower priority. If you select As defined in the business process, then the Web service uses the priority that has been set while modeling the process. However, if a priority is not set, the default priority Normal is assigned. Select any one of the following priorities to execute the business process:</p> <ul style="list-style-type: none"> ■ As defined in the business process ■ High ■ Above Normal ■ Normal ■ Below Normal ■ Low

New policy interface

The new archive policy dialog box helps in defining a new archive policy.

Fields other than on General, Schedule, and Time Line tabs

Description	The description of the archive policy.
Policy Name	The name of the archive policy

General tab

Archive process instance hierarchy	To archive all related instances of a process including instances of its parent process and sub-processes, if any, select the Archive process instance hierarchy check box. If you do not select this check box, only the process instances of the selected process is archived by default and instances of its parent process or sub-processes, if any, are not archived. The Archive process instance hierarchy functionality is applicable only for process instances that are created from Cordys C3 Release with Fix Pack 003.
Process	The name of the process .Allows you to either type a process name or select single or multiple processes from the artifact viewer that appears when you click  .
Status	Select the status of process instances based on which you want to archive the process instances. Choose any of the following options: <ul style="list-style-type: none">■ Select All Statuses: To archive all the completed, terminated, aborted and suspended instances. The Completed, Terminated, Aborted, and Suspended options are selected automatically and these are disabled.■ Specific Status: To archive instances of completed or terminated or aborted or suspended instances, or a combination of these specific statuses. Select the appropriate check boxes (Completed, Terminated, Aborted, or Suspended) in order to select the applicable instances.
Type	Type indicates a custom process instantiation type or source. It can be used by applications to specify their identity for better process monitoring and data filtering. It is important while specifying the type that the type is the same as the value mentioned in the Execute Process (part of Method Set Process Execution 4.2) request parameter source and does not differ from it. For example, Run from Designer, and Customer name.
User	The user whose transactions you want to archive. Type the user name or select single or multiple users from the artifact viewer that appears when you click  .

Schedule tab

At	Type the time at which you want the archive policy to run (in hh:mm format).
On Day	Select the day on which you want the archive policy to run. If you select Weekly as the Type, select the day of the week. If you select Monthly as the Type, select the day of the month. If you select Duration as the Type, specify the interval (number of days) in which the archiving should take place.
Policy Type	Select the type of archiving policy: Manual or Scheduled. Manual keeps the archive policy activated all the time. This option is selected by default. If you select Scheduled option, then Type, On Day, At, and Set fields are enabled.
Set	Click Set to set the schedule.
Type	Select the frequency of archiving: weekly, monthly, or for a specified duration.

Time Line tab

Time Line	<p>This field appear on selecting Manual option from Policy Type list and contains chronological information about the transactions you want to archive. Specify archival of transactions for a certain number of days or within a specific date range.</p> <ul style="list-style-type: none"> ■ Older than: Provide the number of days that have passed since the execution of process instances has ended. ■ From: Select this option to select the start date from which the transactions are to be archived. ■ To: Select this option, to select the end date by which time transactions are to be archived.
-----------	--

Saving document interface

Save Document dialog box

Name	<p>The name of the document. The name must contain values satisfying the following conditions:</p> <ul style="list-style-type: none"> ■ Begins with a letter or an underscore (_) ■ Contains a letter, number, period (.), or underscore (_) ■ Contains at least one letter or number if the code begins with an underscore <p>Avoid white spaces in the name.</p>
Description	The description of the document. Contains only letters, numbers, and special characters. The special characters permitted are periods (.),

	commas (,), parentheses (), single quotation marks ('), hyphens  , and underscores (_).
Save in Folder	Select a folder or project to save your document.

Print preview interface

Print	Prints the business model diagram	
Save	Saves the business model diagram on the client machine either in .jpg or .bmp format	
Fit to Page	Fits the image to A4 paper size	

Send message event properties interface

General tab

Description	A description of the Send Message event. Provide a meaningful description for the Send Message. While typing the description, auto suggest help appears based on existing description of labels and fields. The event description is translated to a preferred language if user configures the language settings. View the description for the activities and models in the PIM activity table and graphical view in the preferred language selected in user preferences.
Execute Condition	The Execute Condition can be of types - Static or Read from Message In a Send Message event construct, the Execute Condition option can be used instead of the Decision construct. The Decision icon appears on the left at the bottom of the construct to indicate that the event includes a condition to be executed. Select the Static option to specify the actual condition for execution in the Value field. Select the Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. You can build a condition by clicking the XPath editor icon () appearing at the end of this field. <ul style="list-style-type: none"> ■ Ensure that the Read from Message condition that is taken from the Message Map does not start with a slash. ■ If a message contains an XPath expression, then the XPath need not be in single quotes.
Target	In this field, you can specify the process to which you want to send a message as part of inter-process communication. Select any one of the following options from the Process Model list:

	<ul style="list-style-type: none"> ■ Parent Process: By default, if you do not select a process, the send message event of the current process will send a message to its parent process. ■ Specific Process: Select this option, if you want the current process to communicate with a specific process. Click  and select required business process model from the Select Business Process Model window. ■ Any Process: Select this option, if you want the current process to communicate with any process.
Instance	<p>Displays the selected process instance ID of the selected business process model.</p> <p>The parameters for this field vary based on the Process Model selected in the previous field.</p> <ul style="list-style-type: none"> ■ If you selected Parent Process as the Process Model parameter, the instance ID is displayed in a non-editable format. ■ If you selected Specific Process, then you need to provide one of the following: <ul style="list-style-type: none"> • Read from Message - Click and provide the required process Instance ID in the XPath editor. • Select by Business Identifiers - Click and select the required process instance ID from the Filter by Business Identifiers dialog box that appears. For more information on adding business identifiers, refer to Business Identifiers. ■ If you selected Any Process as the Process Model parameter, the Read from Message field is disabled and you can click and provide the required process Instance ID in the XPath editor.
Message	<p>Displays the message to be sent to the main process. When you have a send message event in the business process, you need to associate a message to that event. If there is no message specified for the send message event, then the business process will not execute from that send message event onwards.</p> <p>You can drag a message from the Message Map, or a message of any application service to the Send Message activities.</p> <ul style="list-style-type: none"> ■ Click  and from the Select a Message window, select required message. <p>The Select a Message window displays all process specific messages, Web service message and Independent Sub-process messages that are available for the current business process model.</p>

Monitoring tab

Enable Monitoring	Specify the monitor settings for the current event. Enable or disable monitoring for the event.
Monitor Level	Select required monitor level. <ul style="list-style-type: none"> ▪ Activity status: Records activity status, timestamp, user. ▪ Activity status, input and output messages: Records the input and output messages of the activity along with its status. ▪ Store complete activity information when activity aborts: Records activity status and input and output messages information only in case the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.
Store Recovery Data	Specify the store recovery settings for the current activity. Enable or disable recovery information for the activity.

Annotation tab

Annotation	Additional notes or comments on the Send Message event. Provide additional information which you might want to reuse later.
------------	--

Start event properties interface

General tab

Description	A description of the Start event.
Trigger Type	Based on the trigger type, the process flow is instantiated. The following are the trigger types. <ul style="list-style-type: none"> ▪ Message: The process starts with a defined message. ▪ Timer: The process starts from the scheduler. ▪ No Message or Timer: The process starts when it is triggered.
Input Message	This field is visible only when the activity is that of a Receive Message activity or when the Trigger Type is Message for Start. When you have a receive message activity in the business process, you need to associate an input message to that activity. If there is no message specified for the receive message activity, then the business process will not be executed from that receive message activity onwards. You can also Add Document or Add Runtime Reference from the Select a Message window. Drag the required <input message name> from Workspace Documents > Solution > Project > <business process model>.

Monitoring tab

This tab appears only when the Trigger Type is Message.

Configure Monitoring > Monitor Level	<p>Enable/disable monitoring for the Start event activity.</p> <ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, and user information. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status.
--------------------------------------	---

Annotation tab

Annotation	Additional notes or comments on the Start event, if any.
------------	--

Independent subprocess properties interface

Description	A description of the sub-process.
Font Size	Allows the user to decrease or increase the font size of the text within the activity. The default font size is 12.
Sub Process	<ul style="list-style-type: none"> ■ Select Static Value or Read From Message value to the independent sub-process. The Independent Sub-process to be executed is read from the message that is defined in the Read From Message field. ■ When you select Read From Message, set the XPath of the message containing the sub-process name. <p>If you selected the Read From Message option, ensure to provide the exact path of the sub-process. If you typed the name of the sub-process or do not provide the exact path, the sub-process will not be invoked if it is created within a folder in the CWS project. Only the sub-processes that are not created within a folder will be invoked in such scenarios. Alternatively, use the Static Value option and select the appropriate process name from the list that is displayed.</p>
Instance ID	The process model is instantiated with the defined identifier only if this field includes the defined identifier in the XPath of the message. Also, ensure that the defined identifier is unique.
Instantiation Source	<ul style="list-style-type: none"> ■ Select Same as Main Process to display the instantiation source for the linked business process ■ Or select Other to indicate that the instantiation source for the linked business process is not the same as its main process.
Execution Priority	<p>Select one of the following options to set an execution priority for the linked business process:</p> <ul style="list-style-type: none"> ■ Same as Main Process - Priority is same as defined in the main process.

	<ul style="list-style-type: none">■ Same as Sub Process - Priority is same as defined in the linked business process.■ Other - To specify the execution priority that is different to the main process and the subprocess, select one of the following:<ul style="list-style-type: none">• Static - You can select any one of the priority levels: Highest, High, Normal, Low, and Lowest.• Read from message - If you want to dynamically set the execution priority.
Wait until Sub Process is Finished	<p>If the Read from Message is selected and an invalid value is provided, then the priority of the parent process will be considered; value will be considered invalid, if it is either empty, non-numeric, or contains a value less than 1 or greater than 5.</p> <p>If you select this check box, the main process will wait until the sub-process sends the expected output message or indicates that the next activity in the main process can start processing. If you clear this check box, the main process will start the sub-process and continue with the next activity. The main process can synchronize with the sub-process by receiving a message through the Receive Message event. In the sub-process, a Send Message Activity or end Event can send this message or communication to the main process.</p> <p>This option is disabled if the sub-process is a short-lived process.</p>
Execute Condition	<p>The Execute Condition can be of types - None, Static Value or Read From Message.</p> <ul style="list-style-type: none">■ Select Static Value to specify the actual condition for execution.■ Select Read From Message to read the condition from the XPath of the message. You can build a condition by clicking the XPath editor icon ( appearing at the end of this field. <p>Ensure that the Read From Message condition which is taken from the Message Map does not start with a slash.</p> <p>If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is /mycondition and it contains an expression as follows:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple/old /Employees/City = "London"</pre> <p>If the specified execute condition is satisfied, the activity is executed otherwise, and it is skipped. In a sub-process construct, the Execute Condition option can be used instead of the Decision construct. Thus, you do not need to explicitly model a decision construct when only one path from the Decision construct leads to an activity. You can view the decision icon on the left bottom corner of the construct indicating that the activity includes a condition for execution.</p>

Execution User Type	<ul style="list-style-type: none"> ▪ Select Current User or Process Instantiation User for the selected activity. <p>For each instance of a process, there can be either a Current User or a Process Instantiation User. The Process Instantiation User can trigger the process while the Current User is the one who is currently logged on.</p>
---------------------	--

Swimlane properties interface

General tab

Description	Description of the swimlane construct.
Color	Select the color of the swimlane. Each swimlane can have a different color with a combination of the shade for the header and the lane.
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12.

Assignee tab

Assignee Type	<p>Select a Role and click  to select required property for the selected assignee.</p> <p>For Team (now known as Organizational Unit), type the name of the Organizational Unit that was created by the administrator using the Identity dashboards.</p> <p>For information about Identity dashboards, see <i>the AppWorks Platform Low-Code Design Guide</i>.</p>
---------------	---

Annotation tab

Annotation	Additional notes or comments on the swimlane, if any.
------------	---

Annotation configuration interface

Description	A description of the text annotation.
Font Size	<p>Denotes the font size of the annotation. Decrease or increase the font size of the text by selecting values from this list.</p> <p>The default font size is 12.</p>

Time-out properties interface

General tab

Description	A description of the Activity.
Use Business Calendar	<ul style="list-style-type: none"> ▪ Specific to Activity: Select a particular activity that appears in the Business Process dialog box for using the business calendar.

-
- | | |
|--|---|
| | <ul style="list-style-type: none"> ■ Same as that of Process: Use the business calendar for the entire business process. ■ Read from Message: Specify the XPath for reading the business calendar dynamically from the Message Map. |
|--|---|
-

Duration tab

Duration	<p>Specify the duration for a Time-Out that is related to the business days/hours/minutes. When you select the Use Business Calendar option, the business days are calculated based upon the business days specified in the business calendar. Otherwise, the business days are calculated including holidays and non-working hours.</p> <p>The following options appear when the Use Business Calendar option is not selected:</p> <ul style="list-style-type: none"> ■ Static Duration: Enter Days, select Hours, Minutes and Seconds from the respective lists. ■ Read Duration from Message: Click  and from the XPath Editor, select required message, validate, test and click OK. The input for this field should be in XML Schema Duration Data Type. ■ Duration Data Type: The duration data type is used to specify a time interval. The time interval is specified in the following form "PnYnMnDTnHnMnS" where: <ul style="list-style-type: none"> • P indicates the period (required) • nY indicates the number of years • nM indicates the number of months • nD indicates the number of days • T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds) • nH indicates the number of hours • nM indicates the number of minutes • nS indicates the number of seconds For example: P0DT1H0M0S indicates a duration of 1 hour. <p>The following options appear in the Duration list only when you select Use Business Calendar:</p> <ul style="list-style-type: none"> ■ Static Value in Business Days: Enter the business days as a fixed value for the number of days in the Days field. ■ Read Business Days from Message: Click  and from the XPath Editor, select required message, validate, test, and click OK. ■ Static Value in Business Hours / Minutes: Select Hours and
----------	--

	<p>Minutes from the lists.</p> <ul style="list-style-type: none"> ■ Read Business Hours / Minutes from Message: Click  and from the XPath Editor, select required message, validate, test, and click OK.
Include Start Day	Select this option if the start day needs to be included to determine the end of calculation of the due date.
Calculation ends on	<ul style="list-style-type: none"> ■ Start of Next Business Day: Sets the calculation to end on the start of next business day. ■ End of Business Day: Sets the calculation to end on the end of next business day.

Monitoring tab

Configure Monitoring	Enable/disable monitoring for the activity.
Monitor Level	<p>Activity Status: Non-editable field. It records activity status, timestamp, and user.</p>

Annotation tab

Annotation	Additional notes or comments on the Time-out, if any.
------------	---

Transaction construct properties interface

General tab

Description	A description of the Transaction.
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12.
Execute Condition	<p>Enables you to specify the actual condition or the message from where the condition should be picked up for execution.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ None: This option is displayed by default. Select this option if you do not want to specify any execute condition. ■ Static: Select Static option to specify the actual condition for execution. You can build a condition by clicking the XPath editor icon () appearing at the end of this field. ■ Read from Message: Select Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. Click to build a condition.

	<p>Ensure that the Read from Message condition from the Message Map does not start with a slash.</p> <p>If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is <code>/mycondition</code> and it contains an expression as follows:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple/old/ Employees/City = "London", the XPath will be as follows: GetEmployeeOutput/GetEmployeeResponse/tuple/old/ Employees/City = "London".</pre> <p>In a Transaction construct, the Execute Condition option can be used instead of the Decision construct. You can view a small Decision icon on the left at the bottom of the construct to indicate that the activity includes a condition to be executed.</p>
--	---

Monitoring tab

Enable Monitoring	Enable/disable monitoring for the transaction activity.
Monitor Level	<ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, user, etc. When you set the monitor level at this parameter, only the activity status is monitored. This helps in getting the updated activity status and to spot errors should the activity abort. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. When you set the monitor level at this parameter, the activity status, and the input and output messages of the message map is monitored. This helps in tracking and spotting any error occurring at this level. ■ Store complete activity information if activity aborts: Records activity status, input, and output messages information only if the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

Data Source tab

Read Data Source from	Default setting for a transaction is Service Group - Find one automatically . The business process goes to the first application or WS-AppServer activity, picks up the namespace and retrieves the corresponding Service Group details or Service Group 'dn'. The Service Group Data Source Object is then used by the Transaction. If a transaction includes a sub-process consisting of application or WS-AppServer web services, the namespace of the web service determines the Service Group. If there are no application or WS-AppServer web services, then a high priority warning is displayed.
-----------------------	---

Annotation tab

Annotation	Additional notes or comments on the Transaction construct, if any.
------------	--

Until construct properties interface**General tab**

Description	A description of the Until loop.
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12.
Select Condition	<p>Define the XPath of the condition if the condition is known during design-time. The Until group is executed (iterated) for each XML node in the Message Map that matches the XPath condition.</p> <p>The Select Condition can be of types Static Value or Read From Message.</p> <ul style="list-style-type: none"> ▪ Select the Static Value option to specify the select condition in the Value field. ▪ Select the Read From Message option to read the select condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. ▪ You can build a select condition by clicking the XPath editor icon () appearing at the end of this field. ▪ Click this icon to select a condition.
Execute Condition	<p>Enables you to specify the actual condition or the message from where the condition should be picked up for execution.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ▪ None: This option is displayed by default. Select this option if you do not want to specify any execute condition. ▪ Static: Select Static option to specify the actual condition for execution. You can build a condition by clicking the XPath editor icon () appearing at the end of this field. ▪ Read from Message: Select Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. Click  to build a condition. <p>Ensure that the Read From Message condition which is taken from the Message Map does not start with a slash.</p> <p>If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is /mycondition and it contains an expression as follows: GetEmployeeOutput/GetEmployeeResponse/tuple/ old/Employees/City = "London", the XPath will be as follows:</p>

	GetEmployeeOutput/GetEmployeeResponse/tuple/ old/Employees/City = "London". In an Until construct, the Execute Condition option can be used instead of the Decision construct. You can view a small Decision icon on the left at the bottom of the construct to indicate that the activity includes a condition to be executed.
--	--

Monitoring tab

Enable Monitoring	Enable/disable monitoring for the activity.
Monitor Level	<ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, user. When you set the monitor level at this parameter, only the activity status is monitored. This helps in getting the updated activity status and to spot errors should the activity abort. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. When you set the monitor level at this parameter, the activity status, and the input and output messages of the message map is monitored. This helps in tracking and spotting any error occurring at this level. ■ Store complete activity information if activity aborts: Records activity status, input, and output messages information only if the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

Recovery tab

Store recovery data (A point of recovery from where the process will restart if it fails in middle of execution.)	Enable/disable recovery data for the activity. When you enable store recovery data at the activity level, the data is recovered only for the current activity. For more information on crash recovery, see Crash Recovery. If the recovery option is not enabled, lost data cannot be recovered when system and other failures occur or when a virus infects the system.
Recovery option	<p>The Recovery Option field is visible when crash recovery is enabled for the activity.</p> <ul style="list-style-type: none"> ■ At every iteration: The recovery data point is set at every iteration. This means that at every iteration of the current activity, data is stored for recovery. ■ At the end of all iterations: The recovery data point is set at the end of all iterations. This means that data is stored for recovery only after all the iterations of the current activity are completed. At every iteration: The recovery data point is set at every iteration. This means that at every iteration of the current activity, data is stored for recovery. At the end of all iterations: The recovery data point is set at the end of all

	iterations. This means that data is stored for recovery only after all the iterations of the current activity are completed.
--	--

Annotation tab

Annotation	Additional notes or comments on Until construct, if any.
------------	--

While construct properties

General tab

Description	A description of the While loop.
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12.
Select Condition	<p>Define the XPath of the condition if the condition is known during design-time. The While group is executed (iterated) for each XML node in the Message Map that matches the XPath condition.</p> <p>The Select Condition can be of types - Static or Read from Message.</p> <ul style="list-style-type: none"> ■ Select the Static option to specify the select condition in the Value field. ■ Select the Read from Message option to read the select condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. ■ You can build a select condition by clicking the XPath editor icon () appearing at the end of this field. ■ Click this icon to select a condition.
Execute Condition	<p>Specify the actual condition or the message from where the condition should be picked up for execution.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ None: This option is displayed by default. Select this option if you do not want to specify any execute condition. ■ Static: Select Static option to specify the actual condition for execution. You can build a condition by clicking the XPath editor icon () appearing at the end of this field. ■ Read from Message: Select Read from Message option to read the condition from the Process Specific Messages in the Message Map. This ensures that the business process does not have to be modified for every change. Click  to build a condition. ■ Ensure that the Read from Message condition which is taken from the Message Map does not start with a slash.

- If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is /mycondition and it contains an expression as follows:
GetEmployeeOutput/GetEmployeeResponse/tuple/
old/Employees/City = "London", the XPath will be as follows:
GetEmployeeOutput/GetEmployeeResponse/tuple/
old/Employees/City = "London". In a While construct, the Execute Condition option can be used instead of the Decision construct. You can view a small Decision icon on the left at the bottom of the construct to indicate that the activity includes a condition to be executed.

Monitoring tab

Enable Monitoring	Enable or disable monitoring for the activity. When you enable monitoring at the activity level, the monitoring is done for the current activity.
Monitor Level	<ul style="list-style-type: none"> ■ Activity status: Records activity status, timestamp, user, etc. When you set the monitor level at this parameter, only the activity status is monitored. This helps in getting the updated activity status and to spot errors should the activity abort. ■ Activity status, input and output messages: Records the input and output messages of the activity along with its status. When you set the monitor level at this parameter, the activity status, and the input and output messages of the message map is monitored. This helps in tracking and spotting any error occurring at this level. ■ Store complete activity information if activity aborts: Records activity status and input and output messages information only if the activity process aborts. If the activity is successfully completed, none of the monitoring information is recorded.

Recovery tab

Store recovery data (A point of recovery from where the process will restart if it fails in middle of execution.)	Enable/disable recovery data for the activity. When you enable store recovery data at the activity level, the data is recovered only for the current activity. If the recovery option is not enabled, lost data cannot be recovered when system and other failures occur or when a virus infects the system.
Recovery option	<p>The Recovery Option field is visible when crash recovery is enabled for the activity.</p> <ul style="list-style-type: none"> ■ At every iteration: The recovery data point is set at every iteration. This means that at every iteration of the current activity, data is stored for recovery. ■ At the end of all iterations: The recovery data point is set at the end of all iterations. This means that data is stored for

	recovery only after all the iterations of the current activity are completed.
--	---

Annotation tab

Annotation	Additional notes or comments on While construct, if any.
------------	--

Web service generation wizard interface

Activity Name	<p>Name of the activity in the business process model.</p> <p>Read-only field that, based on the business process model, displays the name of the activity that contains the message, which is used as input for the Web service. The default value is Start and the other possible value is ReceiveMessage.</p>
Execution Priority	<p>Specifying an execution priority for the business process helps you to execute important processes necessary for your business first and execute lesser priority ones later.</p> <p>Select required priority for a process. Processes with higher priority are given precedence, during execution, over those having lower priority.</p> <p>If you select As defined in the business process, then the Web service will use the priority that has been set while modeling the process. However, if a priority is not set, the default priority Normal is assigned. Select any one of the following priorities to execute the business process:</p> <ul style="list-style-type: none"> ■ As defined in the business process ■ High ■ Above Normal ■ Normal ■ Below Normal ■ Low
Monitoring	<p>Specifying a monitoring criterion helps you to track your business process and quickly respond to any anomalies.</p> <p>To specify whether you want monitoring to be enabled, select a monitoring criterion from the Process Monitoring list. Select any one of the following required monitoring criteria:</p> <ul style="list-style-type: none"> ■ As defined in the business process: Uses the default setting that is defined during modeling the business process. If the setting has not been used while modeling, then the Enable Admin setting specified in the Business Process Management Service Container properties will be used; so if Enable Admin is selected, then the business process will be

	<p>monitored.</p> <ul style="list-style-type: none">■ On: Monitor the process. This setting will override what has been set during the modeling stage, or in the Business Process Management Service Container properties.■ Off: Prevent the process from being monitored. This action will override the settings done during the modeling stage, or in the Business Process Management Service Container properties.
Select Webservice Definition Set	<p>The Web service definition set for generating the Web service. Select any one of the following options to select a Webservice definition set:</p> <ul style="list-style-type: none">■ New: Select this option if you want to create a new Webservice definition set. The Webservice Definition Set Name and Location fields appear. Selecting this option automatically creates a new Webservice definition set basing the business process that you selected to generate the Web service. The selected business process model name is appended with <code>_DefinitionSet</code> and appears as <code><businessprocessmodelname>_DefinitionSet</code> in the Webservice Definition Set Name field.■ Existing: Select this option if you want to use associate an existing Webservice Definition Set to the Web service. The Existing Webservice Definition Set Name field appears.
Select Webservice Definition Set > Existing > Description	<p>A description of the Web service that is being generated. Provide important information about the Web service that you might need later.</p>
Select Webservice Definition Set > Existing > Existing Webservice Definition Set Name	<p>Field from where to select an existing Webservice Definition Set. This field appears only when you select the Existing option. Click  and select an existing Webservice Definition Set from the required solution, project or folder and click OK.</p>
Select Webservice Definition Set > Existing > Namespace	<p>Provide a unique namespace for the Web service that is being generated.</p>
Select Webservice Definition Set > Existing > Name	<p>Provide a name for the Web service that you want to create.</p>
Select Webservice Definition Set > Existing > Web Service Interface Name	<p>Name of the selected business process basing which the Web service is being generated. By default, the name of the Webservice Interface appears the same as the name of the business process that was selected.</p>

Select Webservice Definition Set > New > Location	However, click  and select required Webservice interface from the project.
Select Webservice Definition Set > New > Webservice Definition Set Name	Location where the newly created Webservice Definition Set is to be saved. Click  and select required solution, project, or folder where you want to save the newly created Webservice Definition Set and click OK.
Webservice Operation Name	Name of the new Webservice Definition Set. Selecting New option automatically creates a new Webservice definition set basing the business process that you selected to generate the Web service. The selected business process model name is appended with <code>_DefinitionSet</code> and appears as <code><businessprocessmodelname>_DefinitionSet</code> in the Webservice Definition Set Name field.

Case activity properties interface

General tab

Description	A description of the subprocess.
Font Size	Decrease or increase the font size of the text within the activity. The default font size is 12.
Case Model	Select Static Value or Read From Message to the Case Activity. The Case Activity to be executed is read from the message that is defined in the Read From Message field. When you select Read From Message, set the XPath of the message containing the Case model name.
Instance ID	The Case model will be instantiated with the defined identifier only if this field contains the defined identifier in the XPath of the message. Also, ensure that the defined identifier is unique.
Execution Priority	Select one the following options to set an execution priority for the linked Case model: <ul style="list-style-type: none"> ■ Same as Main Process - Priority is same as defined in the main process. ■ Same as Sub Case - Priority is same as defined in the linked Case model. ■ Other - To specify the execution priority that is different to the main process and the linked Case model, select one of the

	<p>following:</p> <ul style="list-style-type: none"> • Static - You can select any one of the priority levels: Highest, High, Normal, Low, and Lowest. • Read from message - If you want to dynamically set the execution priority. <p>If Read from Message is selected and an invalid value is provided, then the priority of the parent case will be considered; value will be considered invalid, if it is either empty, non-numeric, or contains a value less than 1 or greater than 5.</p>
Wait until Case is Finished	<p>If you select this option, the current process will wait until the Case model is finished which indicates that the next activity in the current process can start processing. If you clear this option, the main process starts the Case model and continues with the next activity.</p>
Execute Condition	<p>The Execute Condition can be of types - None, Static Value, or Read From Message. Select the Static Value option to specify the actual condition for execution. Select the Read From Message option to read the condition from the XPath of the message. You can build a condition by clicking the XPath editor icon (🔍) appearing at the end of this field.</p> <p>Ensure that the Read From Message condition which is taken from the Message Map does not start with a slash.</p> <p>If a message contains an XPath expression, then the XPath need not be in single quotes. For example, if the message is /mycondition and it contains an expression as follows:</p> <pre>GetEmployeeOutput/GetEmployeeResponse/tuple/ old/Employees/City = "London"</pre> <p>If the specified execute condition is satisfied, the activity is executed, otherwise it is skipped. In a subprocess construct, the Execute Condition option can be used instead of the Decision construct. Thus, you do not need to explicitly model a decision construct when only one path from the Decision construct leads to an activity. You can view the decision icon on the left bottom corner of the construct indicating that the activity includes a condition for execution.</p>

Monitoring tab

Monitor Level	<p>Refers to the level of monitoring that you want to set on the Independent subprocess activity.</p> <ul style="list-style-type: none"> ▪ Activity status: Records activity status, timestamp, and user information. ▪ Activity status, input and output messages: Records the input and output messages of the activity along with its status.
---------------	--

	<ul style="list-style-type: none"> Store complete activity information when activity aborts: Records activity status and input and output messages information only when the process aborts. If the activity is successfully completed, none of the monitoring information is recorded. <p>Monitoring options at activity level are dependent upon the options defined at the process level in the Default Activity Monitoring tab. You can override the monitoring activity and store the relevant recovery point settings to each individual activity level, by clicking the Apply Default Settings button.</p>
--	---

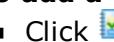
Annotation tab

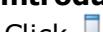
Annotation	Additional notes or comments on the independent subprocess, if any.
------------	---

Constructs used in a business process model

Business Process Modeling Notation (BPMN) is a standardized graphical notation for representing business processes in a workflow. BPMN specifies a standardized way to depict various activities and events that are used to model business processes in a graphical manner. The graphical objects defined by BPMN are known as BPMN constructs.

AppWorks Platform supports the following BPMN constructs.

Construct Name	Description
Start Event	<p>To initiate a process flow:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See Start Event for more information on this construct.
Activity	<p>To add an activity to the business process model:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See Activity for more information on this construct.
Independent Subprocess	<p>To add an independent sub-process:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See Independent subprocess for more information on this construct.
Case Model	<p>To add a case model to the business process model:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox.
Send Message	<p>Send Message Intermediate Event should be used to send a message from one process to the other.</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox to add this construct. See Send message event for more information on this

Construct Name	Description
	construct.
Compensate	<p>To roll back an operation that has been executed:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Compensate event for more information on this construct.</p>
Catch Exception	<p>To catch an exception into the process for error handling:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Catch exception for more information on this construct.</p>
Time-out	<p>To introduce a time-out into the process:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Time-out for more information on this construct.</p>
Decision	<p>To indicate a decision point in the process:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Decision for more information on this construct.</p>
Vertical Lane	<p>To introduce a vertical lane in the process:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Swimlane for more information on this construct.</p>
Horizontal Lane	<p>To introduce a horizontal lane in the process:</p> <ul style="list-style-type: none"> ▪ Click  on the Business Process Modeler toolbox. <p>See Swimlane for more information on this construct.</p>
Text	<p>Use Text to add an annotation, comprising a border and a background color, to the business process diagram. You can use the annotation to add and display information specific to the business process diagram.</p> <p>To set the properties of the annotation:</p> <ol style="list-style-type: none"> 1. Right-click it and select Properties to open the property sheet of the annotation. 2. Set the text to be displayed and the font size of the text.
Text Annotation	<p>Use Text Annotation to add an annotation, which can be linked to a construct. You can use the annotation to add and display information specific to the construct.</p> <p>The annotation comprises a border and a connector that links it to the construct.</p> <p>To set the properties of the annotation:</p> <ol style="list-style-type: none"> 1. Right-click it and select Properties to open the property sheet of the annotation.

Construct Name	Description
	2. Set the text to be displayed and the font size of the text.
Transparent Text	<p>Use to add a transparent annotation to the business process diagram. You can use the annotation to add and display information specific to the business process diagram. The annotation does not contain a border or background color.</p> <p>To set the properties of the annotation:</p> <ol style="list-style-type: none"> Right-click it and select Properties to open the property sheet of the annotation. Set the text to be displayed and the font size of the text.
Receive Message	<p>Receive Message Intermediate Event should be used to receive the message.</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox to add this construct. See Receive message for more information on this construct.
Delay	<p>To introduce a delay or pause in the process flow:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See Delay for more information on this construct.
Embedded Sub-process (While, Until, For Each, Transaction)	<p>To group a set of constructs for executing a set of activities or sub-processes repeatedly:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See Group constructs for more information.
End Event	<p>To end the process flow:</p> <ul style="list-style-type: none"> Click  on the Business Process Modeler toolbox. See End event for more information on this construct.

Start event

A Start event is a BPMN construct that indicates the start of a process, and is mandatory for a valid business process model. A Start event does not have any incoming connectors. A Start event can be instantiated through a variety of triggers, which are described in the following table.

Trigger type	Description	Graphical representation
Timer	When the timer is enabled, the business process model should be attached to a schedule.	

Trigger type	Description	Graphical representation
No message or timer	Start event without any trigger type is defined.	
Message	Start event is triggered by an incoming message.	

Activity

An activity is the smallest unit of a business process that involves an operation. A logical set of activities that are connected and performed in a sequential manner having a start and end point and focus on achieving a common goal form a business process model. While designing a business process model an activity is represented as a BPMN construct and is a generic term for work that is performed within a business process.

An activity is atomic when there is only a single activity to be performed, whereas an activity is non-atomic when there are several other activities that it is dependent upon and vice-versa. The different types of activities that are a part of a process model are simple activities, User Interface, Web services, decision tables, transaction, sub-process activity, and case activity. In AppWorks Platform, an activity is represented using the following graphic.



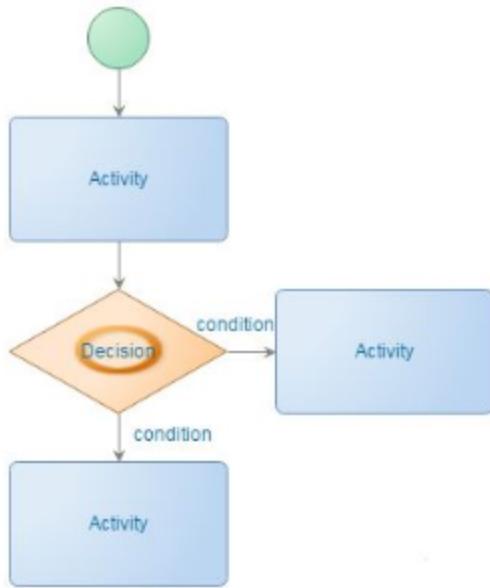
Activity	This type of activity does not have a method or any User Interface attached to it. This activity is typically used to perform message map assignments.
Decision Table	When a decision table is attached to an activity, the decision table is invoked from the business process model .
Entity Layout	Insert a Task Entity layout defined on the Entity. Use this in the context of a Lifecycle building block flow to open the task and the related task information in the Task Layout from the application task panels.
Web Service	When a Web service is attached to an activity, a SOAP request is sent to perform a certain task. No user action is required.
XForm or External User Interface	When an XForm or External User Interface is attached to an activity, the XForm or External User Interface is sent to the Inbox when the business process model is executed. The message can be an XForm, External User Interface or information for the user.

Decision

Decisions are BPMN constructs. They function as gateways in a business process, where the flow control can take one or more alternative paths. A decision has at least one incoming connector and two or more outgoing connectors. Decisions can be broadly categorized into data-based decisions and event-based decisions.

Data-based decisions

The data-based decision represents a branching point where alternatives are chosen based on conditional expressions contained within the outgoing connector. Only one of the alternatives is chosen.



A data-based decision can have multiple incoming and outgoing connectors depending on its type: inclusive, exclusive, and parallel. Connectors can be merged together on input and split apart on output.

Inclusive

In inclusive decisions, alternatives are selected based on conditional expressions contained within the outgoing connector. When multiple outgoing paths exist, conditions are evaluated independently, and all or a combination of paths are executed based on satisfied conditions. If no condition is satisfied, the default defined path is executed. If no default path is defined, a runtime exception occurs.

An inclusive decision is graphically represented as follows.



Inclusive decisions can be converging in nature with two or more incoming or parallel paths that merge (join) into one path.

Exclusive

In exclusive decisions, only one alternative is selected based on conditional expressions contained within the outgoing connector. When multiple outgoing paths exist, conditions are evaluated based on the order in which they are defined on the Conditions tab. If even one condition is satisfied, the remaining conditions are skipped, and the path with the satisfied condition is selected. If no condition is satisfied, the default defined path is executed. If no default path is defined, a runtime exception occurs.

An exclusive decision is graphically represented as follows.



Exclusive decisions can be converging in nature with two or more incoming paths that merge (join) into one path.

Parallel

This decision type is only available when designing business process models in AppWorks Platform 16.

In parallel decisions, an outgoing path is divided, at a certain point, into two or more paths that run in parallel within the process. This is termed as forking (splitting) where activities are executed simultaneously, rather than sequentially.

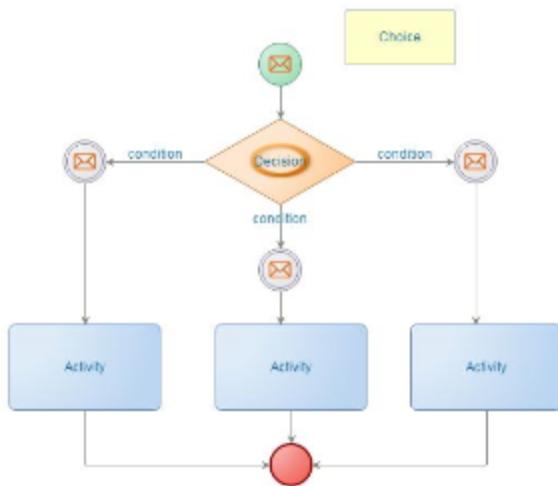
A parallel decision is graphically represented as follows.



Parallel decisions can be converging in nature with two or more alternative incoming parallel paths that merge (join) into one path. The parallel decision waits for all incoming paths before triggering the flow through its outgoing paths.

Event-based decisions

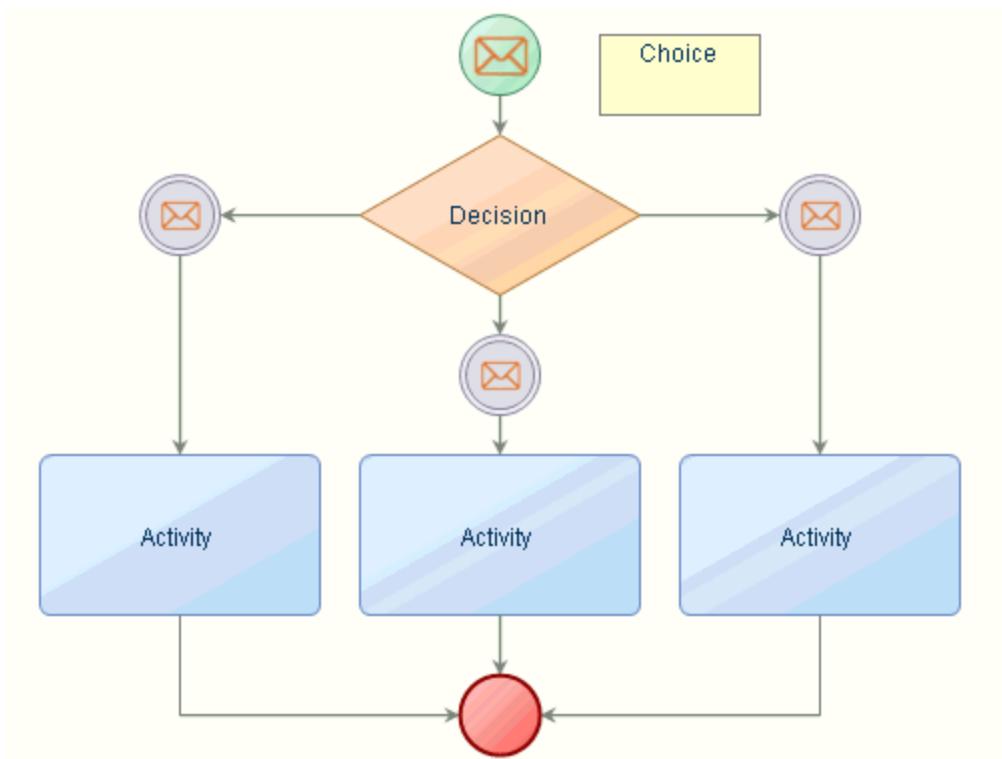
The event-based decision represents a branching point where alternatives are chosen based on an event that occurs at that point in the process. A specific event, usually the receipt of a message, determines the alternative that is chosen. Though events such as timer can also be used, AppWorks Platform uses Receive Message Event for event-based decisions.



Receive message

Receive Message event is a BPMN construct, which is triggered by an incoming message from a participant. The process remains in waiting state until it receives the specified message and once the message is received, the process flow will move to the next immediate activity. A Receive Message event can also be used to synchronize a sub-process with its parent process, in which case, the Receive Message is used in the main process. The Receive Message is graphically represented as

If a Receive Message is modeled after a Decision construct, the Decision is translated into BPML Choice. At run time, the outgoing connector is selected depending on the latest message coming to the process. This message can be the output message received from the previous activity for the Choice Decision or the message triggered from a sub-process using Send Message event. At design time, when a Receive Message is used after a Decision construct, the condition on the Decision cannot be left empty.



During run time, the main process remains in waiting state when a Receive Message event is encountered. This Receive Message specifies an input message that is passed from the Send Message event construct in the sub-process, if the Sub-process is an activity prior to the Receive Message and should not have any output message. If the sub-process has an output message, then, the main process remains in waiting state until the sub-process is executed and the output message is received by the main process.

Compensate event

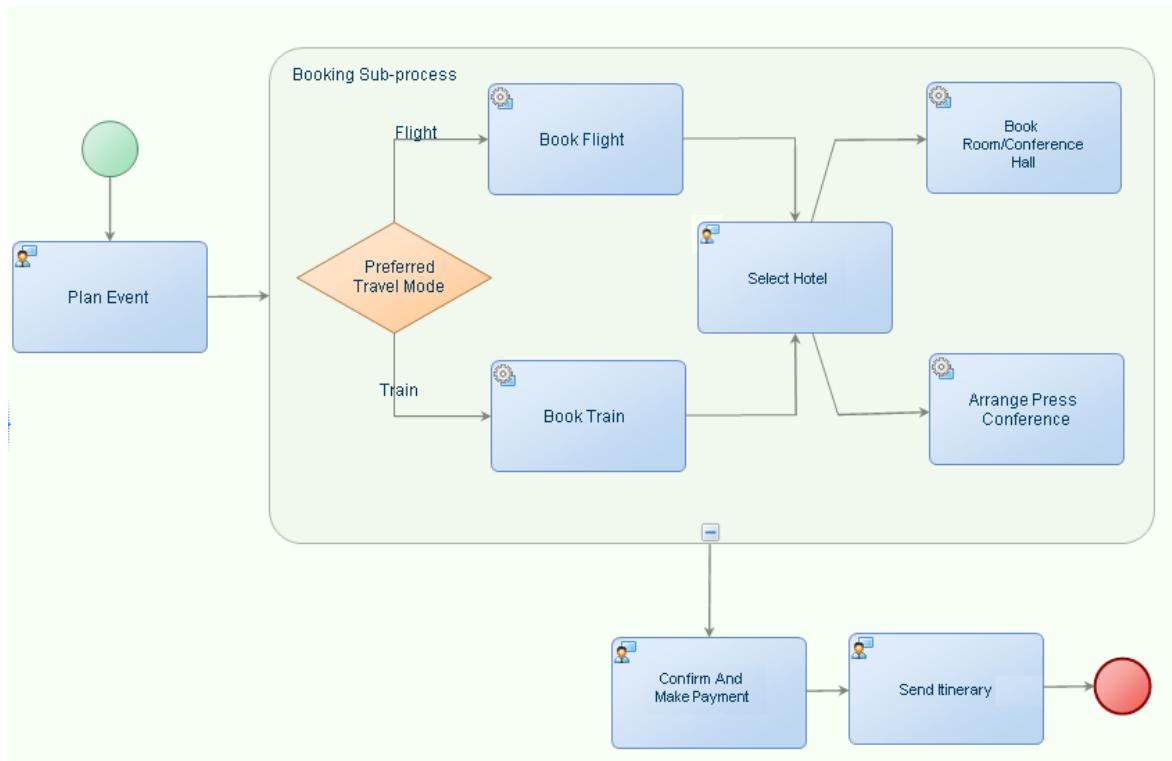
Since BPMN is mostly used for the orchestration of heterogeneous Web services and Asynchronous activities, which are long running, the well known ACID transaction is not always a possibility. However, AppWorks Platform supports ACID transaction for WS-AppServer based activities and application based activities that run in the same application server context. So, to maintain the integrity of the business, BPMN has introduced the concept of compensation. According to this concept, when we're not able to roll back an already performed activity, we can compensate that activity with another activity that can counter the previous action. In the well known example of electronic money transfer, money credited to your account can be rolled back by debiting the same amount to your account.

A Compensate event is a BPMN construct, which reverses or rolls back the effects of an operation that has been executed. A **Compensate event** is triggered when an error is encountered during a process. Compensate event rolls back the actions prior to the occurrence of the error. A Compensate event occurs outside the normal flow of the business process, and is triggered based on the rollback of a transaction. A Compensate event can be defined for an Activity, Sub-process, For Each, While, Until, and Context BPMN constructs.

However, an Activity or Sub-process cannot have more than one Compensate event. Compensate event is graphically represented as .

Use case

Using an example of a Conference Management portal, the portal takes care of your travel itinerary, accommodations, and conference hall booking for conducting conferences in any major cities of Western Europe.



In this example, the business process is simple. The customer provides his/her travel plan and preferred mode of travel. Once the train or flight tickets are booked, the customer is provided with a list of hotels to choose from. Once he/she selects a hotel, the guest room and the conference room are booked and simultaneously, the press conference for that event is arranged. Once all these bookings are done, the customer has to make the payment. When the payment is made, the itinerary is sent to the customer.

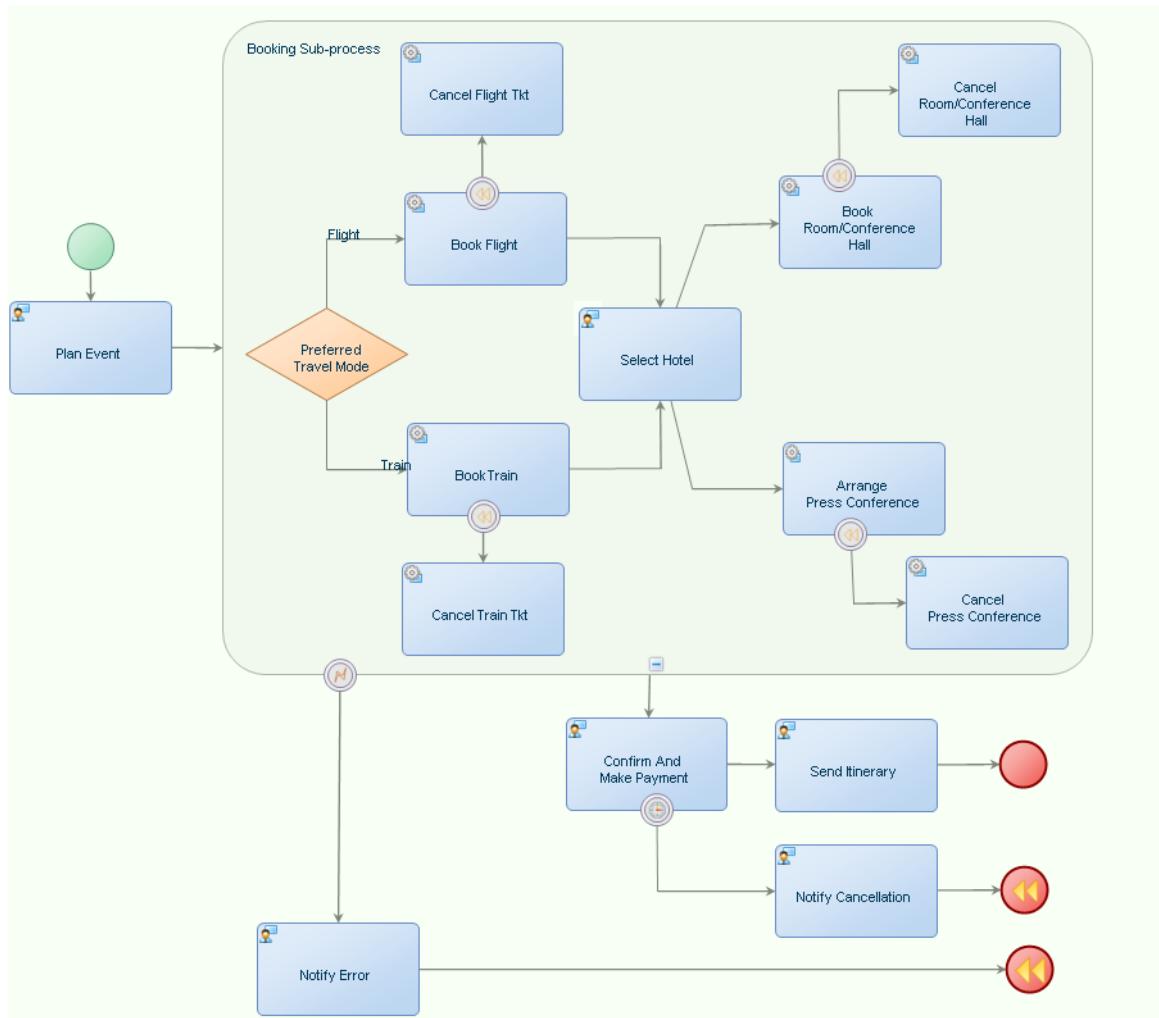
In the booking sub process, you will find that there are four web service operations and a task. The task is just a read operation, which provides a list of hotels to select. But the web services perform update operations and they belong to different systems across the network. So, here it is not possible to define an ACID transaction for these kind of operations. When all these operations in the above process work fine, there will not be any integrity problem. The problems may arise only when:

- The user has selected to travel by train, the train ticket is booked and when the process tries to book the hotel, the hotel booking service is not available or the network connection is down?

- The hotel booking is also done but the press conference request fails?
- It is normal in travel industry to block until the customer pays to make it a confirmed booking. If the customer doesn't respond within the prescribed time limit then, the booking is canceled
- So, here if the customer does not make payment in three days, all the books stand canceled/released.

Solution proposal

To satisfy all the requirements, we have to introduce compensating activities in the above business process. When we associate compensating activities to the update operation in the above process, the following diagram illustrates the process.



In the model, each operation that can modify the state is associated with a compensating activity which nullifies the change that is done by its counterpart. *Cancel TrainTkt* activity deletes the booking performed by the *Book Train* activity. The activity *Select Hotel* in the above model is not associated with any compensating activity because this is just a read operation.

When process instance executes an activity, it checks for the availability of compensating activity for that operation. If the operation exists, the compensating activity is put into a stack. This stack piles up as the process execution continues. In the above case, if the user selects train as the mode of transport, then Book Train activity will be executed and Cancel TrainTkt activity will be put into the compensation stack. Since the Book Flight activity is not executed, the Cancel FlightTkt activity will not be added to the compensation stack.

To execute the compensation activities, the process should end with roll back. When the process ends with a rollback, the process instance pops up the activities one after another from the compensation stack and executes the same. In the model, we have to define an exception handler for the Booking sub-process, to ensure that in case of failure in that context, the customer is notified about the error and the process ends with a rollback. This ensures the all the activities that are performed until then are compensated. Let us assume that the process has booked the train ticket and the hotel, in which case, the compensation stack will have Cancel Hotel and Cancel TrainTkt activities. If the Arrange Press Conference fails, these two compensating activities will undo the train and hotel bookings.

We have defined a timeout of three days for the payment activity, within which time if the customer does not pay, all the booking reverts via compensation.

Delay

Delay is a BPMN construct that is used to stop the process execution for a while and continue after some time. The business process remains in waiting state when Delay is being executed.

Delay is graphically represented as .

Catch exception

A Catch Exception event is a BPMN construct and is used for specific error occurring in the Activity or Context to which it is attached. It reacts to a named error or to any error if no name is specified, when attached to an Activity.

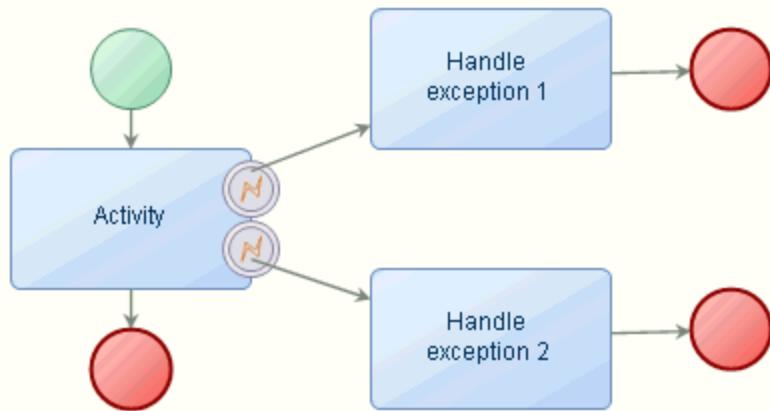


The Catch Exception event is graphically represented as .

The following table describes the different types of errors that fire a Catch Exception event.

Error Type	Description
All	In this type, no error code is specified and a Catch Exception event is fired for all errors.
Custom Error	You can specify the error code in this type of error. For example, every method (SOAP request) can have its own error codes (SOAP faults).
System Error	System error is an internal error that occurs if: <ul style="list-style-type: none"> ■ Business Process Management database is down. ■ Date formats are wrong.

Error Type	Description
	<ul style="list-style-type: none"> Any internal error occurs during execution (for example, model definition errors) . System error relates to fault code 2 supported by AppWorks Platform Business Process Engine.
Communication failure	If an activity in a business process model communicates with an external Web service or an application through SOAP, and if you get SOAP faults, then Communication failure occurs. For example, if you try to generate <code>GetEmployeeObject</code> SOAP method and if WS-AppServer application connector is down, then Communication failure occurs. Communication failure relates to fault code 11 supported by AppWorks Platform Business Process Engine. All SOAP faults are categorized as Communication failure by default, for easy error handling.
Process Loading Error	This relates to fault code 14 supported by AppWorks Platform Business Process Engine. This is applicable only if the Catch Exception event is attached to a sub-process.
Process Instantiation Error	This relates to fault code 16 supported by AppWorks Platform Business Process Engine. This is applicable only if the Catch Exception event is attached to a sub-process.
Process Model not Found Error	This relates to fault code 23 supported by AppWorks Platform Business Process Engine. This is applicable only if the Catch Exception event is attached to a sub-process.



You can link one or more Catch Exception events to an Activity, Embedded Sub-process, For Each, While, Until, Independent Sub Process, Web service, Decision Table and XForm or External User Interface. If you are attaching more than one Catch Exception event to an Activity or Sub Process, then select the All option for one Catch Exception event. In such a case, this Catch Exception event will be the default one.

Independent subprocess

An Independent Subprocess is a compound activity that is included within a process. It can be broken down into a finer level of detail through a set of sub activities. An Independent Subprocess is instantiated through a parent process. The activities of the Independent Subprocess are visible in the business process model diagram only when you click  within the Independent Subprocess.

An independent subprocess can be designed as part of a transaction, in which case there can be three basic possible outcomes of the transaction:

- **Successful completion:** The process continues in a sequence flow and the transaction is successfully executed.
- **Cancel:** The activities within a transaction are rolled back and specific activities may be compensated.
- **Exception:** The activity is disrupted without any rollback and the flow will continue from the error event.

An Independent Subprocess within a transaction behaves differently than a normal subprocess. Each activity within a transaction subprocess is validated to check if all the participants of the transaction have successfully completed their activities, after which the flow moves to a higher-level process. Even if any one of the participants has not completed their activities it results in a Cancel or an Exception and then the flow moves on to an appropriate Intermediate Event.

Send message event

Send Message Intermediate Event is a BPMN construct and is used to send a message from the process to the other. A Send Message Intermediate Event can be used to synchronize the sub-process with another process if the Wait until Sub Process is finished option is not flagged for the independent sub-process. If this option is not selected, then the parent process or the specified process continues irrespective of an incomplete sub-process. After some activities, some information/data may have to be sent to the process that has been specified to receive the intermediate message. So, a Receive Message Intermediate Event is modeled in that process, which picks up the message from the Send Message event construct in the independent sub-process. The Receive Message Intermediate Event in that particular process should have the same message name linked as the one linked in the Send Message Intermediate Event in the independent sub-process.

The message to be sent can be linked to the Send Message event by dragging a process-specific message or activity messages from the project content tree in Workspace Documents and dropping it on the Send Message event in the Independent Sub Process and/or the Receive Message in the parent process.

Time-out

Time-out event is a BPMN construct, and is fired when the Activity or Sub-process is not executed within the specified time. A Time-out event can be defined for an Activity, Group (Context, For Each, While and Until), Receive Message, and Sub-process. An Activity or Sub-

process cannot have more than one Time-out event. Time-out is graphically represented as

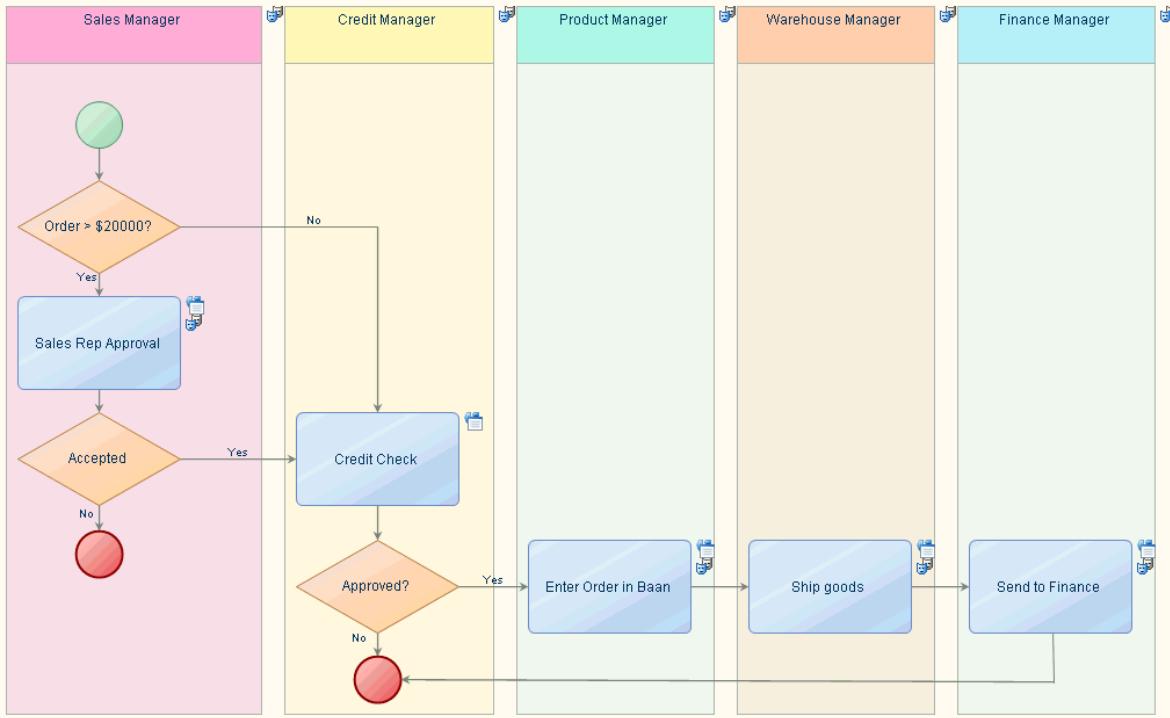


Swimlane

A swimlane layout organizes the BPMN constructs in a business process model into a set of rows called swimlanes. A swimlane is graphically represented as a row or a column, which in turn, represents a logical group of activities in a business process model. Swimlanes help you easily identify activities belonging to a logical group in different rows. The layout does not affect the execution of the business process model. The purpose of creating a swimlane is to demarcate a set of activities or tasks that need to be performed either by an Organization Unit (at design time) or a Team (at run time) or by a Role. Swimlanes can be created by drag-dropping an Organization Unit on to an activity or set of activities in a business process or by drag-dropping a Role on to a business process. Based on what you method you employ for creating a swimlane, the definition of the swimlane will change as follows:

- In an organization model context, a swimlane is a graphical representation of an organization unit at design time and a graphical representation of a Team at run time. This means that all the activities or tasks that fall within this swimlane need to be performed only by the Organization Unit or Team using which the swimlane is created.
- In roles context, a swimlane is a graphical representation of a role that performs all the set of activities or tasks that fall within that lane. This means that all the set of activities or tasks that fall within this swimlane are to be performed only by the specific role using which the swimlane is created.

For example, in the following business process diagram, activities are classified according to the department to which they belong.



A process can have all swimlanes either in a vertical or horizontal layout, but not as a mix of both. The lanes spread over the entire length or breadth of the modeling environment. You can resize the width of a vertical swimlane and the height of a horizontal swimlane.

The swimlane header, located at the top of each swimlane (in a vertical layout) and at the left of each swimlane (in a horizontal layout), identifies the constructs that are associated with the swimlane.

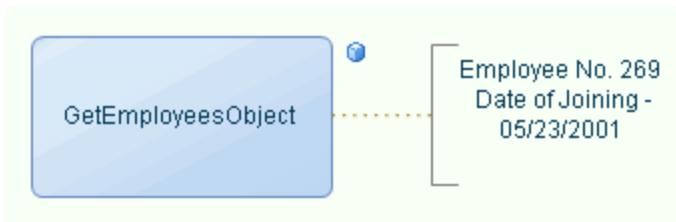
The business process modeling editor allows you to customize the following:

- You change the order in which the swimlanes are displayed in the modeling environment. You can insert new swimlanes in an existing layout. When you insert a new lane, for example, lane 4 between lanes 2 and 3, lane 3 shifts and the sequence changes from lane 1-2-3 to 1-2-4-3.
- The swimlane behaves like a group activity; when a lane is moved, the activities within the lane also move. However, an activity from one lane can be moved to another without moving the lane.
- You can use different colors for the swimlanes, with a combination of the shade for the header and the lane.
- A swimlane can have multiple roles. Roles can be dragged to the swimlane from the project content tree of the Workspace Documents (Explorer) view, or configured in the Properties - Lane pane. An attached role is indicated by the role icon appearing in the swimlane. The default description of the swimlane header changes to the description of the role attached. When a role is attached to a swimlane, all tasks within the lane inherit the role from the swimlane. At any point of time, only a single role is associated with a

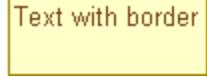
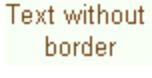
task. However, the role inheritance is applicable for activities within the swimlane and tasks without dynamic user assignment.

Annotation

An annotation is a means to provide additional information in the business process diagram. For example, you may want to display or print details about the business process model, such as modifications made to the process, modified date, and name of the business analyst. A text annotation is graphically represented as follows.



The following table describes the different types of annotation.

Type	Description	Graphical representation
Text Annotation	The text annotation can be associated with a specific activity in a business process model. Text annotations can be connected to any activity in the business process model, but they do not affect the flow of the process.	
Annotation with border and background color	This type of annotation consists a border with a background color, similar to post-it notes. It cannot be associated with an activity in a business process model.	
Transparent annotation without border	This type of annotation is transparent and does not have a border. It cannot be associated with an activity in a business process model.	

Group constructs

BPMN Group constructs are artifacts that provide a mechanism to group process elements informally. A group construct is not an activity or a flow object but a set of Activities or Subprocesses that should be executed repeatedly. The following BPMN group constructs are supported by AppWorks Platform.

- [For each](#)
- [While](#)

- Until
- Embedded sub-process
- Transaction

For each

For Each is a BPMN group construct and represents Activities or sub-processes that should be executed for each sub part of a message. The For Each loop uses a counter, known as an iterator to specify the number of times the same sequence of activities should be repeated.

The iterator has the following three numeric values:

- Initial iterator value
- Incremental value
- Execute Condition

The loop ends when the execute condition fails. The XPath of the recursive element should be copied in the Message Map for this construct to work.

Example of a For Each loop

Consider the case of a HR Manager who wants to view the employee records of the first five employees. When the manager reaches the fifth record, the process of fetching employee records stops.

The example can be modeled as follows:

Incremental value = iterator value +1

All employee records will be fetched until the iterator count reaches the fifth record, after which the process stops.

Consider an example of a filter condition in the For Each loop:

```
'/ns2:GetEmployeesOutput/ns2:GetEmployeesResponse/ns2:tuple/
ns2:old/ns2:Employees[ns2:TitleOfCourtesy = 'Ms.'].'
```

Here the loop is not iterated for all the records in the response, but only for those records which have the value 'Ms.' in the element 'TitleOfCourtesy'.

Since you have defined a filter for these records, you must not use the complete path (' ns2:GetEmployeesOutput/ns2:GetEmployeesResponse/ns2:tuple/ns2:old/ns2:Employees/ ns2:TitleOfCourtesy ') for the assignment within the For Each construct. You must always use the iterator name, for example, /instance:iterator_o_34/ns2:TitleOfCourtesy/text(). Alternatively, you can use the entire path with the filter ('/ns2:GetEmployeesOutput/ns2:GetEmployeesResponse/ns2:tuple/ns2:old/ns2:Employees[ns2:TitleOfCourtesy = 'Ms.'].'). This is because the filter used in the For Each loop filters records that have the value 'Ms.' in the elementTitleOfCourtesy. When you use the entire path without the filter in the assignment, it considers the entire set of records without the filter. However, the assignment will not work for records that do not meet the filter condition; hence, you must always use the iterator name as a reference for the filter condition or use the complete path with the filter.

While

While is a BPMN group construct used to group activities or sub-processes that should be executed while the condition is satisfied. At the start of every While loop, the condition is tested and if found false will not be executed. Accordingly, the activities or sub-processes are executed zero or more times. The While construct is used when the loop is to be executed more number of times or none at all; that is to say, it will not be executed if the condition is false from the start.

Example of a While Loop

In a stock verification and update process, a condition can be set to send a notification to the Purchase Manager while the stock is less than 10 units.

The example can be modeled with the following condition: Notify the Purchase Manager while (stock is less than 10 units).

Until

Until is a BPMN group construct, which is used to group activities or sub-processes that must be executed until the condition becomes true. Unlike the While construct, the activities, or sub-processes are executed at least once.

Example of an Until Loop

Consider the example of a stock verification and update process, when stock reaches re-order level (ROL), the purchase manager is notified and fresh stock is ordered until stock level reaches 100 units. This process can be modeled as:

- Check stock; if stock reaches ROL, notify purchase manager
- Order stock until stock level reaches 100 units

Transaction

A Transaction is a BPMN group construct and is defined as a set of activities or sub-processes that must be executed entirely or not at all. Thus, in a Transaction, either all operations are committed or all operations are rolled back. A Transaction therefore, can be a single unit of work. This group construct is used when either all the activities within the transaction should be executed once or none at all.

Example of a Transaction

An order processing process comprises the following steps:

1. Customer enters order details.
2. Stock inventory is verified to know if the order placed by the customer can be processed.
3. If stock exists, update requested quantity.
4. Generate invoice and update Accounts Receivable.

In this process, consider a scenario where the stock quantity in the inventory is updated, but the generation of invoice and updating of Accounts Receivable fails. In this case, either all the steps should be executed or none at all. To ensure this behavior, these activities can be grouped as a transaction. Thus, in case an exception is raised for any single operation, the entire set of activities will be rolled back.

For a transaction to work in a sub-process, we need to define the transaction in the sub-process too.

Embedded sub-process

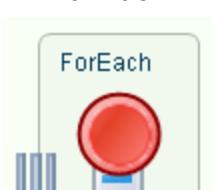
The embedded sub-process is a BPMN group construct used to group activities that belong to a particular context. This helps to hide complexity of activities by collapsing an embedded sub-process. It also helps to link time-outs, exceptions and compensates which are common for all activities within the embedded sub-process.

- When a Time-out event is attached to an embedded sub-process, all incomplete tasks are removed from the Inbox.
- If 'terminate activity' is encountered during execution, then all incomplete tasks are removed from the Inbox, and the activity is set to 'Obsolete'. This status is visible from the PIM activity view. Terminate indicates that the embedded sub-process is terminated and the process flow execution continues from next activity (after embedded sub-process).

End event

An End event is a BPMN construct that indicates an end of a process. An End event will not have any outgoing process flow. A valid process model should have at least one End event, and there can be multiple End events in a process model. The outcome of reaching an End event is known as an End event result, based on which, there are different types of End events.

Type	Description	Graphical representation
Abort	<p>This type of End event functionality aborts process execution.</p> <p>The Abort End Type appears only when you right-click the End event, select the Group as option, and select Transaction.</p>	
Break Loop	<p>This type of End event functionality helps to break and exit out of a loop and the execution continues from the next immediate activity after the loop.</p> <p>The Break Loop End Type appears only when you right-click the End event, select the Group as option and select any one of the following Group Constructs:</p>	

Type	Description	Graphical representation
	While, Until and For Each.	<p>Until:</p>  <p>For Each:</p> 
Continue Loop	<p>This type of End event functionality helps to break the current loop and start with the next iteration of the loop.</p> <p>The Continue Loop End Type appears only when you right-click the End event, select the Group as option, and select any one of the following Group Constructs: While, Until, and For Each.</p>	<p>While:</p>  <p>Until:</p>  <p>For Each:</p> 
Error	<p>This type of End event indicates that a named error should be generated at the conclusion of the process. This error will be caught by an Intermediate event within the event context. In a sub-process, a process error can be used for exception handling, which will be</p>	

Type	Description	Graphical representation
	linked to the exception handling event of the parent process.	
Message	This type of End event indicates that a message is sent to a participant at the conclusion of the process. At the end of the process, the sub-process to the parent process sends an output message.	
None	This type of End event does not have any result defined. In a sub-process, the end event triggers the process flow to go back to the parent process.	
Rollback	This type of End event indicates that a compensation is necessary. The compensation identifier will trigger an Intermediate event when the process is rolled back.	
Terminate	This type of End event functionality terminates process execution. The Terminate End Type appears only when you right-click the End event, select the Group as option, and select Embedded Sub-process.	

Business modeling roles

AppWorks Platform business modeling environment supports the following roles:

- Process Developer
- Process Administrator

Process Developer

A user with the role of a process developer models business processes in the AppWorks Platform environment. A process developer can also perform the functions of a business analyst. A process developer can exclusively perform the following functions:

- Validate a process model
- Publish a process model
- Run a process model

Process Administrator

A user with the role of a process administrator deploys, monitors and archives published business processes in the AppWorks Platform environment. A process administrator can perform the following functions:

- View audit information of published processes
- Monitor business process instances
- Archive process instances

Operation types

Operation type defines the actions that can be carried out on Message Map assignments. The following table describes the operation types supported by AppWorks Platform. Each operation can be performed with certain parameters. The parameters are discussed separately.

Add	The Add operation inserts the source element (either simple or complex) into the target element with already existing children of the target element and forms a combined structure.
Add XML as String	The Add XML as String operation adds the selected source node as a string (data) to the target node. If there are other children (data nodes or element nodes), the selected value will be added as sibling text data. This operation can be used either with Fixed Value or with any of the Select parameters. Parameters including Expression and XML Structure cannot be used for this operation.
Replace With	The Replace With operation replaces the target element tag with the source element at run time. This operation can be used either with a Fixed Value or with any of the Select parameter types.
Replace Content With	The Replace Content With operation replaces all the content (all child elements) of the target element. Consider the following example: <code><element1>example</element1></code> The content of the element1 tag, 'example', is replaced with the source element or text as specified in the XPath.
Replace XML as String	The Replace XML as String operation replaces all child nodes of the target with a string. The selected node is added as data node to the target node as text data. This operation can be used with either Fixed Value or with any of the Select parameters. Parameters such as Expression and XML Structure cannot be used for this operation.
Delete	The Delete operation deletes the target element at run time. You do not need to specify the source element for the Delete operation. Note: This operation deletes the element itself irrespective of the parameter (Select, Children, First Child, Last Child, Fixed Value, Expression and XML Structure) used in the assignment.

Examples of Add

Add with Select

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple and the XML structure of the Target element before mapping the message is as follows:</p> <pre><UpdateInput> <Update> <tuple> <old/> </tuple> </Update> </UpdateInput></pre>	<p>The Source element is /new and the XML structure of the Source element will be as follows:</p> <pre><new> <Orders> <OrderID>100</OrderID> </Orders> </new></pre>	<pre><UpdateInput> <Update> <tuple> <old/> <new> <Orders> <OrderID>100</OrderID> </Orders> </new> </tuple> </Update> </UpdateInput></pre>

Add with Fixed Value

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /Orders/OrderID and the XML structure of Target element before mapping the message is as follows:</p> <pre><Orders> <OrderID/> </Orders></pre>	<p>The Source element is 250 and the XML structure of the Source element is 250.</p>	<pre><Orders> <OrderID>250</OrderID> </Orders></pre>

Add with Expression

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /Employee/AnnualSalary and the XML structure of the Target element before mapping the message is as follows:</p> <pre><Employee> <AnnualSalary/> </Employee></pre>	<p>The Source element is Employee/MonthlySalary * 12 and the XML structure of the Source element is $1000 * 12 = 12000$.</p>	<pre><Employee> <AnnualSalary>12000</AnnualSalary> </Employee></pre>

Add with XML Structure

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple and the XML structure of the Target element before mapping the message is as follows:</p> <pre><UpdateInput> <Update> <tuple> <old/> </tuple> </Update> </UpdateInput></pre>	<p>The Source element is <new><Orders><OrderID/></Orders></new> and the XML structure of Source element will be as follows:</p> <pre><new> <Orders> <OrderID/> </Orders> </new></pre>	<pre><UpdateInput> <Update> <tuple> <old/> </tuple> </Update> </UpdateInput></pre>

Examples of Add XML as string

Add XML as String with Select

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /GetNotificationTaskInput/GetNotificationTask/Task and the structure of Target element before mapping the message is as follows:</p> <pre><GetNotificationTaskInput> <GetNotificationTask> <Header>cn=rsahu, cn=authenticated users, cn=cordys, o=vanenburg.com</Header> <Task>Dummy Task</Task> </GetNotificationTask> </GetNotificationTaskInput></pre>	<p>The Source element is /registeredAddress and the XML structure of the Source element is as follows:</p> <pre><registeredAddress> <string>Winrock Boulevard, Software Units Layout, Freemont CA - 87757 USA</string> </registeredAddress></pre>	<pre><GetNotificationTaskInput> <GetNotificationTask> <Header>cn=rsahu, cn=authenticated users, cn=cordys, o=vanenburg.com</Header> <Task>Dummy Task <registeredAddress> <string>Winrock Boulevard, Software Units Layout, Freemont CA - 87757 USA</string> </registeredAddress> </Task> </GetNotificationTask> </GetNotificationTaskInput></pre>

Add XML as String with Fixed Value

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /Orders/OrderID and the XML structure of the Target element before mapping the message is as follows:</p> <pre><Orders> <OrderID/> </Orders></pre>	<p>The Source element is 250 and the XML structure of the Source element is 250.</p>	<pre><Orders> <OrderID>250</OrderID> </Orders></pre>

Examples of Replace With

Replace With with Select

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple/old and the XML structure of the Target element before mapping the message is as follows:</p> <pre><UpdateInput> <Update> <tuple> <old> <Orders> <OrderID>100</OrderID> </Orders> </old> </tuple> </Update> </UpdateInput></pre>	<p>The Source element is new and the XML structure of the Source element is as follows:</p> <pre><registeredAddress><new> <Orders> <OrderID>250</OrderID> </Orders> </new></pre>	<pre><UpdateInput> <Update> <tuple> <new> <Orders> <OrderID>250</OrderID> </Orders> </new> </tuple> </Update> </UpdateInput></pre>

Replace With with Fixed Value

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /Orders/OrderID and the XML structure of the Target element before mapping the message is as follows:</p> <pre><UpdateInput> <Update> <tuple></pre>	<p>The Source element is 250 and the XML structure of the Source element is 250.</p>	<pre><Orders>250<Orders></pre>

Target element before message mapping	Source element	Target element after message mapping
<pre> <old> <Orders> <OrderID>100</OrderID> </Orders> </old> </tuple> </Update> </UpdateInput> </pre>		

Replace With with Expression

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /Employee/AnnualSalary and the XML structure of the Target element before mapping the message is as follows:</p> <pre> <UpdateInput> <Update> <tuple> <old> <Orders> </pre> <p><OrderID>100</OrderID></p> <pre> </Orders> </old> </tuple> </Update> </UpdateInput> </pre>	<p>The Source element is Employee/MonthlySalary * 12 and the XML structure of the Source element is 1000 * 12 = 12000.</p> <pre> <new> <Orders> <OrderID/> </Orders> </new> </pre>	<pre><Employees>12000<Employees></pre>

Replace With with XML Structure

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple /old and the XML structure of the Target element before mapping the message is as follows:</p> <pre> <UpdateInput> <Update> <tuple> <old> <Orders> </pre>	<p>The Source element is <new><Orders><OrderID/></Orders></new> and the XML structure of the Source element is as follows:</p> <pre> <new> <Orders> <OrderID/> </Orders> </new> </pre>	<pre> <UpdateInput> <Update> <tuple> <new> <Orders> <OrderID/> </Orders> </new> </tuple> </Update> </UpdateInput> </pre>

Target element before message mapping	Source element	Target element after message mapping
<pre><OrderID>100</OrderID> </Orders> </old> </tuple> </Update> </UpdateInput></pre>		<pre></Update> </UpdateInput></pre>

Examples of Replace XML as String

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /GetNotificationTaskInput/ GetNotificationTask/Task and the XML structure of the Target element before mapping the message is as follows:</p> <pre><GetNotificationTaskInput> <GetNotificationTask> <Header>cn=jdoe, cn=authenticated users, cn=cordys, o=vanenburg.com</Header> <Task>Dummy Task</Task> </GetNotificationTask> </GetNotificationTaskInput></pre>	<p>The Source element is Address/ registeredAddress and the XML structure of the Source element is as follows:</p> <pre><Address> <registeredAddress> <string>Winrock Boulevard, Software Units Layout, Freemont CA - 87757 USA</string> </registeredAddress> </Address></pre>	<pre><GetNotificationTaskInput> <GetNotificationTask> <Header>cn=jdoe, cn=authenticated users, cn=cordys, o=vanenburg.com</Header> <Task> <Address> <registeredAddress> <string>Winrock Boulevard, Software Units Layout, Freemont CA 87757 USA</string> </registeredAddress> </Address> </Task> </GetNotificationTask> </GetNotificationTaskInput></pre>

Examples of Replace Content With

Replace Content With with Select

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple and the XML structure of the Target element before mapping the</p>	<p>The Source element is /new and the XML structure of the Source element is as follows</p>	<pre><UpdateInput> <Update> <tuple> <new></pre>

Target element before message mapping	Source element	Target element after message mapping
message is as follows: <UpdateInput> <Update> <tuple> <old/> </tuple> </Update> </UpdateInput>	<new> <Orders> <OrderID>100</OrderID> </Orders> </new>	<Orders> <OrderID>100</OrderID> </Orders> </new> </tuple> </Update> </UpdateInput>

The following is another example of this operation type.

Target element before message mapping	Source element	Target element after message mapping
The Target element is /Orders/OrderID and the XML structure of the Target element before mapping the message is as follows: <Orders> <OrderID>100</OrderID> </Orders>	The Source element is /OrdersDetails/OrderID/text() and the XML structure of the Source element is as follows: <OrdersDetails> <OrderID>250</OrderID> </OrdersDetails>	<Orders> <OrderID>250</OrderID> </Orders>

Replace Content With with Fixed Value

Target element before message mapping	Source element	Target element after message mapping
The Target element is /Employee/AnnualSalary and the XML structure of the Target element before mapping the message is as follows: <Orders> <OrderID>100</OrderID> </Orders>	The Source element is 250and the XML structure of the Source element is 250.	<Orders> <OrderID>250</OrderID> </Orders>

Replace Content With with Expression

Target element before message mapping	Source element	Target element after message mapping
The Target element is /Employee/AnnualSalary and the XML structure of the Target element before mapping the message is as follows:	The Source element is Employee/MonthlySalary * 12 and the XML structure of the Source element is 1000 * 12 = 12000.	<Employee> <AnnualSalary>12000 </AnnualSalary> </Employee>

Target element before message mapping	Source element	Target element after message mapping
<pre><Employee> <AnnualSalary>5000</AnnualSalary> </Employee></pre>		

Replace Content With with XML Structure

Target element before message mapping	Source element	Target element after message mapping
<p>The Target element is /UpdateInput/Update/tuple and the XML structure of the Target element before mapping the message is as follows:</p> <pre><UpdateInput> <Update> <tuple> <old> <Order> <OrderID>100</OrderID> <Order> <old> <Order> <OrderID>100</OrderID> <OrderAmount>5000</OrderAmount> </Order> </old> </Order> </old> </tuple> </Update> </UpdateInput></pre>	<p>The Source element is <new><Orders><OrderID/></Orders></new> and the XML structure of the Source element is as follows:</p> <pre><new> <Orders> <OrderID/> </Orders> </new></pre>	<pre><UpdateInput> <Update> <tuple> <new> <Orders> <OrderID/> </Orders> </new> </tuple> </Update> </UpdateInput></pre>

Examples of Delete

Target element before message mapping	Target element after message mapping
<p>The Target element is /Orders/OrderID and the XML structure of the Target element before mapping the message is as follows:</p> <pre><Orders> <OrderID>100</OrderID> <OrderAmount>5000</OrderAmount> </Orders></pre>	<pre><Orders> <OrderAmount>5000</OrderAmount> </Orders></pre>

Parameter types

The following table describes the parameter types supported by AppWorks Platform that enable you to map the source content to the target with all the necessary transformations based on the requirement.

Parameter Type	Description	Availability
Select	The Select parameter type denotes the XPath from which the input is considered. If the operation is of the type Select, then the Source column must contain a single XPath.	Available in both Business Process Model and Case model. When the value is left empty, the Case model results in a validation failure during design time. For the same scenario, a Business Process Model provides a warning message in design time and ignores the assignment during run time.
Select Multiple	The Select Multiple parameter type denotes the multiple XPath expressions from which the input must be considered.	Available in Business Process Model only. Not available in Case model. To achieve the required functionality, use the Expression source type and provide the relevant XPath.
Fixed Value	In the parameter type Fixed Value, the value in the Source column is considered as a literal value.	Available in both Business Process Model and Case model. When the '<!CDATA [concat]]>' value is considered in a Business Process Model, the net result of such content evaluation results only in 'concat'. The same in a Case model results in '<!CDATA [concat]]>'.
Expression	In the parameter type Expression, the result of expression evaluation is considered as the source value for assignment. If the '-' (minus) operator is used in the expression, then it must be delimited from operands using a space. Right-click the source field to select all the possible expression constructs. The functions that are used in the expression are executed by the XPath engine at run time. Refer	Available in both Business Process Model and Case model. When the value is left empty, the Case model results in a validation failure during design time. For the same scenario, a Business Process Model provides a warning message in design time and ignores the assignment during run time.

Parameter Type	Description	Availability
	to Creating an XPath Expression for information on the XPath expressions.	
Children	The Select parameter type denotes the XPath from which the input is considered. If the operation is of the type Children, then all the sub-elements are assigned to the target element. For example, all the tuple elements containing all orders can be assigned to the source element with the parameter type Children.	Available in Business Process Model only. Not available in Case model. To achieve the required functionality, use the Expression source type and provide the relevant XPath. For example, if the children of the element Employees must be selected, then use <code>emp:Employees/*</code> .
First Child	If the operation is First Child, then only the first sub-element is assigned to the target element. For example, the first tuple element containing the first order can be assigned to the source element with the parameter type First Child.	Available in Business Process Model only. Not available in Case model. To achieve the required functionality, use the Expression source type and provide the relevant XPath. For example, if the first child of the element Employees must be selected, then use <code>emp:Employees/*[1]</code> .
Last Child	If the operation is Last Child, only the last sub-element is assigned to the target element. For example, the last tuple element containing the last order can be assigned to the source element with the parameter type Last Child.	Available in Business Process Model only. Not available in Case model. To achieve the required functionality, use the Expression source type and provide the relevant XPath. For example, if the last child of the element Employees must be selected, then use <code>emp:Employees/*[last()]</code> .
XML Structure	In the XML Structure parameter type, the complete XML structure must be specified in the source column. For example:	Available in both Business Process Model and Case model. From this latest version of AppWorks Platform, a Case model evaluates XML input,

Parameter Type	Description	Availability
	<pre><tuple> <new> <Orders> <OrderID>90</OrderID> </Orders> </new> </tuple></pre>	and provides a validation error if the XML input provided is invalid.
Select with Target NS	Same as Select. If this option is selected, the namespace of the target element is considered.	Available in Business Process Model only. Not available in Case model.
Select Multiple with Target NS	Same as Select Multiple. If this option is selected, the namespace of the target elements are considered.	Available in Business Process Model only. Not available in Case model.
Children with Target NS	Same as Children. If this option is selected, the namespace of the target element is considered.	Available in Business Process Model only. Not available in Case model.
First Child with Target NS	Same as First Child. If this option is selected, the namespace of the target element is considered.	Available in Business Process Model only. Not available in Case model.
Last Child with Target NS	Same as Last Child. If this option is selected, the namespace of the target element is considered.	Available in Business Process Model only. Not available in Case model.
XML Structure with Target NS	Same as XML Structure. If this option is selected, the namespace of the target element is considered.	Available in Business Process Model only. Not available in Case model.
Add Attribute nil="true"	Should pass an empty value. If this option is selected, the target element value is empty.	Available in Business Process Model only. Not available in Case model.

Note: The operations dealing with XML namespace manipulation, such as Replace content with target NS, are not made available in the Case model. This was provided in BPM message map to maintain backward compatibility with the C2 version of AppWorks Platform.

Interprocess communication in business process models

In any business process based application, interaction between different business process models is inevitable. The following sections describe the four variants in which the inter-process communication can happen.

Synchronous instantiation of a new sub-process

It is possible to trigger synchronously a sub-process instance with a message, and receive a message once the sub-process instance completes its execution. This is a scenario where the communication between the main process and sub-process happens during the instantiation of the sub process, and for receiving back the message when it is complete. In this scenario, the constructs that enable the communication are 'Message Start Event' and 'Message End Event'. In AppWorks Platform, the parent process will maintain the reference (InstanceId) for the sub-process and the sub-process will maintain the reference (InstanceId) of the parent process.

Process instance communicating to an already instantiated sub-process

It is possible to send across a message to an already instantiated sub-process. In this case, the sub-process should have an event to receive the particular message. For this, Receive Message Intermediate Event has to be used in the sub-process and the parent process should have a Send Message Intermediate Event to send the appropriate message. Since the sub-process is instantiated from the main process, the main process maintains the InstanceID of the sub process. Use this instanceID to send across the message to the specific sub-process.

Sub-process instance communicating back to parent process instance

AppWorks Platform provides support for having message exchange from the sub-process instance to the parent instance. In this scenario, the parent process should be in a position to receive a particular message and the sub-process should be sending the respective message. The instanceID of the parent process should first be passed to the sub process while instantiating the same. The sub-process to communicate the message to the parent process should use the instanceID received from the parent process.

Communication between two unrelated process instances

AppWorks Platform supports communication between instances that are not directly associated during the design-time. In this scenario there can exist a communication between two unrelated process instances. Apart from the structure of the message that has to be sent across, the instanceID of the target process instance to which message needs to be sent is also needed. One approach to assign the target instanceID, is by managing the same in message map and using it to pass across the message. Alternatively, it is possible to identify the instance based on the query formed using Business Identifiers. If the second approach is being considered, care should be taken by the process developer to ensure that the query results in one specific target instance.

Process execution modes

Define the process execution mode of a business process in the properties of a business process. You can execute a business process in the following modes:

- **Long Lived:** Select the execution mode of a business process as Long Lived if it includes human interactions, delays, or intermediate messages and takes a long time to execute. A long-lived process has a long execution span whereas a short-lived process has a very short execution span. By default, a process is long-lived but can be defined as a short-lived process.
- **Short Lived:** Select the execution mode of a business process as Short Lived if it does not require human intervention and its average life span is short. This optimizes the process behavior. To achieve higher performance, short-lived processes are executed synchronously within the context of a single server thread. Since server resources are blocked for higher performance, it is recommended to design short-lived processes for a set of activities that have a short execution time and do not require human intervention for their completion. They can be used if a set of activities require transactional behavior.
- **Page Flow:** Select the execution mode of a business process as Page Flow if you want the user to perform a sequence of operations such as filling multiple forms in a wizard-like mode instead of accessing the same through AppWorks Platform Inbox. A page flow is a logical sequence of HTML pages or User Interfaces (UI) that require a user to fill multiple forms. The interactive page flow support in AppWorks Platform BPM is seamlessly integrated with User Interfaces that appear in the same frame upon filling the required information and submitting it. You can execute a business process in page flow mode by configuring the properties of the business process. In the page flow mode, the business process model has multiple tasks modeled for the process user. Only task notification messages are published to the client UI. Informative messages cannot be published to the client UI as an Info task can branch into one or more notifications, which will not produce a page flow behavior to the application. For example, in a business process designed to allow users to apply for a credit card, the user must fill multiple forms with a possibility that some forms may be rendered after evaluating the inputs in the previous form. You can specify that if the annual income of the applicant is less than the specified limit, the form for the Silver Card must be displayed; otherwise, the form for the Gold Card must be displayed.

You can generate Web services for both short-lived and long-lived published business processes. If a short-lived process ends with a message, it must have either only one End event with a message or multiple End events with the same message.

The following table describes the differences between long-lived and short-lived processes.

Long-Lived Process	Short-Lived Process
Complete level of monitoring along with activity monitoring is enabled by default.	Basic level of monitoring without activity monitoring is enabled by default.

Long-Lived Process	Short-Lived Process
Crash recovery along with activity monitoring is enabled by default.	Crash recovery is disabled by default.
Long-lived processes are standalone processes and may comprise multiple short-lived processes.	Short-lived processes usually form a part of long-lived processes.
Long-lived processes involve human interactions, delays, or intermediate messages. They encompass integration such as system-to-system and workflow such as human-to-human layers, and span multiple systems, people, and organizations. Therefore, it is necessary to monitor and store information at each step of the process.	Short-lived processes do not involve human interactions, delays, or intermediate messages. A process is not monitored by default. However, the start time, end time, and final status are monitored by default. You can model one-sided notifications that do not require human interaction for short-lived processes.
The average life span of a long-lived process is long, that is, the process takes a long time to execute. By default, all processes in the business process modeling environment are executed as long-lived.	The average life span of a short-lived process is short. The default completion time is 30 seconds. You may increase the time-out value if the process takes longer.
The process is executed by multiple threads.	The process is executed by a single thread.
The processes are asynchronous in nature such that for a request sent, a response is not required. The process continues once the instance ID is returned.	The processes are synchronous in nature such that for a request sent, a response is received after the process is executed.

Important: If you enable crash recovery or monitoring at the message map level for a short-lived process, the performance will reduce significantly. This occurs as the requests are synchronous in nature for a short-lived process, which thereby increases the overhead on the Process Engine while updating the database with monitoring information.

Modeling a transaction

A set of activities is grouped as part of a transaction when they must be executed either entirely or not at all. In a transaction, all operations are either committed or rolled back.

Before you begin:

- Ensure that message mapping for all activities is completed.

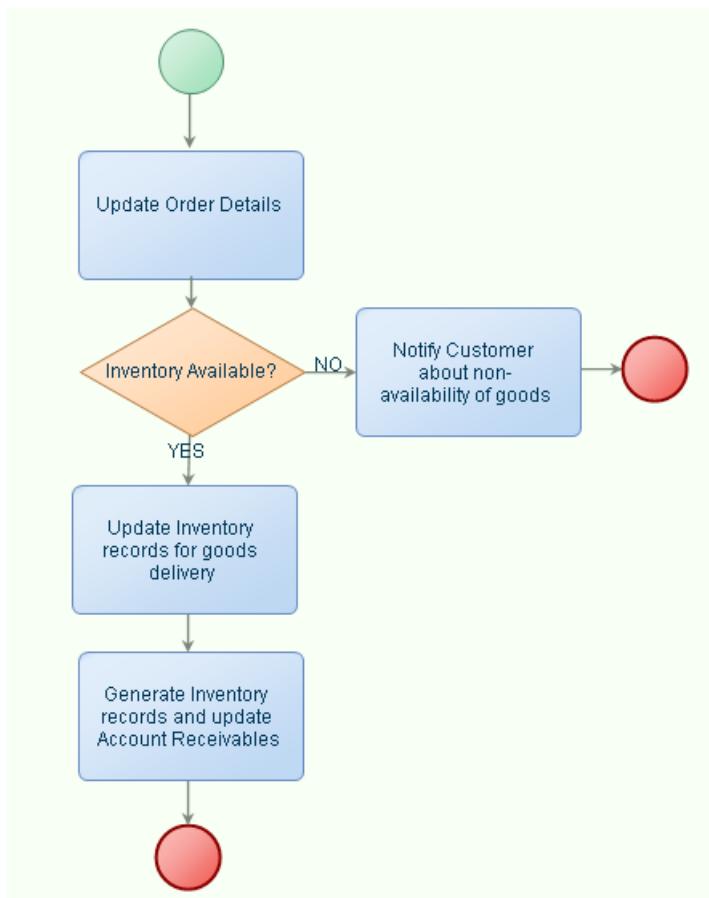
To model a transaction:

1. To design your business process model you may do any one of the following:
 - [Select a starting point](#) and click  (Business Process Model) to open an existing business process mode.
 - If you have the business process model already opened, then perform Step 2.
2. To group multiple activities as a transaction, right-click in the business process modeler and select **Group as > Transaction**. Alternatively, press Shift key to select required activities, right-click and select Group as > Transaction.
The selected activities are grouped as a transaction.

The following example describes the procedure to model a transaction in a business process model to check units in stock for each product, generate an invoice, and update Accounts Receivable.

To satisfy the business process model requirements:

1. An order placed by a customer through an order entry system should trigger the transaction.
2. The business process model should update the details of the order.
3. The business process model should check if inventory is available to process the order.
4. If inventory is not available, a notification should inform the customer that the request cannot be processed.
5. If inventory exists, it should be updated for delivery of goods.
6. An invoice must be generated and details of Accounts Receivable should be updated.



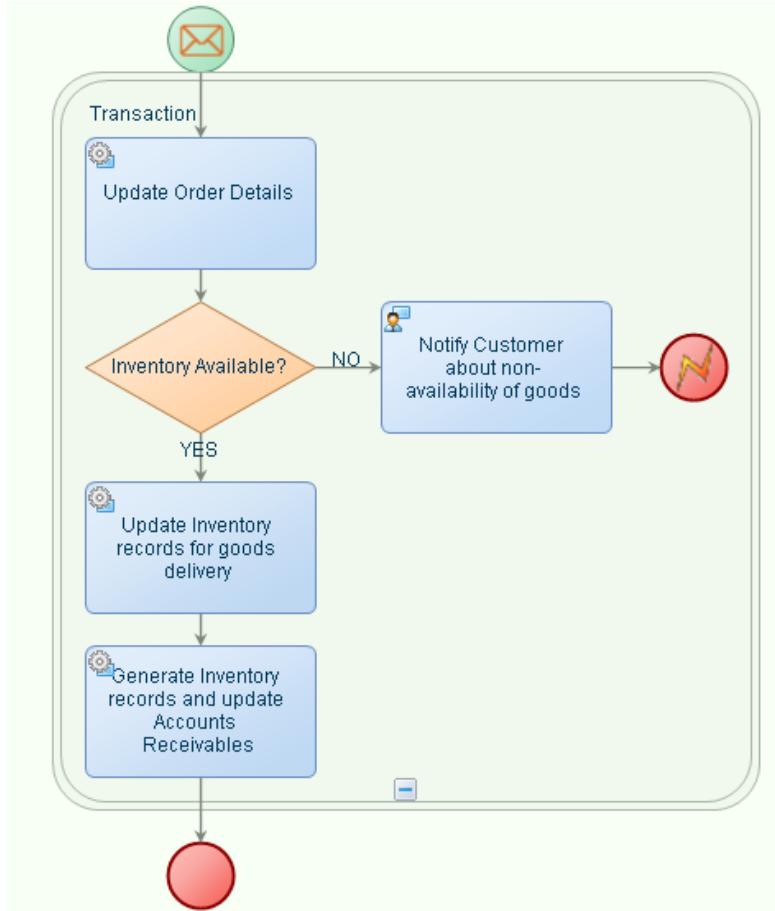
7. Create a business process model using the listed requirements.
 - a. [Create Web Service Interface on WS-AppServer Models](#) for the following activities:
 - Update order details
 - Update inventory for goods delivery
 - Generate inventory and update Accounts Receivable
8. Drag the Notify Application Service from the Workspace Documents > <Project> on to the business process model.

To model the transaction:

You must next model the transaction in the business process model and complete the Message Map for each activity in the business process model.

9. Drag the GetOrdersInput message to the Start event.
The Trigger type selected is Message in the Properties - Start Event pane. This message triggers the transaction.
10. Drag the appropriate WS-AppServer method to the corresponding activity in the business process model.

11. Select the activities that you want to group in the transaction; right-click and select **Group as > Transaction**.
12. Double-click the transaction outline.
Alternatively, right-click the transaction outline and select Properties.|
The Properties - Transaction pane appears.
13. Click the **Data Source** tab to ensure that **SOAP Processor - Find one automatically** is displayed by default in the text box beside Read Data Source from.
The **Participate in Transaction** option (available for WS-AppServer methods that are grouped in a transaction) is selected by default.
14. Double-click the End event.
Alternatively, right-click the End event and select Properties.
The Properties - End Event pane appears.
15. Select **Error** as the End Type.
The business process model appears as follows.



16. Right-click in the business process modeler and select **Business Process Execution > Validate and generate BPML**.
Warnings if any, appear in the Validate pane. Resolve the warnings appropriately.

17. Right-click in the business process modeler and select **Business Process Execution > Publish to Organization**.
The business process model is published to CoBOC.
18. Right-click in the business process modeler and select **Business Process Execution > Debug** to verify that the business process model is executed properly.

In this example, when a customer places an order through an order entry system, the entire transaction is triggered. The business process model first updates details of the order placed using the input parameters given in the Message Map. Next, the business process model checks if inventory is available for the order placed. If there is no inventory available, a notification is sent to the customer informing him or her that the order could not be processed and an error message is sent. If inventory is available, the inventory is updated to deliver goods to the customer. Finally, the invoice is generated and the Accounts Receivable is updated. If any Activity, for example, the Updating of Inventory or Updating of Accounts Receivable fails, the entire transaction is rolled back.

Data modeling reference

This section contains the following topics:

- [Rule actions](#)
- [Rule Engine Function Library](#)
- [XSLT functions](#)

Rule elements

The following elements are used to define rule behavior.

Condition

A condition element is an expression that evaluates whether a condition is true or false. It is the 'if' part of a rule. The condition consists of an expression defined on the elements of an object's template. A condition can consist of multiple conditions connected by logical operators (AND, OR, etc.), which should all be part of a single expression. You can either drag the required attributes or operators from the function library and the object templates onto the condition section to build the condition expression, or type the complete expression, or do a combination of both.

Example of a Condition

Based on the example in the precondition section, suppose you want to design a rule that will give a discount on the order based on the customer's rating. If the customer's rating is 'A', then a discount of 15% should be given. Accordingly, a condition can be formulated, which makes the necessary check. The condition would be as follows:

```
Cust:Customer/Cust:Rating = "A"
```

Action

The action element is the 'then' part of a rule, which specifies what needs to be done when the condition is evaluated as true or false. Rule engine offers you a host of action types that you can use to build simple as well as complex rules. It allows you to use nested actions in a rule.

Example of an Action

If the customer's rating is 'A', then a discount of 15% should be given. If the customer's rating is other than 'A', a discount of 2% should be given. Accordingly, a condition can be formulated, which makes the necessary check. Based on the condition, the action (the 'then' and the 'else' part of the rule) would be expressed as follows:

```
Cust:Customer/Cust:Rating = "A" then Business Object Attribute -  
Ord:Order/Ord:Discount Value - 15  
else Business Object Attribute - Ord:Order/Ord:Discount Value - 2
```

Rule actions

The rule engine provides an array of rule to develop simple as well as complex rules. These rule actions are applicable while [creating rules](#) or [decision tables](#). The following table lists the various types of actions a rule can perform.

Types of Rule Actions	Rule Actions	Description
External Rule actions enable you to invoke functions like triggering processes, sending notifications, running methods, and so on.	Trigger Business Process	You can use this rule action to trigger a process using a rule. You will need to specify the name of the process and the message to be sent to the process.
	Invoke Web Service	You can use this rule action to invoke a Web service.
	Send Notification	You can use this rule action to send a notification to one or more users or roles.
Assign Rule action enables you to modify the value of a particular attribute of an object or assign a value to an attribute.	Assign	You can use this rule action to assign a value to an attribute.
Abort Transaction Rule action allows you to abort a CoBOC transaction.	Abort Transaction	You can use this rule action to abort the CoBOC transaction for the operation that has resulted in the execution of this rule, and

Types of Rule Actions	Rule Actions	Description
		compose an abort message to be sent back as a fault.

Using trigger business process action

This task applies while defining the action of a rule or a [decision table](#). You perform this action when you want to trigger a business process based on certain conditions. For instance, you may want to trigger a Re-order business process whenever the stock in hand reaches a certain level.

Note: When using Trigger Business Process action in a decision table, perform Step 3 and Step 4.

To use trigger business process action:

1. In the Rule Definition pane, right-click **then** > **Actions** > **Trigger Business Process**.
2. Provide an **Action Name**.
For more information on naming the action, see Guidelines for Naming Rule Actions in the *AppWorks Platform Administrator's Guide*.
3. Depending on the way the rule modeler is accessed, do any of the following:
 - If you accessed the rule modeler from Workspace Explorer, drag the required Business Process from the Workspace Explorer to the Message box in the Action - Trigger Business Process box.
 - If you accessed the rule modeler from My Documents:
 - Click  on the rule modeler toolbar.
A Quick Access Menu is displayed.
 - Click  Insert on the sidebar.
All the related artifacts are displayed on the Quick Access Menu
 - Drag the required business process from the Quick Access Menu to the Message box in the Action - Trigger Business Process box.
The name of the process is displayed in the Process Name field and the input message of the selected process is displayed in the Message box
4. Specify the required parameters in the input message to be sent to the process in the message box.

The type of action(s) along with the name is displayed after then in the Rule Definition pane. Repeat the task to add more business processes to the rule.

Using Invoke Web Service action

This task applies while defining the action of a rule or a decision table.

Important: For using Invoke Web service action in a decision table, perform Step 3 and Step 4.

To use Invoke Web Service action:

1. In the Rule Definition pane, right-click then select **Actions > Invoke Web service**.
2. Provide an **Action Name**.
For more information on naming the action, see Guidelines for Naming Rule Actions.
3. Depending on the way the rule modeler is accessed, do the following:
 - If you accessed the rule modeler from Workspace Explorer, drag the required Web service from the Workspace Explorer to the Request box in the Action - Invoke Web service box.
 - If you accessed the rule modeler from My Documents:
 - Click  on the rule modeler tool bar.
A Quick Access Menu is displayed.
 - Click  Insert on the sidebar.
All the related artifacts are displayed on the Quick Access Menu.
 - Drag the required Web service from the Quick Access Menu to the Request box in the Action - Invoke Web service box.
The name of the Web service is displayed in the Method Name field and the request based on the WSDL is displayed in the Request text box.

When attaching an external service, you can view the header details of the method by clicking Header tab, if required.

4. Click **Add**.

The type of action along with the name is displayed after then in the Rule Definition pane. Repeat the task to add more than one Web service actions to the rule.

Using Send Notification action

This task applies while defining the action of a rule, a [decision table](#), or an action template. Depending upon the validations/conditions you set on the schema, you can send notifications to a particular role apprising him of these validations.

Important: For using Send Notification action in a decision table or an action template, perform Steps 3 to Step 6.

To use Send Notification action:

1. In the Rule Definition pane, right-click then select **Actions > Send Notification**.
An Action - Send Notification text box is displayed.
2. Provide an **Action Name**.
For more information on naming the action, see Guidelines for naming rule actions.
3. To select the role, do the following:
 - a. Click  next to Roles.
The Select Roles dialog box opens displaying All Roles List.

- b. Select the role/s and click . The selected users are displayed in Inbox Recipients.
 - c. Click **OK**. The selected roles are displayed in the Roles text box separated by semi-colons.
4. Type the URL of the application or Web page you want to load in **URL to Load**. For example, if the application is using XForm, type the URL of the XForm as `/cordys/Orderdetails.caf`. When the user opens the message in the Inbox, this XForm or HTM page is displayed.
5. Type a **Description** for the title of the URL page.
6. In the Message text box, provide the data to be displayed in the application, in XML format. Alternatively, if you want to use the business object attributes, you can click Insert tab on the editor, select Schema Fragment and drag the required attributes onto the Message text box.
- The procedure to display the message data in the Notification that is sent to the user's Inbox is explained with the help of the following example. Let us assume that there is an Order object with OrderID and OrderQty as elements. When the rule is executed, the notification that is sent to the user opens an application that contains the OrderId and OrderQty fields. To display the data in these fields, do the following:
- a. Provide the OrderID and OrderQty elements in the Message text box. Alternatively, you can drag the OrderID and OrderQty elements from the Schema Fragment section of the Insert tab on the editor, to the corresponding elements in the Messagetext box. The sample message is as shown:

```
<Order>
  <OrderID>
    <path>Order/OrderID</path>
  </OrderID>
  <OrderQty>
    <path>Order/OrderQty</path>
  </OrderQty>
</Order>
```

To use the business attribute values in the input message, the Order ID and OrderQty elements of the business object must be enclosed in `<path>` tags.

When the rule is successfully executed, this notification message is sent to the user's Inbox.

- b. To display the values in the message when the user opens the notification in the Inbox, do the following:
 - On initialization of the application, access the input message data using the `application.event.data` property, as shown:

```
// namespace for xpath expression to select the input message
var namespaces =
{wfl:"http://schemas.cordys.com/notification/workflow/1.0"};
// select the parent node of the input message data
var messageData = cordys.selectXMLNode(application.event.data,
".//wfl:Order", namespaces);
// get the actual values
var OrderID = cordys.getNodeText(messageData, ".//wfl:OrderID", null,
namespaces);
var OrderQty = cordys.getNodeText(messageData, ".//wfl:OrderQty", null,
namespaces);
```

- Bind this data to the corresponding fields in your application.
7. Click **Add**.
The type of action along with the name is displayed next to then in the Rule Definition pane. Repeat the task to add more than one Inbox action to the rule.

The action is set on the object and on successful validation of the object, a notification is sent to the Inbox of the assigned role with the specified message.

Using Fire Rule action

This task is part of defining the action of a rule.

To use Fire Rule action:

1. In the Rule Definition pane, right-click and select **Actions > Fire Rule**.
An Action-Fire Rule text box opens.
2. Provide an **Action Name**.
For more information on naming the action, see Guidelines for Naming Rule Actions.
3. From **Select a rule**, select a rule.
4. Click **Add**.
Repeat the task to add more than one fire rule action to the rule. The type of action(s) along with the name is displayed in the Rule Definition pane.

Using Abort Transaction action

This task applies while defining the action of a rule or a decision table.

To use Abort Transaction action:

1. In the Rule Definition pane, right-click and select **Actions > Abort Transaction**.
2. Provide an **Action Name**.
For more information on naming the action, see Guidelines for Naming Rule Actions.
3. In the box below Action Name, provide the abort message.
The following table describes the types of abort messages.

Static Abort Messages	Static abort messages are strings delimited by double quotes. For example ("Error Occurred ").
Dynamic abort messages	Dynamic abort messages allow you to include values of runtime variables in the abort message. In the above example, the message is delimited with double-quotes (" "), and the variables should be concatenated with the abort message. For Example: concat("Error occurred due to Invalid OrderId:", Order/OrderId)

4. Click **Add**.
The type of action along with the name is displayed after then in the Rule Definition pane.
5. To add more actions, repeat the task.

Using Assign action

This task applies while defining the action of a rule or a decision table.

For using Assign action in a decision table, perform Step 3 and Step 4.

To use Assign action:

1. In the Rule Definition pane, right-click and select **Actions > Assign**.
An Action - Assign text box is displayed.
 2. Provide an **Action Name**.
 3. Drag the required attributes from the Business Object pane to the Business Object Attribute column and type the values to be assigned to these attributes in the Value column.
- The following table describes the types of values that can be assigned in the Value column.

Literal	You can hard-code a value to be assigned; it must be enclosed either between 'single quotes' or "double quotes". For example, a string 'To be Processed'.
Parameter	You can assign a parameter so that the value is assigned during runtime; for example, prod:Product/prod:Name. The value of the Name element will be assigned at runtime.

While defining the assignment action in a decision table, you can drag the root node of the business object on to the Properties pane of the decision table. This will enable you to assign values to multiple attributes.

4. Click **Add**.

5. Repeat the task to add more assignment actions to a rule.

The type of action(s) along with the name is displayed after then in the Rule Definition pane.

Rule engine function library

The rule engine in AppWorks Platform uses the following library functions.

- [Mathematical Functions of Rule Engine](#)
- [String Functions of Rule Engine](#)
- [Date and Time Functions of Rule Engine](#)
- [Operators and Constants](#)
- [Sprintf Function Library](#)

String functions of rule engine

String functions can be used to manipulate string values. You can convert strings to integers, return length of the strings, extract substrings, and remove leading and trailing spaces.

The following table describes the various string functions available in the rule engine function library and their usage.

String functions	Description	Usage
asc(e1)	Returns the ASCII code for a specified character.	asc(e1), where e1 is the string representation of the character for which you want the ASCII code. For Example: asc('A')- returns the value 65.
chr(e1)	Returns the character represented by the given integer value.	chr(e1), where e1 is a numeric expression representing the ASCII value of a character. For example: chr(65)- returns the value A.
concat(e1,e2)	Concatenates the given string parameters.	concat(e1,e2), where e1 and e2 are string expressions which you want to join.
contains(e1, e2)	Returns true if the first argument string contains the second argument string, and otherwise returns false	contains(e1, e2), where e1 and e2 are strings, and returns true if the string e1 contains the string e2. For example: contains('Arnold', 'n'). The result of the above function is true.
normalize-space (' e1 ')	Removes the trailing spaces from the input string	normalize-space(' e1 '), where e1 is the string expression or string literal from

String functions	Description	Usage
	expression.	which you want to remove the trailing spaces. For example: normalize-space('Arnold '). The result of the function is 'Arnold'.
number(e)	Converts a string expression into an integer.	number(e), where e is a string expression that you want to convert into an integer.
pos(e1,e2)	Returns the starting position (first index) of a specified string within the source string.	<p>pos(e1,e2), where: e1 is the source string expression or string literal on which the pos function will perform the search. e2 is a part of the string, or the target string expression, or the string literal to be searched in the source string (e1). For example: pos('hello','e')- returns the value 2.</p> <p>Note: If the pos function does not find a match of e2 in e1, it returns an integer value of 0.</p>
rpos(e1,e2)	Calculates the starting position of a specified string from its rightmost occurrence within the source string.	<p>rpos(e1,e2), where: e1 is the source string expression or string literal on which the pos function will perform the search. e2 is a part of the string, or the target string expression, or the string literal to be searched in the source string (e1). For example: rpos('racecar', 'c')- returns the value 5.</p> <p>Note: If the rpos function does not find a match of e2 in e1, it returns an integer value of 0.</p>
starts-with(e1, e2)	Returns true if the first argument string starts with the second argument string, and otherwise returns false	<p>starts-with(e1, e2), where e1 and e2 are strings, and returns true if the string e1 starts with the string e2. For example: starts-with('Arnold', 'A'). The result of the function is true.</p>
string (e)	Converts a numeric expression into a string.	string(e), where e is a numeric expression that you want to convert into a string.

String functions	Description	Usage
string-length (e1)	Returns the number of characters in a given string.	string-length (e1), where e1 is the string expression or string literal to calculate the length of the parameter.
substring (e1,x,y)	Returns the substring from the start position to the specified length. Index of the first character is 1. If length is omitted it returns the substring from the start position to the end	substring(e1,x,y), where e1 is the string, x is an integer that indicates the start position, y is an integer that indicates the length. For example: substring('Arnold', 2,3). The result of the function is 'rno'.
substring-after (e1, e2)	Returns the remainder of string1 after string2 occurs in it	substring-after(e1, e2), where e1 and e2 are strings. For example: substring-after('Arnold', 'n'). The result of the function is 'old'.
substring-before (e1, e2)	Returns the start of string1 before string2 occurs in it	substring-before(e1, e2), where e1 and e2 are strings. For example: substring-before('Arnold', 'o'). The result of the function is 'Arn'.

Mathematical functions of rule engine

Mathematical functions can be used to perform mathematical operations on data, following the XPath Notations. Apart from basic mathematical operations such as addition, subtraction, and so on, you can use functions to round off numbers to the nearest integer, and find the sum of all the numeric values in a nodeset.

The following table describes the various mathematical functions available in the rule engine function library and their usage.

Mathematical Functions	Description	Usage
sum(e)	Returns the sum of the numeric value of each node in the specified node-set	sum(e), where e is the nodeset. For example: sum(Item/Quantity) returns the sum of all the numeric values of the Quantity nodeset in the Item object.
floor(e)	Returns the largest integer that is not greater than the number argument	floor(e), where e is a numeric expression. For example: floor(44.12) returns the value 44.

Mathematical Functions	Description	Usage
ceiling(e)	Returns the smallest integer that is greater than the number argument	ceiling(e), where e is a numeric expression. For example: ceiling(44.12) returns the value 45.
round(e)	Rounds the number argument to the nearest integer	round(e), where e is a numeric expression. For example: round(44.12) returns the value 44.

Date and time functions of rule engine

Date and time functions are used to manipulate the date and time data of a business object.

The following date-time functions are used to parse a string to a date:

- YYYY-MM-DD
- T HH:MI:SS
- YYYY/MM/DD
- T HH:MI:SS

Note

- If you do not specify any arguments, the function is processed based on the present date and time.
- If you specify a value that is out of the acceptable range, the function returns -1.

The following table describes the elements in the date-time format.

Element	Description
YYYY	Year
MM	Month
DD	Day
T	Delimiter to separate the date and time
HH	Hours
MI	Minutes
SS	Seconds

The following table describes the date and time functions, and their usage.

Date Functions	Syntax	Parameters
date returns the number of days from the first of January 1 A.D. to the date you specified.	date(year, month, day)	<ul style="list-style-type: none"> year should be an integer that can be greater than or equal to 1. month should be an integer that is between 1 (representing January) and 12 (representing December). day should be an integer between 1 and 31. Example: date(2007,06,22). If date is given without parameters, it takes the current date as the parameter.
	date(e)	e is a string expression that should contain the date in the expected date-time format. Example: date("2007-06-22T00:00:00")
utc returns the number of seconds that elapsed from January 1, 1970, 00:00:00, Universal Coordinated Time to the specified date.	utc(year, month, day, hours, minutes, seconds)	For year specify the full year to ensure cross-century accuracy. If you represent a year with a number between 0 and 99, then the year would be 1900 + year.
	utc(e)	<ul style="list-style-type: none"> month should be an integer that is between 1 (representing January) and 12 (representing December). day should be an integer between 1 and 31. T represents the delimiter to separate the date and time. hour should be an integer that can be between 0 (representing midnight) and 23 (representing 11 p.m.). minute should be an integer between 0 and 59. second should be an integer between 0 and 59. Example: utc (2007,06,22,11,46,10) e is a string expression that should contain the date in the expected date-time format.

Date Functions	Syntax	Parameters
		Example: <code>utc("2007-06-22T11:46:10")</code>
utcgmt returns the number of seconds that elapsed from January 1, 1970, 00:00:00, Universal Coordinated Time to the current GMT.	<code>ustgmt()</code> - does not take any parameters as arguments.	Example: The value returned by <code>utcgmt()</code> when the date and time values are 2007-06-22 and 01:51:27 respectively is 1280285487 seconds.

To customize the date format such that the output (date) is derived in different numeric and text formats, you can use `sprintf` functions. For more information on `sprintf` functions, refer to [Sprintf Functions for Formatting Dates](#).

To customize the time format such that the time is displayed in either the 12-hr format or the 24-hr format, see [Sprintf Function for Formatting Time](#)

Operators and constants in function library

The Function Library in the rule creator provides a set of functions and operators that can be used for framing simple as well as complex rules. The following sections describe the operators and constants that are provided by the function library.

Arithmetic operators

Arithmetic operators allow you to define mathematical operations in rule expressions.

The following table describes the arithmetic operators you can select from the function library area while creating a rule.

Operator	Description	Syntax
+	Adds e1 and e2	<code>e1 + e2</code>
	Subtracts e2 from e1	<code>e1 - e2</code>
	Multiplies e1 and e2	<code>e1 * e2</code>
div	Divides e1 and e2 (If e2 is 0, then the expression evaluates to 0.)	<code>e1 div e2</code>
mod	Gives the modulus (division remainder) of e1 and e2	<code>e1 mod e2</code>

Boolean operators

Boolean operators evaluate expressions to true or false. The result `true` takes the integer value 1, while `false` takes the value 0.

Operator	Description	Syntax
and	Logical and operator	e1 and e2
or	Logical or operator	e1 or e2
=	Equality operator	e1 = e2
!=	Inequality operator	e1 != e2
<	Less than operator	e1 < e2
<=	Less than or equal to operator	e1 <= e2
>	Greater than operator	e1 > e2
>=	Greater than or equal to operator	e1 >= e2

Boolean functions

The rule engine provides a set of Boolean functions that can be used directly in a rule.

Constant	Description	Value
true()	function call for returning true	1
false()	function call for returning false	0

Operator precedence and associativity

Operator precedence describes the order in which the rule engine reads expressions. Operators are arranged in descending order of precedence, that is, from the highest to the lowest. More than one operator appearing in a single row has the same precedence.

The following table lists the operators in the order in which the rule engine reads them. The associativity column specifies the direction in which each operator is read.

Operator	Description	Associativity
in	Logical in operator	Left to right
+ -	Unary plus/minus operators	Right to left
^	Power operator	Left to right
div Mod	Multiplication, division, modulus operators	Left to right
+ -	Addition/subtraction operators	Left to right
< <=	Relational less than/less than or equal to operators	No Associativity
> >=	Relational greater than/greater than or equal to operators	
= !=	Relational is equal to/is not equal to operators	

Operator	Description	Associativity
and	Logical and operator	Left to right
or	Logical or operator	Left to right

Conditional expression function

AppWorks Platform rule engine supports conditional expression (apart from XPath 1.0 support) based on the keywords if, then, and else.

Expression	Description
if (testExp) then (thenExp) else (elseExp)	If the effective Boolean value of the <ul style="list-style-type: none"> ■ testExp is true, the value of the thenExp is returned. ■ If the effective Boolean value of the test expression is false, the value of the elseExp is returned. For example: if (/Customers/Country ="UK") then (/Customers/City) else (/Customers/Country)

Sprintf function library

For the data output to conform to various business applications and locales, it needs to be presented in various formats. Data formatting options are primarily used to ensure the validity and applicability of data across different platforms, and in various environments. While cross-platform consistency is bound to be an irresolvable issue, data formatting provides a simple workaround for enhancing the acceptability of the data across various processing environments.

Using the sprintf functions, you can make data versatile. These functions help you format various data parameters such as the date, time, and currency, besides the generic features of the output data, to suit diverse data processing environments.

The following are the sprintf functions available in the rule engine function library:

- [Sprintf Function for Formatting Generic Data](#)
- [Sprintf Function for Formatting Currency](#)
- [Sprintf Function for Formatting Dates](#)
- [Sprintf Function for Formatting Time](#)

Sprintf function for formatting dates

You can use the sprintf function to customize the date format such that the output (date) is derived in different numeric and text formats to suit the application requirements.

Format	Description
%d	Day of the month represented by a number between 1 and 31
%dd	Day of the month represented by a number between 01 and 31
%ddd	Name of the day abbreviated to three letters in user's locale
%dddd	Full name of the day in user's locale
%M	Month represented by a number between 1 and 12
%MM	Month represented by a number between 01 and 12
%MMM	Name of the month abbreviated to three letters in user's locale
%MMMM	Full name of the month in user's locale
%y	Year represented by a number between 0 and 99
%yy	Year represented by a number between 00 and 99
%yyyy	Year represented by a four-digit number

The `sprintf` function for formatting dates can be used in the following ways.

```
sprintf("%D(format)", UTCTime)
```

```
sprintf("%D(format)", UTCTime)
```

This function expects the `CordysDateFormat` to be between 1970-01-01T00:00:01.0 GMT and 2038-01-19T03:14:08.0 GMT.

For example, if the `CordysDateFormat` is given as 1970-01-01T00:00:01.0 GMT, the function returns 1970-Jan-01. Any time beyond these boundaries returns the value as -1.

```
sprintf("%D(format)T%T(format)", UTCTime, UTCTime)
```

Function	Expected Output
<code>sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%omm:%ss.0)", 0, 0)</code>	Returns the reference date and the reference time - 1970-01-01T05:30:00.0
<code>sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%omm:%ss.0)", 0, utc())</code>	Returns the reference date and current locale time - 1970-01-01T00:00:01.0
<code>sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%omm:%ss.0)", utc(),0)</code>	Returns the current locale date and the reference time - 2009-01-01T05:30:00.0
<code>sprintf("%D(%yyyy-%MM-%dd)T%T(%HH:%omm:%ss.0)", utc(),utc())</code>	Returns the current locale date and the current locale time - 2009-01-01T00:00:01.0

Function	Expected Output
sprintf("%D(%yyyy-%MM-%dd)T%T (%HH:%mm:%ss.0)", utc(),utcgmt())	Returns the current locale date and the current GMT time - 2008-12-31T18:30:00.0
sprintf("%D(%yyyy-%MM-%dd)T%T (%HH:%mm:%ss.0)", utcgmt(),utcgmt())	Returns both current date and time in GMT format - 2008-12-31T18:30:00.0

The following are the examples of the output generated by this function:

- `sprintf ("%D (%yyyy-%MM-%dd) ", 0)` displays the date as 1970-01-01
- `sprintf ("%D (%yyyy-%MMM-%dd) ", 0)` displays the date as 1970-Jan-01
- `sprintf ("%XD (%yyyy-%MMM-%dd) ","2009-01-01T00:00:01.0")` displays the date as 2009-Jan-01.

To display the current date and time in CordysDateFormat:

```
sprintf ("%D (%yyyy-%MM-%dd) T%T (%HH:%mm:%ss.0)", utc(),utc())
```

The following is the example of the output generated by this function:

`sprintf ("%D (%yyyy-%MM-%dd) T%T (%HH:%mm:%ss.0)", 0,0)` displays the date and time as 1970-01-01T05:30:00.0

Sprintf function for formatting time

You can use the sprintf function to customize the time format such that the time is displayed in either the 12-hr format or the 24-hr format.

The following table lists the formats supported by the sprintf function for formatting time.

Format	Description
%H	Hour in the 24-hour clock represented by a number between 0 and 23
%HH	Hour in the 24-hour clock represented by a number between 00 and 23
%h	Hour in the 12-hour clock represented by a number between 1 and 12
%hh	Hour in the 12-hour clock represented by a number between 01 and 12
%m	Minutes represented by a number between 0 and 59
%mm	Minutes represented by a number between 00 and 59
%s	Seconds represented by a number between 0 and 59
%ss	Seconds represented by a number between 00 and 59
%t	A or P (in user's locale) respectively for ante-meridian and post-meridian
%tt	AM or PM respectively for ante-meridian and post-meridian

The sprintf function for formatting time can be used in the following ways:

```
sprintf("%T(format)", UTCTime)
```

```
sprintf("%XT(format)", CordysDateFormat)
```

The following are the examples of the output generated by various combinations of this function when the locale is Asia (GMT+5.30):

- `sprintf("%T(%HH:%mm:%ss)", 0)` displays the time as 05:30:00
- `sprintf("%T(%HH:%mm:%ss %tt)", 0)` displays the time as 05:30:00 AM
- `sprintf("%XT(%HH:%mm:%ss %tt)", "2008-01-01T00:00:01")` displays the time as 00:00:01 AM
- `sprintf("%T(%HH:%mm:%ss %tt)", utc())` displays the current locale time as 00:00:01 AM
- `sprintf("%T(%HH:%mm:%ss %tt)", utcgmt())` displays the current GMT time as 06:30:00 PM

Sprintf function for formatting currency

You can use the sprintf function to customize the currency format such that both the value and the denomination are displayed in various formats.

The following table lists the formats supported by the sprintf function for formatting currency.

Format	Description
%dd.dd	Currency value including a decimal separator
%g(3,2)	Sets off two digits to the left of the decimal separator and then every third digit. The output, for example, could be 1,123,123,12.00.
%g(3)	Sets off digits to the left of the decimal separator into groups of three. The output, for example could be: 1,123,123,123.00.
%s(.)	Specifies period (.) to be the decimal separator
%S(,)	Specifies comma (,) to be the thousands separator
%n(-\$n)	Specifies the negative currency mode where \$ denotes the currency symbol and n the currency value
%p(\$n)	Specifies the positive currency mode where \$ denotes the currency symbol and n the currency value
%c(\$)	Specifies \$ to be the currency symbol

The sprintf function for formatting currency can be used in the following way:

```
sprintf("%C(format)", Amount)
```

- `sprintf("%C(%ddd.dd%g(3,3))", 123123123)` returns the currency as 123,123,123.00
- `sprintf("%C(%g(3,2)%s(.))", 123123123)` returns the currency as 1,231,231,23.00

Sprintf function for formatting generic data

You can use the sprintf function to customize generic data to produce the output in any form such as figures, digits, exponential form, numeric representation in non-decimal form, and so on.

The following table lists the formats supported by the sprintf function for formatting generic data.

Format	Description
%d, %I	Signed integer values
%e	Doubles with exponent, with 2 or 3 positions for exponent. For example: 1.23e+02
%E	Similar to e. For example, 1.23E+02
%f	Doubles to 6 decimal places. For example, 32.123456
%g	Double values; %e or %g, whichever is more compact, is used
%G	Double values; %E or %g, whichever is more compact, is used
%s	Strings
%o	Octal representation of unsigned integers
%x	Hexadecimal representation of unsigned integers. For example, 374fff
%X	Hexadecimal representation using uppercase for A-F. For example, 374FFF
%XD	Date format that uses XML string as input
%XT	Time format that uses XML string as input
%u	Unsigned integer
%l	The locale

The locale, %l, cannot be specified in conjunction with other formats. When you set the locale, all subsequent formatting for date, time, and currency values will be in that locale only. Therefore, locale must be installed on your system. Otherwise, the default Posix notation (for example, en_US, nl_NL) is used by the system.

Using Java in rules

Rules are highly configurable and offer you a host of options to suit business specifications. A powerful rule engine deals with the evaluation and execution of every rule. It provides a set of built-in data types to provide flexibility in defining complex rules. Apart from using the default functions (mathematical functions, string functions, and so on) to build a rule,

you can extend and enhance your rules by using Java constructs from within a rule definition.

The Rule Engine supports a framework that provides:

- Invoking a Java method and assigning Java expression results
- Passing a NOM nodeset to a Java method

XPath and XSLT

XML is an excellent medium for packaging and exchanging data. An XML document is a flat text file that contains data in a structured format. Usually, this information is retrieved by making a tree representation of the XML document.

An XML document tree is made up of nodes. Different types of nodes that can constitute an XML document are:

- Root nodes
- Element nodes
- Text nodes
- Attribute nodes
- Comment nodes
- Processing instruction nodes
- Namespace nodes

Extensible Stylesheet Language Transformations (XSLT) is an XML-based language used for the transformation of XML documents into other XML or documents that can be read by human beings. For more information on XSLT, see [Understanding XSLT](#).

XPath is a language for addressing parts of an XML document. It is designed to be used by XSLT. For more information on XPath, see [Understanding XPath](#).

For best practices on using XPath, see [Optimizing XPath Expression Usage](#).

Though AppWorks Platform supports most of the features provided by XSLT, some features are not supported. For the list of features/specs that are not supported in AppWorks Platform, see [XSLT Features that not supported in AppWorks Platform](#) and a few [limitations](#) of XSLT in AppWorks Platform.

Overview of XSLT

Understanding XSLT

EXtensible Stylesheet Language (XSL) is a style sheet language for XML. XSL Transformations (XSLT) is a language for transforming XML documents into other XML documents. XSLT is designed for use as part of XSL. An XSLT style sheet specifies the presentation of a class of XML documents by describing the way an instance of the class is transformed into an XML document that uses a formatting vocabulary, such as (X)HTML or XSL-FO.

XSLT can be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. It is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL.

XSLT and XPath are closely linked together. For more information on XPath, refer to XPath. XSLT uses XPath to navigate through the documents that must be transformed. The prefix 'xsl:' is used for XSLT namespace. For more information on XSL, see <http://www.w3.org/1999/XSL/Transform>.

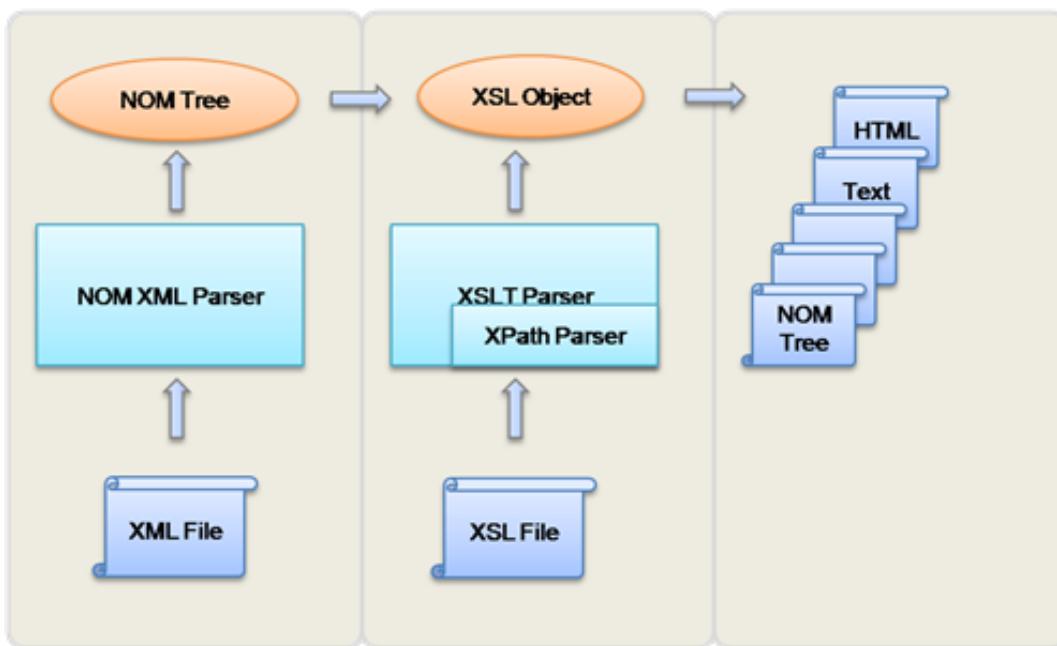
Benefits of XSLT

Through XSLT, you can present XML documents or data to the user in an attractive way in browser, mobile phone, or PDF format, by converting the original data to the necessary output format. The advantage of using XSLT lies in its built-in XPath functionality that fragments any XML document into several sections to facilitate easier transformation.

XSLT architecture

Architecture of XSLT

The XML file that needs to be transformed is converted into a NOM tree after being parsed through the NOM XML parser. Simultaneously, using the XPath parser, the XSLT parser generates an XSL object based on the XSL document provided. For more information on XPath parser, see [Parsing XPath expressions](#). At the point of transformation, the XSL object is used to transform the NOM tree into various formats such as HTML, text, or any other NOM tree.



Requirements of XSLT

XSLT requires the following:

- A style sheet describing transformation rules
- A transformation rule consists of a pattern and a template
- A pattern is a configuration in the source tree
- A template is a structure to be instantiated in the result tree

To understand how XSLT operates, consider the following example:

```
<xsl:template match="Title">
  <H1>
    <xsl:apply-templates/>
  </H1>
</xsl:template>
```

Where,

Input is<Title>Introduction</Title>

Output is<H1>Introduction</H1>

XSLT Extensions

If the specialized elements of the XSLT namespace are not enough to perform the transformations you need, XSLT provides you with many ways to incorporate additional instruction elements and functions into your stylesheet. Using XSLT is in turn using the various elements from the XSLT namespace such as `xsl:template`, `xsl:apply-templates`, and `xsl:output`. XSLT elements such as `xsl:apply-templates`, `xsl:text`, and `xsl:element` that tell the XSLT processor to add something to the result tree, are called instructions. There are also 'top-level' elements such as `xsl:output` and `xsl:strip-space` that give more general instructions to the XSLT processor about how to perform the transformation.

As per the XSLT 1.0 specification, there are two types of extensions:

- Elements
- Functions

You can define the extension element by defining namespace and including a value of 'extension-element-prefixes' attribute of style sheet element or literal result element as the prefix. For function call, if the function name has a prefix and if it is mapped to a namespace, it is considered as an extension function.

NOM Specific Extension

Although the XSLT extensions help in transforming XML documents, they do not cater to all the transformation requirements. Therefore, while using Java to transform XML documents in a NOM environment, the NOM XSLT has one extension element, called `js:script`. Inside this element, you can write custom JavaScript functions and call it from XPath expression.

The following example shows a use of script:

```

<XSL:stylesheet
    xmlns:XSL="http://www.w3.org/1999/XSL/Transform"
    xmlns:ns1= "http://hello.com"
    xmlns:js = "http://www.cordys.com/XSL/script/javascript"
    extension-element-prefixes ="js"  version="1.0">
<js:script>
    function todayStr()
    {
        var today = Date();
        return today.toString();
    }
</js:script>
<XSL:template match="/">
    <XSL:variable  name ="hi">
        <hello name ={js:todayStr()}>
        </hello>
    </XSL:variable >
    <XSL:apply-templates select = "//PurchaseOrder"/>
</XSL:template>
<XSL:template match="PurchaseOrder" >
    <Invoice date = "{js:todayStr() }"/>
</XSL:template >
<XSL:stylesheet>

```

Note that script element should originate from the namespace <http://www.cordys.com/XSL/script/javascript>. Also note that the element-available() function will not return true for this element.

JavaScript

SpiderMonkey from Mozilla.org is being used as the JavaScript engine. You can pass String, Boolean, and Number as arguments to the JavaScript function. If you want to have a more complex manipulation that includes Nodeset, you must use the custom Java functions or native functions. XPath custom Java method call can return Nodeset. However, the Nodeset must include only those nodes which are part of the input NOM tree originally provided. Also, there are no JavaScript NOM APIs for NOM.

Using XSLT API

The XSLT mechanism is used in identifying, updating, and transforming specific portions of an XML document.

XSLT uses the XSL and creates an XML object that needs to be transformed to an XML format that is based on another XML schema. After using XPath to identify the XML fragment to be transformed, XSLT is used to finally transform the XML document.

To use XSLT:

1. Create an XSL object from the XSL map.
XSL map can be in the form of a string or can be inside a file.
2. Perform the mapping using the XSLT object.
3. Delete the XSLT object (if used from native layer).

To understand the procedure, consider the following example:

```
try {
    Document oDocument = new Document();
    int iXml = oDocument.load("birds.xml");
    XSLT oXSL = XSLT.parseFromFile("birds.xsl");
    if (oXSL != null) {
        String iTransformedXML = oXSL.xslTransformToString(iXml);
        oXSL.delete();
    }
} catch (XMLException e) {
    e.printStackTrace();
}
```

XSLT functions

XSLT functions can be used to manipulate data while mapping. For example, you can have two or more elements map to a function block that concatenates the values, and route the output value to the target element. In another example, you can have a function block run a Web service and assign the output returned by that Web service to a target element. In this way, a range of XSLT functions are provided to perform simple to complex operations on data.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code can use the equivalent in-house javascript functions and their wrapper implementations. If users want to use the functions as specified in the XSLT 2.0 and XPath 2.0 specifications, they can create their own transformation scripts in the Model Source. However, these will not reflect on the Map canvas in the modeler.

The following are the various functions that are available in AppWorks Platform Data Transformation modeler:

- [String functions](#)
- [Mathematical functions](#)
- [Date and time functions](#)
- [Scientific functions](#)
- [Logical functions](#)
- [Cumulative functions](#)
- [Advanced functions](#)
- [Instructions](#)

String functions

String functions can be used to manipulate string values. You can concatenate output values to a target, change the capitalization of strings, extract substrings, replace strings with new values, and remove leading spaces. Refer to [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

The following table lists the various string functions as they appear on the Map canvas of the Data Modeler supported by AppWorks Platform, their equivalent functions on the Model Source tab, their functionality, and parameters.

String Function (in Map canvas)	String function (in Model Source tab)	Description	Parameters
Concat	concat (string,string,...)	Concats two or more string values	Initial String - The string that is to be concatenated. String to Concat - The string that concatenates to the initial string. Note: One or more parameters of any data type can be specified for this function.
Lower Case	translate(string, A, a)	Converts all alphabetic characters to lower case	String - The string to be converted to lower case.
Upper Case	translate(string, a, A)	Converts all alphabetic characters to upper case	String - The string to be converted to upper case.
Replace	replace(string, pattern, replace)	Replaces the source string character by character	This function consists of the following parameters: <ul style="list-style-type: none"> ■ String - This parameter specifies the source string on which this function will be applied. ■ Character to be

String Function (in Map canvas)	String function (in Model Source tab)	Description	Parameters
			<p>Replaced - This parameter specifies the character in the source string to be replaced.</p> <ul style="list-style-type: none"> ■ Character to Replace - This parameter specifies the character to replace the character specified in the second parameter. For example: replace ("Joanne", "n", "l") returns Joalle.
Substring	substring (string,start,len) or substring (string,start)	Returns string of a specified length starting from a specified location. Index of the first character is 1. If length is omitted it return s the substring from the start position to the end	This function consists of the following parameters: <ul style="list-style-type: none"> ■ String - This parameter specifies the source string from where the substring is to be extracted. ■ Start Position of Substring - This parameter specifies the starting position of the substring that is to be extracted, from the string. ■ Length of Substring - This parameter specifies the length or number of characters to be extracted. For Example: <ul style="list-style-type: none"> • substring('Joanne',3,4) returns anne • substring('Joanne',2) returns oanne
Substring-Before	substring-before (string1,string2)	Returns the substring before a	This function consists of the following parameters:

String Function (in Map canvas)	String function (in Model Source tab)	Description	Parameters
		specified character in the string	<ul style="list-style-type: none"> ■ String - This parameter specifies the source string from where the substring is to be extracted. ■ Character Next to Substring - This parameter specifies the character next to the substring to be extracted. The function will extract the substring that is before the character specified in this parameter. For example: substring-before ('1999/04/01','/') returns 1999.
Substring-After	substring-after(string1,string2)	Returns the substring after a specified character in the string	<p>This function consists of the following parameters:</p> <ul style="list-style-type: none"> ■ String - This parameter specifies the source string from where the substring is to be extracted. ■ Character Next to Substring - This parameter specifies the string before the substring to be extracted. The function will extract the substring that is after the string specified in this parameter. <p>For example: substring-after('1999/04/01','/') returns /04/01.</p>
String Length	string-length(string) or	Returns a number equal to the	String - The string whose length is to be returned.

String Function (in Map canvas)	String function (in Model Source tab)	Description	Parameters
	string-length()	number of characters in a given string. If there is no string argument it returns the length of the string value of the current node	For example: string-length ('Joanne') returns 7.
Trim	normalize-space()	Trims leading and trailing whitespaces and replaces the sequences of whitespace characters within the string by a single space.	String - The string for which the leading and trailing spaces have to be trimmed. For example: trim(' Joanne ') returns Joanne.

Mathematical functions

Mathematical functions are used to perform mathematical operations on data during transformation. For example, if two collaborating enterprises use different units to express quantity, you can plug in a mathematical function that converts the quantity from the source to make it compatible with the target system. See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

The mathematical function block accepts only integers as valid inputs. Convert all source data into a numeric format using the Numeric function.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

The following table lists the various mathematical functions supported by AppWorks Platform as they appear on the Map canvas of the Data Modeler, the equivalent functions in the Model Source tab, their functionality, and parameters.

Mathematical Function (in Map canvas)	Mathematical function (in Model Source tab)	Description	Parameters
Integer	js:fnMathematicalInteger()	Converts the source data into an integer	Number - Specify the source element that has to be converted into an integer.
Absolute	js:fnMathematicalAbsolute()	Returns the absolute value of a number	Number - Specify the source element for which the absolute value has to be returned.
Modulo	js:fnMathModulo()	Returns the remainder as a result of a division	Number - This parameter in the function takes the dividend as input. Divisor - This parameter in the function takes the divisor as input.
Square Root	js:fnMathSqrt()	Returns the square root of a number	Number - Specify the source element for which the square root has to be returned.
Addition	(element) + (element)	Returns the sum of two or more numbers	Number to Add - This parameter in the function takes the first element to be added. Specify the additional elements to be added in the subsequent parameters.
Subtraction	(element) - (element)	Returns the difference of two numbers	Number - This parameter in the function block takes the element to be subtracted from. Number to be

Mathematical Function (in Map canvas)	Mathematical function (in Model Source tab)	Description	Parameters
			Subtracted - Specify the elements to be subtracted in this parameter.
Multiplication	(element) * (element)	Returns the product of two or more numbers	Number - This parameter in the function block takes the first element to be multiplied. Specify the additional elements to be multiplied in the subsequent parameters.
Division	(element) div (element)	Returns the result of dividing two numbers	Number - This parameter in the function takes the dividend as input. Divisor - This parameter in the function takes the divisor as input.
Rounding	round()	Rounds off a number to its nearest integer	Number - Specify the source element that has to be rounded to the nearest integer.

Date and time functions

Date and time functions are used to manipulate the date and time data of a business object during transformation. These functions play a critical role in handling transactions involving business processes that collaborate from different time zones. You can translate dates that belong to a range of predefined date formats. See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

Important

- The input parameter for Date functions must be in the following format:
YYYY-MM-DD or YYYY/MM/DD
- The input parameter for Date and Time functions must be in the following format:
YYYY-MM-DDTHH:mm:ss or YYYY-MM-DDTHH:mm:ss.ms

If the date is given in any other format, the function assumes the input to be in Current Date and processes it.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

The following table describes the various date and time functions supported by AppWorks Platform as they appear on the Map canvas of the Data Modeler, their equivalent functions on the Model Source tab, their functionality, and the parameters.

Date and Time Function (in Map canvas)	Date and Time Function (in Model source tab)	Description	Parameters
Current Date	<ul style="list-style-type: none"> ■ js:currentDateFormat1() ■ js:currentDateFormat8() ■ js:currentDateFormat2() ■ js:currentDateFormat3() ■ js:currentDateFormat4() ■ js:currentDateFormat5() ■ js:currentDateFormat10() ■ js:currentDateFormat9() ■ js:currentDateFormat6() ■ js:currentDateFormat7() 	Returns the present date in a variety of formats.	<p>The function returns the current date in one of the following formats:</p> <ul style="list-style-type: none"> ■ DD-MM-YYYY ■ DD/MM/YYYY ■ DD-MM-YYYY HH:mm ■ DD-MM-YYYY HH:mm am/pm ■ DD-MM-YYYY HH:mm:ss ■ DD-MM-YYYY HH:mm:ss am/pm ■ YYYY-MM-DD ■ YYYY-MM-DD YYYY-MM-DDTHH:mm:ss ■ YYYY-MM-DDTHH:mm:ss.ms ■ BaanDate (MMDDYYHHMMSS) - stores dates as the number of days from January 1, 1970 to the specified current date. This returns the current date and time in the

Date and Time Function (in Map canvas)	Date and Time Function (in Model source tab)	Description	Parameters
			format.
Convert Date to specific formats	<ul style="list-style-type: none"> ■ js:convertFormat1() ■ js:convertFormat8() ■ js:convertFormat2() ■ js:convertFormat3() ■ js:convertFormat4() ■ js:convertFormat5() ■ js:convertFormat10() ■ js:convertFormat9() ■ js:convertFormat6() ■ js:convertFormat7() ■ js:convertFormat11() 	Returns the specified date in a selected format	<p>Specify the date (in YYYY-MM-DD format) to be converted into the desired format.</p> <ul style="list-style-type: none"> ■ DD-MM-YYYY ■ DD/MM/YYYY ■ DD-MM-YYYY HH:mm ■ DD-MM-YYYY HH:mm am/pm ■ DD-MM-YYYY HH:mm:ss ■ DD-MM-YYYY HH:mm:ss am/pm ■ YYYY-MM-DD ■ YYYY-MM-DD YYYY-MM-DDTHH:mm:ss ■ YYYY-MM-DDTHH:mm:ss.ms ■ BaanDate (MMDDYYHHMMSS) - stores dates as the number of days from January 1, 1970 to the specified current date. This returns the current date and time in the format. ■ DD-MM-YYYY to YYYY-MM-DDTHH:mm:ss.ms
Date to UTC format	js:fnDateToUTC()	Converts a date into the UTC (Coordinated Universal Time) format.	Specify the date to be converted into the UTC format.

Date and Time Function (in Map canvas)	Date and Time Function (in Model source tab)	Description	Parameters
UTC to Date	<ul style="list-style-type: none"> ■ js:fnUTCToDateFormat1() ■ js:fnUTCToDateFormat8() ■ js:fnUTCToDateFormat2() ■ js:fnUTCToDateFormat3() ■ js:fnUTCToDateFormat4() ■ js:fnUTCToDateFormat5() ■ js:fnUTCToDateFormat10() ■ js:fnUTCToDateFormat9() ■ js:fnUTCToDateFormat6() ■ js:fnUTCToDateFormat7() 	<p>Converts a date in the UTC format into a specified format.</p>	<p>UTC Number - Specify the UTC number to be converted into the specified date format.</p> <ul style="list-style-type: none"> ■ DD-MM-YYYY ■ DD/MM/YYYY ■ DD-MM-YYYY HH:mm ■ DD-MM-YYYY HH:mm am/pm ■ DD-MM-YYYY HH:mm:ss ■ DD-MM-YYYY HH:mm:ss am/pm ■ YYYY-MM-DD ■ YYYY-MM-DD YYYY-MM-DDTHH:mm:ss ■ YYYY-MM-DDTHH:mm:ss.ms ■ BaanDate (MMDDYYHHMMSS) - stores dates as the number of days from January 1, 1970 to the specified current date. This returns the current date and time in the format.
Add days to a Date	<ul style="list-style-type: none"> ■ js:fnAddDaysFormat1() ■ js:fnAddDaysFormat8() ■ js:fnAddDaysFormat2() ■ js:fnAddDaysFormat3() ■ js:fnAddDaysFormat4() ■ js:fnAddDaysFormat5() ■ js:fnAddDaysFormat10() ■ js:fnAddDaysFormat9() 	<p>Adds a specified number of days to a date.</p>	<p>Date - This parameter in the function block takes as input the string value of the date to which a specific number of days have to be added in one of the following formats:</p> <ul style="list-style-type: none"> ■ DD-MM-YYYY ■ DD/MM/YYYY ■ DD-MM-YYYY HH:mm

Date and Time Function (in Map canvas)	Date and Time Function (in Model source tab)	Description	Parameters
	<ul style="list-style-type: none"> ■ js:fnAddDaysFormat6() ■ js:fnAddDaysFormat7() 		<ul style="list-style-type: none"> ■ DD-MM-YYYY HH:mm am/pm ■ DD-MM-YYYY HH:mm:ss ■ DD-MM-YYYY HH:mm:ss am/pm ■ YYYY-MM-DD ■ YYYY-MM-DD YYYY-MM-DDTHH:mm:ss ■ YYYY-MM-DDTHH:mm:ss.ms ■ BaanDate (MMDDYYHHMMSS) - stores dates as the number of days from January 1, 1970 to the specified current date. This returns the current date and time in the format. <p>Number of Days to be Added - The second parameter in the function block takes as input the number of days to be added.</p>

Scientific functions

Scientific functions allow you to perform trigonometric calculations on data during transformation. See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

The scientific function block accepts only integers as valid inputs. Convert all source data into a numeric format using the Numeric function. Input parameters passed to all trigonometric functions (Sine, Cosine, Tangent, Arc Sine, Arc Cosine, Arc Tangent) should be in radians.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the

Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

The following table lists the various scientific functions supported by AppWorks Platform as they appear on the Map canvas of the Data Modeler, their equivalent functions on the Model Source tab, their functionality, and parameters.

Scientific function (in Map canvas)	Scientific function (in Model Source tab)	Description	Parameters
Sine	js:fnScientificSine()	Returns the Sine value of a specified number	Number - Specify the source element for which the Sine value has to be calculated.
Cosine	js:fnScientificCosine()	Returns the Cosine value of a specified number	Number - Specify the source element for which the Cosine value has to be calculated.
Tangent	js:fnScientificTangent()	Returns the Tangent value of a specified number	Number - Specify the source element for which the Tangent value has to be calculated.
Arc Sine	js:fnScientificArcSine()	Returns the Arc Sine value of a specified number	Number - Specify the source element for which the Arc Sine value has to be calculated.
Arc Cosine	js:fnScientificArcCosine()	Returns the Arc Cosine value of a specified number	Number - Specify the source element for which the Arc Cosine value has to be calculated.
Arc Tangent	js:fnScientificArcTangent()	Returns the Arc Tangent value of a specified number	Number - Specify the source element for which the Arc Tangent value has to be calculated.
Exponential	js:fnScientificExponential()	Returns 'e' raised to a power, that is, the value of enumber, where 'e' is approximately equal to 2.718, and number is the input parameter.	Number - Specify the source element for which the exponential value has to be calculated.

Scientific function (in Map canvas)	Scientific function (in Model Source tab)	Description	Parameters
Logarithm	js:fnScientificLog()	Returns the natural logarithm of a specified number	Number - Specify the source element for which the natural logarithm has to be calculated.
Power Of	js:fnScientificPowerof()	Returns the number raised to a specified power	Base Number - This parameter in the function block takes as input the base number that has to be raised. Index Number - This parameter takes as input the power to which the base number has to be raised.

Logical functions

Logical functions allow you to compare data and pass the result of the comparison to the target. See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

While using integers as valid inputs, convert the source data into a numeric format using the Numeric function.

The following table describes various logical functions supported by AppWorks Platform as they appear on the Map canvas of the modeler, their equivalent functions on the Model Source tab, their functionality, and parameters.

Logical Function (in Map canvas)	Logical function (in Model source tab)	Description	Parameters
Equal-To (Integer)	js:fnLogicalEqualNumber()	Compares if a numeric	The inputs to this function block are as follows:

Logical Function (in Map canvas)	Logical function (in Model source tab)	Description	Parameters
Comparison)		expression is equal-to another numeric expression, and returns a specified value based on the result of the expression	<ul style="list-style-type: none"> ■ Number1 - First integer value for comparison is specified in this parameter ■ Number2 - Second integer value to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first integer value is equal to the second integer value ■ If False - This parameter specifies the value to be returned if the first integer value is not equal to the second integer value
Equal-To (String Comparison)	js:fnLogicalEqualString()	Compares two strings for equality, and returns a specified value based on the result of the expression	<p>The inputs to this function block are as follows:</p> <ul style="list-style-type: none"> ■ String1 - First string for comparison is specified in this parameter ■ String2 - Second string to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first string is equal-to the

Logical Function (in Map canvas)	Logical function (in Model source tab)	Description	Parameters
			<p>second string</p> <ul style="list-style-type: none"> ■ If False - This parameter specifies the value to be returned if the first string is not equal to the second string
Greater - Than-or- Equal-To	js:fnLogicalGreaterThanOrEqual()	Compares if a numeric expression is greater-than-or-equal-to another numeric expression, and returns a specified value based on the result of the expression	<p>The inputs to this function block are as follows:</p> <ul style="list-style-type: none"> ■ Number1 - First integer value for comparison is specified in this parameter ■ Number2 - Second integer value to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first integer value is greater than or equal to the second integer value ■ If False - This parameter specifies the value to be returned if the first integer value is not greater than or equal to the second integer value
Greater-Than	js:fnLogicalGreaterThan()	Compares if a numeric expression is	<p>The inputs to this function block are as follows:</p> <ul style="list-style-type: none"> ■ Number1 - First

Logical Function (in Map canvas)	Logical function (in Model source tab)	Description	Parameters
		greater than another numeric expression, and returns a specified value based on the result of the expression	<p>integer value for comparison is specified in this parameter</p> <ul style="list-style-type: none"> ■ Number2 - Second integer value to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first integer value is greater than the second integer value ■ If False - This parameter specifies the value to be returned if the first integer value is not greater than the second integer value
Less-Than	js:fnLogicalLessThan()	Compares if a numeric expression is less-than another numeric expression, and returns a specified value based on the result of the expression	<p>The inputs to this function block are as follows:</p> <ul style="list-style-type: none"> ■ Number1 - First integer value for comparison is specified in this parameter ■ Number2 - Second integer value to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first integer value is less

Logical Function (in Map canvas)	Logical function (in Model source tab)	Description	Parameters
			<p>than the second integer value</p> <ul style="list-style-type: none"> ■ If False - This parameter specifies the value to be returned if the first integer value is not than the second integer value
Less-Than-or-Equal-To	js:fnLogicalLessThanEqual()	<p>Compares if a numeric expression is less-than-or-equal-to another numeric expression, and returns a specified value based on the result of the expression</p>	<p>The inputs to this function block are as follows:</p> <ul style="list-style-type: none"> ■ Number1 - First integer value for comparison is specified in this parameter ■ Number2 - Second integer value to be compared is specified in this parameter ■ If True - This parameter specifies the value to be returned if the first integer value is less than or equal to the second integer value ■ If False - This parameter specifies the value to be returned if the first integer value is not less than or equal to the second integer value

Cumulative functions

Cumulative functions are used to perform operations on repeating elements along the XPath of a source XML.

Note: For ease of use and understanding, the functions in the Map canvas of the Modeler tab are displayed in a readable format, abstracting the underlying XSLT 2.0 and XPath 2.0 standard specifications. The Edit Model Source tab contains the XSLT source code with in-house javascript functions. While users are recommended to utilize the functions from the Map canvas, advanced developers who prefer to work with code to transform data can use the equivalent in-house javascript functions and their wrapper implementations.

Note: The cumulative function block accepts only integers as valid inputs. Convert all source data into a numeric format using the Numeric function.

The following table lists the various cumulative functions supported by AppWorks Platform as they appear on the Map canvas of the Data Modeler, their equivalent functions in the Model Source tab, their functionality, and parameters.

Cumulative Function	Cumulative functions (in Model Source tab)	Description	Parameters
Sum	Sum()	Returns the sum of all the occurrences of the input element in the XPath of the source XML.	Source Element - The source element whose occurrence has to be summed is the input parameter for this function block.
Average	Sum() div count()	Returns the average of all the occurrences of the input element in the XPath of the source XML.	Source Element - The source element whose occurrence has to be averaged is the input parameter for this function block.
Count	count()	Returns the number of occurrences of the input element in the XPath of the source XML.	Source Element - The source element whose occurrence has to be counted is the input parameter for this function block.

Constant functions

Constant functions allow you to insert formatted numbers or strings in the transformed XML. See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions. The constant functions supported by AppWorks Platform are listed in the following table.

Constant Function	Description
Numeric	Number - The Numeric function converts the specified parameter to a number. Use this function to convert all source data into a numeric format.
String	String - The String function converts the specified parameter to a string.

Advanced functions

Advanced functions are used to perform complex operations on data.

To perform advanced functions:

- Right-click the Map Canvas (middle) pane of the Modeler tab in the Data Transformation model, and select Advanced from the required <Function> library.
See [Transforming data using XSLT functions](#), for more information on the procedure to transform the data using XSLT functions.

The various advanced functions supported by AppWorks Platform, their functionality, and parameters that have to be specified are described in the following table.

Advanced Function	Description	Parameters
Value Mapping	For a given source element, value pairs can be defined. Each value pair consists of the value of the source element, and a value specified by the user. During transformation, based on the value pair defined by the user, the source element's value is replaced by the corresponding value. The new value can in turn be mapped to the target.	The inputs to this function block are as follows: <ul style="list-style-type: none"> Source Element - The source element for which the value pair is defined is the input for this parameter. Value to be Replaced - The value of the source element that should be replaced is the input for this parameter. Value to Replace - The value of the source element that should replace the value specified in the second parameter is the input for this parameter. The above inputs are required for each value pair.
Trigger Web	This function is used to	The inputs to this function block are as

Advanced Function	Description	Parameters
Service	trigger a Web service for retrieving value pairs from a database.	<p>follows:</p> <ol style="list-style-type: none"> 1. Request - The parameters of the Web service are the inputs for this block. Do the following: <ul style="list-style-type: none"> ■ Drag the required Web service, from the Workspace Explorer, or the Quick Access Menu from My Documents view, to this block. ■ Click the attribute in the Source Template and click the required parameter in the request block. The XPath of the attribute is displayed as a parameter 2. Response Element to be Added - The element/node of the response, that is to be mapped to the target element/node, is the input for this parameter. 3. Suppress Error - This parameter allows you to supply a Boolean value of 'true' or 'false' to suppress errors. <ul style="list-style-type: none"> ■ If the Boolean value is set to 'true', errors (SOAP fault) during the execution of the Web service will be suppressed. However, data transformation will be carried out and the target element will take the value provided in the 'Default Value'. ■ If the Boolean value is set to 'false', in the event of errors during the execution of the Web service, data transformation will be aborted, and the error details will be displayed in the response. 4. Default Value - This parameter takes as input the default value that is to be provided in the target element, when there is a 'suppressed error'.
Local Lookup	This function is used to retrieve values for data translation from a content map.	<p>The inputs to this function block are as follows:</p> <ol style="list-style-type: none"> 1. Source - The element in the source tree

Advanced Function	Description	Parameters
		<p>for which the value pair should be retrieved from the content map.</p> <ol style="list-style-type: none"> 2. Content Map Path - The path of the content map from which the value pairs should be retrieved. 3. Entity - The value of the entity in the content map for which the value pair should be retrieved. 4. Source System - The value of the source system in the content map, based on which the entities will be picked up. 5. Destination System - The value of the destination system in the content map, based on which the entities will be picked up. 6. Default Value - The default action to be taken, if the entity is not found in the content map. Takes a Boolean value of true or false. <ul style="list-style-type: none"> ■ True if you want the target to contain the same value as the source. ■ False if you do not want the target to contain any value. <p>Drag the required content map from the Workspace Explorer to the map canvas area. The Content Map Path is populated in the properties section, by default</p>
Custom Java Call	This function is used to retrieve the value returned from a Custom Java method.	<p>The inputs for this function block are as follows:</p> <ol style="list-style-type: none"> 1. Class Name (Fully Qualified Name) - The class name of the Java method. Type the Fully Qualified Name of the class that contains the custom Java method. For example: <code>com.cordys.cpc.translation.utils.ContentMapping</code>. You should type the path of the jar, which contains the above method, in the Classpath of the JRE Configuration tab of

Advanced Function	Description	Parameters
		<p>Data Transformation Service Container.</p> <p>2. Method Name - Name of the custom Java method. For example: customMethod.</p> <p>3. Parameter- The parameters of the method are the inputs for this block. Do any of the following:</p> <ul style="list-style-type: none"> ■ Click the attribute in the Source Template and click the required parameter in the request block. The XPath of the attribute is displayed as a parameter. ■ Type the value in the parameter field directly.
Custom XSLT	This function is used to embed Custom XSLT.	<p>The input for this function block is the Custom XSLT content. You can use the Validate option to verify the correctness of the XSLT content.</p> <ul style="list-style-type: none"> ■ The Source element (input) link is optional for this function. However, when a source is defined, the repetition and context are considered during custom XSLT code execution. ■ The XSLT provided here is used to create the target node structure and values are mapped according the XPath of the Source element provided in the XSLT. ■ To view the XPath of the source or target element, right-click the required element and click Show XPath option in the context menu. The complete path of the element from the root node is displayed in the dialog box that appears. <p>For more information on using the Custom XSLT, see Using the custom XSLT function.</p>

Chapter 22

Using the custom XSLT function

The Custom XSLT function is used to embed custom XSLT in the data transformations. While you can use the functions and instructions provided by platform, you can also define your custom XSLT and use them in the data maps. However, you must ensure to follow the XSLT standards while defining the custom functions. Based on the XSLT provided, the target node structure will be created.

The following sections describe the usage of this Custom XSLT function in various scenarios:

1. Access the Data Transformation Modeler () to create a data transformation model.
The Data Transformation modeler page is displayed.
2. Drag the source schema from the project tree to the **Source** pane of the **Modeler** tab.
The schema instance of the Source is displayed on the modeler.
3. Drag the target schema to the **Target** pane of the **Modeler** tab.
The schema instance of the Target is displayed on the modeler.
4. Right-click the **Map Canvas** (middle) pane of the **Modeler** tab and select **Functions > Advanced > Custom XSLT**.
The Custom XSLT icon appears on the Map canvas.
5. Click **Custom XSLT**.
The properties table of the Custom XSLT is displayed at the bottom.

You must provide the Custom XSLT content along with the root node `xsl:template` as shown.

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- Custom XSLT content is given here -->
</xsl:template>
```

Important

- Ensure to provide proper XPaths in the XSLT to create the correct target node structure.
- To view the XPath of the source or target element, right-click the required element and click Show XPath option in the context menu. The complete path of the element from the root node is displayed in the dialog box that appears.
- Always ensure to use absolute XPath when the source context is not selected. You can provide a relative XPath when the source context is selected.

- AppWorks Platform supports XPATH 1.0 specifications. However, refer to features in AppWorks Platform that are not supported and limitations, for information on the list of features in AppWorks Platform that are not supported and limitations respectively.

The following scenarios demonstrate some of the usages of Custom XSLT through samples.

Scenario 1: Creating repeated target elements based on multiple source elements without linking from Source

Multiple source elements will be mapped to repeated target element.

Source schema

```
<element name="Company" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element maxOccurs="unbounded" name="Employee">
                <complexType>
                    <sequence>
                        <element name="Mobile">
                            <complexType>
                                <sequence>
                                    <element name="Home"
                                         type="xs:string"/>
                                    <element name="Office"
                                         type="xs:string"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

Target schema

```
<element xmlns="http://www.w3.org/2001/XMLSchema" name="Team"> <complexType>
<sequence> <element name="Employee" maxOccurs="unbounded"> <complexType> <sequence>
<element xmlns:xs="http://www.w3.org/2001/XMLSchema"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema" type="xs:string" name="Mobile"
         maxOccurs="unbounded" /> </sequence> </complexType> </element> </sequence>
</complexType> </element> </sequence> </complexType> </element>
```

To map the elements between the source and target templates using the Custom XSLT function:

1. Complete XPath of the source element should be given in the XSLT as no source context is selected.
2. Select **Custom XSLT** and click Team\Employee\Mobile in the Target pane.
3. Double-click Custom XSLT icon to open the properties pane and provide the following XSLT.

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <Mobile>
        <xsl:value-of select="/Company/Employee/Mobile/Home"/>
    </Mobile>
    <Mobile>
        <xsl:value-of select="/Company/Employee/Mobile/Office"/>
    </Mobile>
</xsl:template>
```

4. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
5. In the **Source XML** pane, type appropriate values for the source elements.
Alternatively, you may click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required.
The elements in the Source XML pane are populated with the specified values.
6. Click **Transform** to start the data transformation process.

The data is transformed and displayed in the Transformed XML pane indicating the completion of the process as shown.

Test source XML

```
<element name="Catalog" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Products">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" minOccurs="0" name="Product">
                            <complexType>
                                <sequence>
                                    <xsd:element name="ProductId"
                                        type="xsd:string"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                    <xsd:element
                                        name="ProductDescription"
                                        type="xsd:string"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

```
        </complexType>
    </element>
</sequence>
</complexType>
</element>
</sequence>
</complexType>
</element>
```

Generated target XML

```
<Team>
    <Employee>
        <Mobile>987654321</Mobile>
        <Mobile>987654367</Mobile>
    </Employee>
</Team>
```

Scenario 2: Creating an element under repeated target node linking from Source

Only one target element is created.

Source Schema 2

```
<element name="Catalog" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Products">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" minOccurs="0" name="Product">
                            <complexType>
                                <sequence>
                                    <xsd:element name="ProductId"
                                        type="xsd:string"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                    <xsd:element
                                        name="ProductDescription"
                                        type="xsd:string"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

Target Schema 2

```

<element name="PurchaseOrder" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Items">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" minOccurs="0" name="Item">
                            <complexType>
                                <sequence>
                                    <xsd:element name="ItemCode"
                                        type="xsd:string"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                    <element name="ItemDescription"
                                        type="xs:string"
                                        xmlns:xs="http://www.w3.org/2001/XMLSchema"
                                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>

```

Do the following to map the elements between the source and target templates using the Custom XSLT function:

1. Select **Custom XSLT** and click PurchaseOrder/Items/Item/ItemCode in the Target pane.
2. Obtain the complete XPath of the element Catalog/Products/Product/ProductId in the Source pane.
3. Double-click Custom XSLT icon to open the properties pane and provide following XSLT.

```

<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <ItemCode>
        <xsl:value-of select="/Catalog/Products/Product/ProductId"/>
    </ItemCode>
</xsl:template>

```

4. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
5. In the Source XML pane, type appropriate values for the source elements. Alternatively, you can click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required. The elements in the Source XML pane are populated with the specified values.

6. Click **Transform** to start the data transformation process.
7. The data is transformed and displayed in the Transformed XML pane indicating completion of the process.

Test source XML

```
<Catalog>
  <Products>
    <Product>
      <ProductId>1010</ProductId>
      <ProductDescription>Furniture</ProductDescription>
    </Product>
    <Product>
      <ProductId>1020</ProductId>
      <ProductDescription>clothing</ProductDescription>
    </Product>
  </Products>
</Catalog>
```

Generated target XML

```
<PurchaseOrder>
  <Items>
    <Item>
      <ItemCode>1010</ItemCode>
    </Item>
  </Items>
</PurchaseOrder>
```

Scenario 3: Creating repeated target elements when Source element is selected and an Absolute XPath is provided

This scenario is similar to scenario 2 with a link to the source element is provided additionally.

Consider the Source Schema and Target Schema that are given in Scenario 2.

1. Do the following to map the elements between the source and target templates using the Custom XSLT function:
 - a. Select **Custom XSLT** and click `PurchaseOrder/Items/Item/ItemCode` in the Target pane.
 - b. Select `Catalog/Products/Product/ProductId` from the Source pane and click **Custom XSLT**.
 - c. Get XPath of the source element `Catalog/Products/Product/ProductId` to prepare the XSLT.

2. Double-click the Custom XSLT icon to open the properties pane and provide the following XSLT:

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <ItemCode>
    <xsl:value-of select="/Catalog/Products/Product/ProductId"/>
  </ItemCode>
</xsl:template>
```

3. Click the **Test Model** tab.

The Test Model tab with the source XML is displayed.

4. In the Source XML pane, type appropriate values for the source elements. Alternatively, you may click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required. The elements in the Source XML pane are populated with the specified values.
5. Click **Transform** to start the data transformation process.
6. The data is transformed and displayed in the Transformed XML pane indicating the completion of the process.

Test source XML

```
<Catalog>
  <Products>
    <Product>
      <ProductId>1010</ProductId>
      <ProductDescription>Furniture</ProductDescription>
    </Product>
    <Product>
      <ProductId>1020</ProductId>
      <ProductDescription>clothing</ProductDescription>
    </Product>
  </Products>
</Catalog>
```

Generated target XML

```
<PurchaseOrder>
  <Items>
    <Item>
      <ItemCode>1010</ItemCode>
    </Item>
    <Item>
      <ItemCode>1010</ItemCode>
    </Item>
  </Items>
</PurchaseOrder>
```

Scenario 4: Creating repeated target elements when the Source element is not selected but a direct link is provided for other element

Consider the Source Schema and Target Schema that are used in Scenario 2.

1. Do the following to map the elements between the source and target templates using the Custom XSLT function:
 - a. Select Catalog/Products/Product/ProductId from Source pane and click on PurchaseOrder/Items/Item/ItemCode in the Target pane.
A direct mapping is created.
 - b. Select **Custom XSLT** and click PurchaseOrder/Items/Item/ItemDescription in the Target pane.
Ensure to provide the complete XPath of the source element in the XSLT, as you do not link to the Source directly.
 - c. Double-click Custom XSLT icon to open the properties pane and provide following XSLT:

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <ItemDescription>
        <xsl:value-of select="/Catalog/Products/Product/ProductDescription"/>
    </ItemDescription>
</xsl:template>
```

2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the Source XML pane, type appropriate values for the source elements. Alternatively, you may click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required.
The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.
5. The data is transformed and displayed in the Transformed XML pane indicating the completion of the process. The transformed XML is similar to the transformed XML in Scenario 2.

Test source XML

```
<Catalog>
    <Products>
        <Product>
            <ProductId>1010</ProductId>
            <ProductDescription>Furniture</ProductDescription>
```

```

</Product>
<Product>
    <ProductId>1020</ProductId>
    <ProductDescription>Clothing</ProductDescription>
</Product>
</Products>
</Catalog>

```

Generated target XML

```

<PurchaseOrder>
    <Items>
        <Item>
            <ItemCode>1010</ItemCode>
            <ItemDescription>Furniture</ItemDescription>
        </Item>
        <Item>
            <ItemCode>1020</ItemCode>
            <ItemDescription>Furniture</ItemDescription>
        </Item>
    </Items>
</PurchaseOrder>

```

Scenario 5: Creating repeated target elements when the Source element is selected and a relative XPath is provided

Consider the Source Schema and Target Schema that are used in Scenario 2.

1. Do the following to map the elements between the source and target templates using the Custom XSLT function:
 - a. Select Catalog/Products/Product/ProductId from Source pane and click PurchaseOrder/Items/Item/ItemCode in the Target pane.
A direct mapping is created.
 - b. Select **Custom XSLT** and click PurchaseOrder/Items/Item/ItemDescription in the Target pane.
 - c. Select Catalog/Products/Product/ProductDescription and click Custom XSLT. The source element is now linked with the Custom XSLT.
 - d. In the XSLT, provide the XPath from the Source; that is, the relative path to the source element.
The Source element is Product and the XPath is considered from there; that is, ProductDescription.
 - e. Double-click **Custom XSLT** icon to open the properties pane and provide following XSLT:

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <ItemDescription>
    <xsl:value-of select="ProductDescription"/>
  </ItemDescription>
</xsl:template>
```

2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the Source XML pane, type appropriate values for the source elements. Alternatively, you can click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required. The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.
5. The data is transformed and displayed in the Transformed XML pane indicating the completion of the process. The transformed XML is similar to the transformed XML in Scenario 2.

Test source XML

```
<Catalog>
  <Products>
    <Product>
      <ProductId>1010</ProductId>
      <ProductDescription>Furniture</ProductDescription>
    </Product>
    <Product>
      <ProductId>1020</ProductId>
      <ProductDescription>Clothing</ProductDescription>
    </Product>
  </Products>
</Catalog>
```

Generated target XML

```
<PurchaseOrder xmlns="">
  <Items>
    <Item>
      <ItemCode>1010</ItemCode>
      <ItemDescription>Furniture</ItemDescription>
    </Item>
    <Item>
      <ItemCode>1020</ItemCode>
      <ItemDescription>Clothing</ItemDescription>
    </Item>
  </Items>
</PurchaseOrder>
```

Scenario 6 : Creating a Target element based on the Source elements value without the Source element selected directly

Consider the following Source schema and Target Schema for this Scenario.

Source schema

```
<element name="Orders" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element maxOccurs="unbounded" minOccurs="0" name="Order">
                <complexType>
                    <sequence>
                        <xsd:element name="CustomerID" type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                        <element maxOccurs="unbounded" minOccurs="0" name="date">
                            <complexType>
                                <sequence>
                                    <xsd:element name="type"
type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                    <xsd:element name="value"
type="xsd:string"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

Target schema

```
<element name="PurchaseOrder" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="OrderId" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
            <element name="Dates">
                <complexType>
                    <sequence>
                        <element minOccurs="0" name="OrderDate"
type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
    <element minOccurs="0" name="BillingDate"
        type="xs:string"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
    <element minOccurs="0" name="DeliveryDate"
        type="xs:string"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
        </sequence>
    </complexType>
</element>
</sequence>
</complexType>
</element>
```

1. Do the following to map the elements between the source and target templates using the Custom XSLT function:
 - a. Select **Custom XSLT** and click /PurchaseOrder/Dates/OrderDate in the Target pane.
 - b. Provide the XPath of the source elements in the XSLT as required to create target nodes.
 - c. Double-click Custom XSLT icon to open the properties pane and provide following XSLT:
Note: The variable orderpath is created and used throughout the XSLT and xsl:if with a condition is used.

```
<xsl:template xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:variable name="orderpath" select="/Orders/Order"/>
    <OrderDate>
        <xsl:value-of select="$orderpath/date[type='Order']/value"/>
    </OrderDate>
    <xsl:if test="$orderpath/date[type='Billing']">
        <BillingDate>
            <xsl:value-of select="$orderpath/date[type='Billing']/value"/>
        </BillingDate>
    </xsl:if>
    <xsl:if test="$orderpath/date[type='Delivery']">
        <DeliveryDate>
            <xsl:value-of select="$orderpath/date[type='Delivery']/value"/>
        </DeliveryDate>
    </xsl:if>
</xsl:template>
```

2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the Source XML pane, type the appropriate values for the source elements.
Alternatively, you may click Fill Source Data , which assigns sample string values to the

source elements. Make changes to these values, if required. The elements in the Source XML pane are populated with the specified values.

4. Click **Transform** to start the data transformation process.
5. The data is transformed and displayed in the Transformed XML pane indicating the completion of the process. The transformed XML is similar to the transformed XML in Scenario 2.

Test source XML

```
<Orders>
  <Order>
    <CustomerID>GREAL</CustomerID>
    <date>
      <type>Order</type>
      <value>09/07/2013</value>
    </date>
    <date>
      <type>Billing</type>
      <value>09/07/2013</value>
    </date>
    <date>
      <type>Shipping</type>
      <value>09/07/2013</value>
    </date>
  </Order>
</Orders>
```

Generated target XML

```
<PurchaseOrder>
  <Dates>
    <OrderDate>09/07/2013</OrderDate>
    <BillingDate>09/07/2013</BillingDate>
  </Dates>
</PurchaseOrder>
```

Important

The following actions on the Custom XSLT function are not allowed:

- Custom XSLT does not allow mapping to or mapping from other function blocks.
- Custom XSLT can be mapped from one Source Element only. Mapping to multiple source elements is not allowed.
- The XPaths defined in custom XSLT may be invalid when a second schema is added to the Source or Target tree.

Instructions

Apart from the XSLT functions, AppWorks Platform also provides three instructions that enable you to check if the source element exists, copy source node values to the target elements, and create target element based on the number of occurrences of source elements.

The Instructions supported by AppWorks Platform, their functionality, and parameters that have to be specified are described in the following table.

Instruction	Description	Parameters
If Exists	Checks the existence of the source element	<p>Source Element - Select the source element from the Source tree, map it to the If Exists block, and map the instruction to the Target element.</p> <p>If the source element does not exist, then the corresponding target element is not created.</p>
Copy	Copies the source element and value to the target element	<p>Source Element - Select the source element from the Source tree, map it to the parameter on the Copy instruction block, and map the instruction block to the Target element.</p>
Repeat Target	Repeats the target element with the value of the mapped source elements.	<p>Value - Select the source element from the source tree, map it to the parameter on the Repeat Target instruction block, and map the instruction block to the Target element.</p> <p>Note</p> <ul style="list-style-type: none"> ■ The Repeat Target instruction can be used to map any number of Source element nodes to any number of target nodes. ■ The Source element nodes must be: <ul style="list-style-type: none"> • non-repeating leaf element nodes • repeating leaf element nodes. A repeating node is the node that contains the minOccurs or maxOccurs values set as >1. • combination of both. The Target element node must be a repeating leaf node. ■ The Repeat Target instruction cannot be used: <ul style="list-style-type: none"> • for mapping to and mapping from other function or instruction blocks • if the source or target nodes contain child nodes ■ The attributes of a source element node can be

Instruction	Description	Parameters
		directly mapped to the attributes of a target element node only if the source and target elements are mapped using the Repeat Target instruction.

Chapter 23

Using the Repeat Target instruction

The Repeat Target instruction is used to map multiple source elements to multiple target elements. Based on the number of elements mapped from Source, the target elements are created. The following sections describe the usage of this instruction in various scenarios: The sample Source and Target schemas used in the scenarios follow.

Source schema

```
<element name="Developement" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Developers">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" name="EName">
                            <complexType>
                                <simpleContent>
                                    <extension base="xs:string"
                                        xmlns:xs="http://www.w3.org/2001/XMLSchema">
                                        <attribute name="ENumber" type="xs:string"/>
                                    </extension>
                                </simpleContent>
                            </complexType>
                        </element>
                        <element maxOccurs="unbounded"
                            name="Known_Tools" type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"
                            xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                        <element maxOccurs="unbounded"
                            name="Known_ProgrammingLanguages"
                            type="xs:string"
                            xmlns:xs="http://www.w3.org/2001/XMLSchema"
                            xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
</element>
```

Target schema

```

<element name="TeamDetails" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Team">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" name="Employee">
                            <complexType>
                                <simpleContent>
                                    <extension base="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                                        <attribute name="id" type="xs:string"/>
                                    </extension>
                                </simpleContent>
                            </complexType>
                        </element>
                    <element name="TeamSkills">
                        <complexType>
                            <sequence>
                                <element maxOccurs="unbounded"
name="skill" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </complexType>
        </sequence>
    </element>
</complexType>
</element>

```

To use the Repeat Target instruction:

1. Access the Data Transformation Modeler () to create a data transformation model.
The Data Transformation modeler page is displayed.
2. Drag the source schema from the project tree to the **Source** pane of the **Modeler** tab.
The schema instance of the Source is displayed on the modeler.
3. Drag the target schema to the **Target** pane of the **Modeler** tab.
The schema instance of the Target is displayed on the modeler.
4. Right-click the **Map Canvas** (middle) pane of the **Modeler** tab and select **Instructions** > **Repeat Target**.
The Repeat Target icon appears on the Map canvas.
5. Click the **Repeat Target** icon.
The properties table of the Repeat Target is displayed at the bottom.

You can test the following scenarios.

Scenario 1: When a single Source element is mapped to Repeat Target instruction

The target element will be created as many times as the Source element available.

1. Do the following to map the elements between the source and target templates using the Repeat Target instruction:
 - a. Select `Developement\Developers\EName` from the **Source** pane, hold the **CTRL** key and select the corresponding row on the Repeat Target properties table.
 - b. Select **Repeat Target** and click `TeamDetails\Team\Employee` in the Target pane.
 - c. Similarly, map the attribute `Developement\Developers\EName\@ENumber` to `TeamDetails\Team\Employee\@id` directly.
You can do this only when the source and target elements are mapped using the Repeat Target instruction.
A line appears connecting the mapped elements in the source, the function block, and target templates, indicating that the output values, based on the inputs to the `<Function>` from the source template, are mapped to the selected target element.
2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the **Source XML** pane, type appropriate values for the source elements.
Alternatively, you may click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required.
The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.

The data is transformed and displayed in the Transformed XML pane indicating the completion of the process.

Test source XML

```
<dt:Developement xmlns:dt="dt">
  <dt:Developers>
    <!-- Repeated EName, ENnumber is used in source -->
    <dt:EName ENumber="1">EMP1</dt:EName>
    <dt:EName ENumber="2">EMP2</dt:EName>
    <dt:EName ENumber="3">EMP3</dt:EName>
    <dt:Known_Tools>January</dt:Known_Tools>
    <dt:Known_ProgrammingLanguages>August</dt:Known_ProgrammingLanguages>
  </dt:Developers>
</dt:Developement>
```

Generated target XML

```
<dt:TeamDetails xmlns:dt="dt">
  <dt:Team>
    <!-- Repeated dt:Employee is created with respect to the source EName and ENumber --
    >
    <dt:Employee id="1">EMP1</dt:Employee>
    <dt:Employee id="2">EMP2</dt:Employee>
    <dt:Employee id="3">EMP3</dt:Employee>
  </dt:Team>
</dt:TeamDetails>
```

Scenario 2: When multiple source elements of the same parent are mapped to the Repeat Target instruction

The Target element will be created as many times as source elements available. The Source and Target schemas are same as used in Scenario 1.

1. Map the elements between the source and target templates using the Repeat Target instruction as follows:
 - a. Select `Development\Developers\Known_Tools` from the **Source** pane, hold the **CTRL** key and select the corresponding row on the Repeat Target properties table.
 - b. Select `Development\Developers\Known_ProgrammingLanguages` from the **Source** pane, hold the **CTRL** key and select the corresponding row on the Repeat Target properties table.
 - c. Select **Repeat Target** and click `TeamDetails\Team\TeamSkills\skill` in the **Target** pane.

A line appears connecting the mapped elements in the source, the function block, and target templates, indicating that the output values, based on the inputs to the `<Function>` from the source template, are mapped to the selected target element .
2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the **Source XML** pane, type appropriate values for the source elements.
Alternatively, you may click **Fill Source Data**, which assigns sample string values to the source elements. Make changes to these values, if required.
The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.

The data is transformed and displayed in the Transformed XML pane indicating the completion of the process.

Test source XML

```
<dt:Developement xmlns:dt="dt">
  <dt:Developers>
    <dt:EName ENumber="">February</dt:EName>
    <!-- Repeated 'Known_Tools' used in source -->
    <dt:Known_Tools>tool_1</dt:Known_Tools>
    <dt:Known_Tools>tool_2</dt:Known_Tools>
    <dt:Known_Tools>tool_3</dt:Known_Tools>
    <!-- Repeated 'Known_ProgrammingLanguages' used in source -->
    <dt:Known_ProgrammingLanguages>Lang_1</dt:Known_ProgrammingLanguages>
    <dt:Known_ProgrammingLanguages>Lang_2</dt:Known_ProgrammingLanguages>
    <dt:Known_ProgrammingLanguages>Lang_3</dt:Known_ProgrammingLanguages>
  </dt:Developers>
</dt:Developement>
```

Generated target XML

```
<dt:TeamDetails xmlns:dt="dt">
  <dt:Team>
    <dt:TeamSkills>
      <!-- Target 'skill' will be repeatedly generated as many times as source
elements ('Known_Tools', 'Known_ProgrammingLanguages') are available -->
      <dt:skill>tool_1</dt:skill>
      <dt:skill>tool_2</dt:skill>
      <dt:skill>tool_3</dt:skill>
      <dt:skill>Lang_1</dt:skill>
      <dt:skill>Lang_2</dt:skill>
      <dt:skill>Lang_3</dt:skill>
    </dt:TeamSkills>
  </dt:Team>
</dt:TeamDetails>
```

Scenario 3 : When multiple Source elements of different parents are mapped to the Repeat Target instruction

The Target element will be created as many times for one of the source elements and one time for the remaining source elements.

The Target element will be created as many times as the source elements available if and only if the parent nodes of both the source and target elements are non-repeating elements.

Source schema

Consider the following source schema for this scenario.

Source XML

```

<element name="Developement" xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType>
        <sequence>
            <element name="Developers">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" name="EName">
                            <complexType>
                                <simpleContent>
                                    <extension base="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                                        <attribute name="ENumber" type="xs:string"/>
                                    </extension>
                                </simpleContent>
                            </complexType>
                        </element>
                        <element maxOccurs="unbounded"
name="Known_Tools" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                            <element maxOccurs="unbounded"
name="Known_ProgrammingLanguages"
type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </complexType>
            </element>
            <element maxOccurs="unbounded" name="Architect">
                <complexType>
                    <sequence>
                        <element maxOccurs="unbounded" name="AName">
                            <complexType>
                                <simpleContent>
                                    <extension base="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
                                        <attribute name="ANumber" type="xs:string"/>
                                    </extension>
                                </simpleContent>
                            </complexType>
                        </element>
                        <element maxOccurs="unbounded"
name="Known_Tools" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                            <element maxOccurs="unbounded"
name="Known_Designs" type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
                            <element maxOccurs="unbounded"
name="Known_Products" type="xs:string"

```

```

    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" />
        </sequence>
    </complexType>
</element>
</sequence>
</complexType>
</element>

```

Target schema

You can use the same Target schema used in Scenario 1.

1. Map the elements between the source and target templates using the Repeat Target instruction as follows:
 - a. Select Developement\Developers\Known_Tools from the **Source** pane, hold the **CTRL** key and select the corresponding row on the **Repeat Target** properties table.
 - b. Select Developement\Developers\Known_ProgrammingLanguages from the **Source** pane, hold the **CTRL** key and select the corresponding row on the Repeat Target properties table.
 - c. Select Developement\Architect\Known_Tools from the **Source** pane, hold the **CTRL** key and select the corresponding row on the **Repeat Target** properties table.
 - d. Select Developement\Architect\Known_ProgrammingLanguages from the **Source** pane, hold the **CTRL** key and select the corresponding row on the **Repeat Target** properties table.
 - e. Select **Repeat Target** and click TeamDetails\Team\TeamSkills\skill in the **Target** pane.

A line appears connecting the mapped elements in the source, the function block, and target templates, indicating that the output values, based on the inputs to the <Function> from the source template, are mapped to the selected target element.

2. Click the **Test Model** tab.
The Test Model tab with the source XML is displayed.
3. In the **Source XML** pane, type appropriate values for the source elements.
Alternatively, you may click Fill Source Data, which assigns sample string values to the source elements. Make changes to these values, if required.
The elements in the Source XML pane are populated with the specified values.
4. Click **Transform** to start the data transformation process.

The data is transformed and displayed in the Transformed XML pane indicating the completion of the process.

XSLT functions that are not supported in AppWorks Platform

The following features of XSLT are not supported in AppWorks Platform.

Elements

The following are rarely used XSL constructs and will be supported in next version.

- XSL:number
- XSL:decimal

For these two elements, the 'XSL:element-available' function will return the value 'true', because they are not extension elements. However, stylesheet compilation will throw errors for these elements.

Functions

- Document() function accepts only one argument, whereas it could be two arguments as per the XSLT specification
- Format-number() is not supported
- Unparsed-entity-uri() is not supported

Attributes for XSL:output element

- encoding
- standalone
- doctype-public
- doctype-system
- cdata-section-elements
- media-type

Attributes for XSL:sort

For sort element, case-order is not supported.

XSL:import and XSL:include behavior

The XSL:import and XSL:include instructions do not fetch documents from remote locations. As top level elements, they support only XSLT1.0 elements and other elements that originate from other namespaces.

Limitations

This topic lists the limitations of XPath and XSLT functions used in the product.

Namespace

For performance reasons, namespace processing is disabled by default. You have to use a prefix inside the XPath expression to enable namespace processing.

XSLT variables

For NOM-XSLT, variable or parameters must be declared before using them. This condition applies to the top level parameters and variables.

XML Declarations

XML declaration (<? xml version=1.0?>) are not be added to the result tree.

White space handling

- White space characters are ignored by default, to save parsing time and to ensure that the parsed object is smaller in size. However, you can use <XSL:text /> to enforce white space character into the result tree.
- Output escaping is enabled by default, that is, special characters like &, >, <, and " will be replaced with &, >, < and ".

CDATA

Content of CDATA elements are treated as literal result elements. However, based on user feed back, this will be changed.

Qname validation

The name attribute of many XSLT constructs require a QName. The validation is kept at minimum to reduce the parsing time.

Comparison of strings

For performance reasons, strings are always compared. A parser will not convert strings to numbers and then compare. This is because there may be ways to determine what is required - a string comparison or a number comparison.

XSL:variable's namespace

There can be multiple prefix declarations for same namespace URI. However, you have to use the same prefix with the variable declaration as well as inside the reference to the variable.

For example, the following usage will not work with nomxslt.

```
<XSL:stylesheet xmlns:XSL="http://www.w3.org/1999/XSL/Transform"
    xmlns:prefix1="http://hello.com" xmlns:prefix2="http://hello.com">
    <XSL:variable name="prefix1:MyVar" select="."/>
    <XSL:template match="/">
        <XSL:value-of select="$prefix2:MyVar"/>
    </XSL:template>
</XSL:stylesheet>
```

Importing XSLT files

Importing and including other XSLT files will not work if the input XSLT is given as a string (not a file).

XPath `text()` function

Default size for a text node is 5 kb. If the size of the data is greater than 5 kb, it will be stored in multiple text nodes causing the `text()` function to retrieve only the first 5 kb of data.

- To increase the size of the text node use the property
`bus.xml.nom.normalizedatabuffersize`.

Overview of XPath

AppWorks Platform supports all the functions in [XPATH 1.0](#) specifications as recommended by [W3C](#) and specific XPath 2.0 functions supported by AppWorks Platform are specified [here](#).

Understanding XPath

XPath is one of the most useful features while using XML elements. Its primary purpose is to address specific parts (nodes) of an XML document. In simple terms, it functions more like the SQL 'select' statement for databases. As the name suggests, it uses path notation as in URL(s) for navigating through the hierarchical structure of an XML document and identifying the nodes in it. This process is similar to the path expressions of your file system.

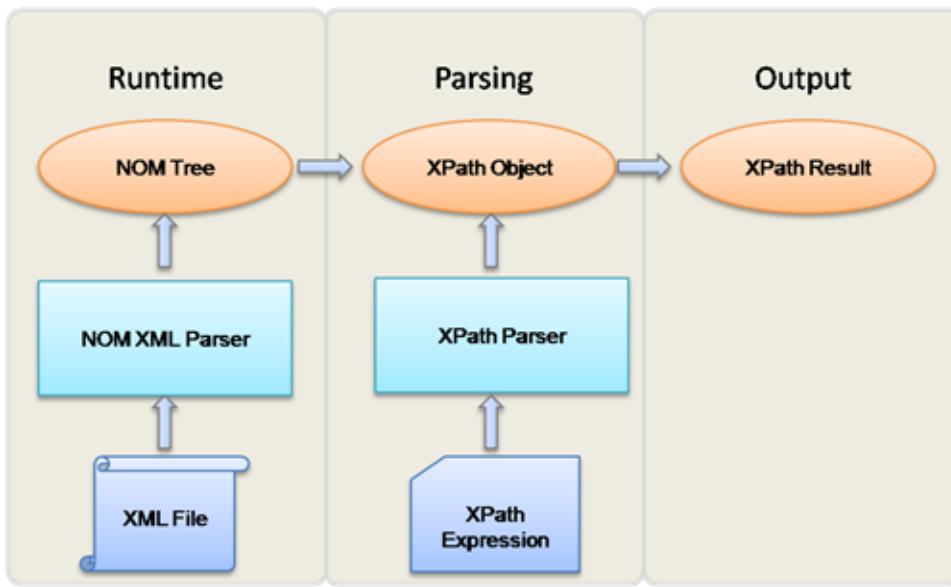
Additionally, XPath also provides basic facilities for manipulating strings, numbers, and Boolean values. It operates on the abstract, logical structure of an XML document, rather than its surface syntax and uses a compact, non-XML syntax to identify particular parts of XML documents within URI(s) and XML attribute values.

XPath lets you write expressions that refer to the document's specific elements with respect to their position (first person element, the seventh child element of the third person element, and so on) including all xmlstylesheet processing instructions in the document's prolog.

XPath is a specification from W3C (<http://www.w3.org>). For more details on the specification as recommended by W3C, see <http://www.w3.org/TR/xpath>.

Architecture of XPath

The following diagram demonstrates the way XPath processes an XML document.



XPath Concepts

To understand XPath, you must understand the concepts of XPath. The following are the concepts of XPath:

- Understanding XPath Expressions
- Defining XPath Expressions
- XPath Parser, XPath Object, and XPath Output

You can use XPath API(s) that are implemented in AppWorks Platform. For more information on XPath API(s), refer to Using XPath API(s).

Benefits of using XPath

XPath is designed for XML documents. It provides a single syntax that you can use for queries, addressing, and patterns. XPath is concise, simple, and powerful. The benefits of XPath are as follows:

- Queries are compact, easy to type and read, and are easily parsed.
- Syntax is simple for the most common cases that are simple in nature.
- Query strings are easily embedded in programs, scripts, and XML or HTML attributes.
- Enables specifying any path that can occur in an XML document and any set of conditions for the nodes in the path.
- Enables uniquely identifying any node in an XML document.
- Queries return any number of results, including zero, but not repeated nodes.
- Query conditions can be evaluated at any level of a document and are not expected to navigate from the top node of a document.
- Enables usage in many contexts. It is used for providing links to nodes, for searching repositories, and for many other applications.

Invoking Java functions using XPath

Java function can be invoked through an XPath to facilitate implementation of custom logic.

Sample code

```
package com.cordys.xml.nom;
import com.eibus.xml.xpath.XPath;
import com.eibus.xml.xpath.XPathResult;
public class XPathExample {
    public static String getHelloString(String arg) {
        return "Hello " + arg;
    }
    public static void main(String[] args) {
        XPath xpath = XPath
            .getXPathInstance("com.cordys.xml.nom.SampleXPath.getHelloString
(\\"World\\")");
        XPathResult result = xpath.evaluate(null);
        System.out.println(result.getStringResult());
    }
}
```

Output

```
Hello World
```

Understanding XPath expressions

An XPath expression is the first and foremost requirement for implementing XPath. These expressions help you to select information in an XML document. The XPath expression should be syntactically correct, as defined by XPath 1.0 specification.

To understand XPath expression, consider the following XML document as an example:

```
<PersonList>
    <Person>
        <Name>John</Name>
        <Age>24</Age>
        <Gender>M</Gender>
        <PostalCode>54879</PostalCode>
    </Person>
    <Person>
        <Name>Jasmin</Name>
        <Age>28</Age>
        <Gender>F</Gender>
        <PostalCode>78745</PostalCode>
    </Person>
</PersonList>
```

To select different types of information from the XML document, different XPath expressions are required. The sample expressions listed below (with reference to the example) pick different portions of the document, based on the specific requirement.

To select all names in the PersonList or Person element, use:

```
/PersonList/Person/Name
```

Other than the equal operator (=), you can use the greater than (>) and less than (<), <= (less than or equal to), >= (greater than or equal to) and != (not equal).

The logical operators are:

- and
- or

To search for persons who are older than 18 years old, use:

```
/PersonList/Person[Age>18]
```

Strings are compared case sensitively. You can use '*' as a wildcard. The operators < and > can be used to compare strings alphabetically.

To get the second person in the XML document, use:

```
/PersonList/Person/[2]
```

If you want to create a generic expression that can potentially select every person, replace the number '2' with a variable reference. For more information on variable binding, see [Variable Binding](#).

To get the last node, use:

```
/PersonList/Person/[last()]
```

This is useful for iterations. However, there is no function named first().

To select several paths use the (|) operator:

```
/PersonList/Person/Name | /PersonList/Person/Age
```

This will select all name elements and age elements.

In NOM, the expression to select the root will be '/node()' and not '/'. However, there is one exception. If you attach the XSLT 'document' function to XPath expression, the document element (first child of root) will be passed to the attached expression.

For example, if the expression is:

```
document('weather.xml')/child::node()
```

And if the document is

```
<weather>
  <forecast/>
</weather>
```

Then, `<forecast>` will be selected. Therefore, such expressions should be used with caution.

Defining XPath expressions

To control the expression evaluation, you can pass dynamic information such as, variable binding, namespace binding, and so on to the XPath execution context with the help of `XPathMetaInfo` object. This `XPathMetaInfo` object can be shared across different XPath objects.

The different options for using `XPathMetaInfo` Object are:

- Variable binding
- Namespace binding
- Function registration

Variable binding

XPath expressions can contain variable reference that requires a value to be assigned to it. For example, the expression `//personalInfo[@name=$var]`, will select the `personalInfo` elements having the value of the name attribute equal to the value of the variable reference `var`. To get the expression evaluated, the variable `var` has to be assigned a value. Here is the algorithm for doing the same:

```
XPath xpathObject = new XPath("//personalInfo[@name = $var]"); XPathMetaInfo
xpathMetaInfoObj = new XPathMetaInfo(); XPathResult XpathResultObj = new XPathResult
("Toma"); XPathMetaInfoObj.addVarBinding("var" , XpathResultObj); XPathResult res =
XpathObject.evaluate(iNomNode, XpathMetaInfoObj);
```

In the example, `iNomNode` refers to the NOM handle containing the XML.

Namespace binding

Similar to variable binding, you can bind a URI to a prefix used in an XPath expression:

```
XPath xpathObject = new XPath("//ns1:onalInfo[@name = $var]"); XPathMetaInfo
xpathMetaInfoObj = new XPathMetaInfo(); XPathMetaInfoObj.addNamespaceBinding("ns1",
"http://hello.com"); XPathResult res = XpathObject.evaluate(iNomNode,
XpathMetaInfoObj);
```

When you use namespace specific search, ensure the following:

- Use a prefix inside the XPath expressions to find a specific element that is originating from a particular namespace. The prefix could either be the actual prefix used in the input document or the default namespace (no prefix). But the prefix you use inside the expression should be mapped to the actual URI that you expect inside the input document.
- For performance reasons, the namespace processing is disabled by default. However, if you use a prefix inside XPath expression, it will enable the namespace processing automatically. The expression will match nodes if the local part of the element name is matched. For example, consider following expression.

```
"/mycat/name"
```

And the input is:

```
<cat xmlns="http://cat.com">
  <mycat>
    <name>Toma</name>
  </mycat>
</cat>
```

The expression will select the 'name' element even though it is originating from `http://cat.com`. However, for the following structure (which is semantically equivalent to the above input), the same expression will not select the nodes.

```
<ns:cat xmlns:ns="http://cat.com">
  <ns:mycat>
    <ns:name>Toma</ns:name>
  </ns:mycat>
</ns:cat>
```

For this input, you need to add a namespace binding. The above expression can be changed to:

```
"/ns1:mycat/ns1:name"
```

Binding 'ns1' to '`http://cat.com`' will work. This limitation is to improve the performance and it was observed that most of the searches happen irrespective of the namespace declaration.

Function registration

Most of the functions described by XPath1.0 and XSLT1.0 are available in AppWorks Platform. However, for a list of missing functions, see [Features that are not supported in AppWorks Platform](#).

If you want to use custom functions, you must register them (for native users) from the native layer. These functions must be called from XPath expressions, using MetaInfo object. Users can associate a name with a function pointer of the type:

```
void (*lib_fun) ( void *pXPathContext , CXPathResult *pArg[], CXPathResult  
*retValue);
```

The returnValue is used to pass the result back to the engine. Initially, it will contain the number of arguments actually passed through the expression.

For using custom Java functions, though a separate registration is not needed from the Java layer, the following conditions must be satisfied:

- Java function must be static
- The class must be set in the classpath
- The Java method call can return Nodeset however, the Nodeset must include only those nodes which are part of the input NOM tree originally provided.

According to the XPath1.0 specification, the XPath function id(Object) selects nodes by their unique identity (ID).

NOM does not have ID support, but XPath specification has ID aspect. As a limited work around, an user can have a particular attribute designated as the ID attribute during XPath evaluation by using the following API of the XPathMetaInfoclass:

```
registerAttribute(java.lang.String Attr)
```

This makes the XPath processor consider the registered attribute as the ID attribute and select nodes accordingly.

Parsing XPath expressions

This topic explains parsing the XPath expressions.

XPath parser

XPath Parser is a library comprising a parser for XPath expressions, and a set of classes which are used to represent the structure of the expressions. All XPath expressions are parsed to a syntax tree (a binary tree) inside which, the lookup mechanism is embedded. This parser creates class-based structure of an XPath expression that can be manipulated as required.

One of the following search methods is selected based on the given expression:

- Breadth First Search (BFS)
- Depth First Search (DFS)
- A combination of both

XPath object

After an XPath expression is parsed, the XPath object created with the syntax tree is placed inside it. This syntax tree is used to evaluate the XPath expression for a given input. The syntax tree is also cached by the engine for later use (get multiple XPath objects with the same syntax tree when the same expression is used iteratively). The syntax tree remains in the cache even when the XPath object is deleted. The generated XPath object is thread-safe and can be shared across multiple threads.

The generated XPath object is used to find nodes in a given NOM tree, as specified by the XPath expression. The XPath object containing the lookup criteria performs the lookup in NOM tree.

XPath output

The output is the XPathResult object that contains any of the following types, as defined by XPath1.0 specifications:

- Boolean
- Nodeset
- Number
- String
- Custom (User defined)

Nodeset contains XPathResultNode objects that are wrappers for the selected nodes of the NOM tree.

Using XPath API

XPath API(s) can be used in C or C++ or Java.

C or C++

There is no automatic garbage collection in the native layer. Therefore, ensure that you free the memory after use.

Include `xmlLib.h`. If you want a C++ specific style with overloaded functions, then include `XPath_xsl_optional.h`.

```
tHXmlDoc xmlDoc = (tHXmlDoc)malloc(sizeof(tXmlDoc)); xmlInitDoc(xmlDoc); tChar
pErrorBuf[1024]; tHXmlNode pXmlNode = xmlReadFromFile(xmlDoc, (tChar
*)"d:\\weather.xml", pErrorBuf, 1024); /* Creating XPath */ tXPath hXPath =
xpathGetXPathInstance( "/weather/forecast/temperature[1]" ); tXPathResult
hXPathResult = 0; tHXPathNodeset hXPathNodeset = 0; tHXPathMetaInfo hMetaInfo = 0;
if(hXPath) { /* Selecting NOM nodes */ hXPathNodeset = xpathSelectNodes(hXPath ,
pXmlNode, hMetaInfo); /* You can also use the more generic form - xpathValueOf()
methods, */ deleteXPathNodesetInstance(hXPathNodeset); deleteXPathInstance(hXPath);
}
```

Java

In Java, the usage is more or less like C or C++. For more information, see SDK documentation.

```
Document oDocument = new Document(); int iXml; try { iXml = oDocument.load
("weather.xml"); XPath oXPath0 = XPath.getInstance("//root/table "); NodeSet
oNodeset = oXPath0.selectNodeSet(iXml); oNodeset.delete(); oXPath0.delete(); } catch
(XMLEception e) { e.printStackTrace(); }
```

XPath 2.0 functions supported by AppWorks Platform

AppWorks Platform supports only a specific set of XPath 2.0 functions. This topic lists the supported functions and limitations.

Supported functions

- fn:min

Selects an item from the input sequence whose value is less than or equal to the value of every other item in the input sequence.

```
fn:min($arg as xs:anyAtomicType*) as xs:anyAtomicType?
```

Sample code

```
Document document = new Document();
int xml = document.parseString("<weather>
    + "<forecast day=\"1\" ><temperature>3.0</temperature> "
    + "</forecast> <forecast day=\"2\">
    + "<temperature>5.0</temperature></forecast>"
    + "<forecast day=\"3\"><temperature>4.0</temperature> "
    + "</forecast> </weather>");
XPath oXPath = XPath.getInstance("min(/weather/forecast/temperature/text())");
System.out.println(oXPath.evaluateNumberResult(xml));
```

Output

```
3.0
```

- fn:max

Selects an item from the input sequence whose value is greater than or equal to the value of every other item in the input sequence.

```
fn:max($arg as xs:anyAtomicType*) as xs:anyAtomicType?
```

Sample code

```
5.0
```

- fn:upper-case

Returns the value of \$arg after translating every character to its upper-case correspondent.

```
fn:upper-case($arg as xs:string?) as xs:string
```

Sample code

```
Document document = new Document();
int xml = document.parseString("<root><testdata
attr='aTrI*7'>hA1@wE</testdata></root>");
XPath oXPath = XPath.getInstance("upper-case(/root/testdata/text())");
System.out.println(oXPath.evaluateStringResult(xml));
```

Output

```
HA1@WE
```

Limitation

This function is restricted to ASCII character set.

- fn:lower-case

Returns the value of \$arg after translating every character to its lower-case correspondent.

```
fn:lower-case($arg as xs:string?) as xs:string
```

Sample code

```
Document document = new Document();
int xml = document.parseString("<root><testdata
attr='aTrI*7'>hA1@wE</testdata></root>");
XPath oXPath = XPath.getInstance("lower-case(/root/testdata/text())");
System.out.println(oXPath.evaluateStringResult(xml));
```

Output

```
hal@we
```

Limitation

This function is restricted to ASCII character set.

Optimizing XPath expression usage

While defining XPath expressions, providing an optimized path (varying according to situations) improves performance.

Follow these guidelines while defining XPath expressions:

1. Understand the search scenario completely.

2. Be specific while searching for specific nodes.

For example, if 'weather' is your first and only one node that has the name 'weather', and you are searching for the child 'forecast' inside 'weather', state `/weather/forecast` instead of `//weather1/forecast`.

3. Use 'descendant-or-self' axis instead of `//`.

If you write `//weather`, the parser expands it to `/descendant-or-self::node() /weather`, that contains the expression with three location steps:

- a. Select root.

- b. Select all descendants of it.

- c. Select all 'weather' elements from it.

You can write the expression `/descendant-or-self::weather`. This is an expression with just two steps and selects the same nodeset.

4. While searching for a particular node that is the first of such nodes inside an XML document, use `Find. findFirstMatch`, because it is the fastest search mechanism available for NOM.
5. While searching for a particular node that is the first of such nodes inside an XML document, use `Find. findFirstMatch`, because it is the fastest search mechanism available for NOM.

```
<weather xmlns:ns="http://hello.com">
    <forecast by="shilu" day="6" id="35">
        <temperature id="36" scale="F11">75</temperature>
        <humidity id="37">60</humidity>
        <temperature id="38" scale="F12">357</temperature>
        <wind id="39" scale="ns:mytype" unit="mph">33.5</wind>
        <precip id="40">Fog</precip>
    </forecast>
</weather>
```

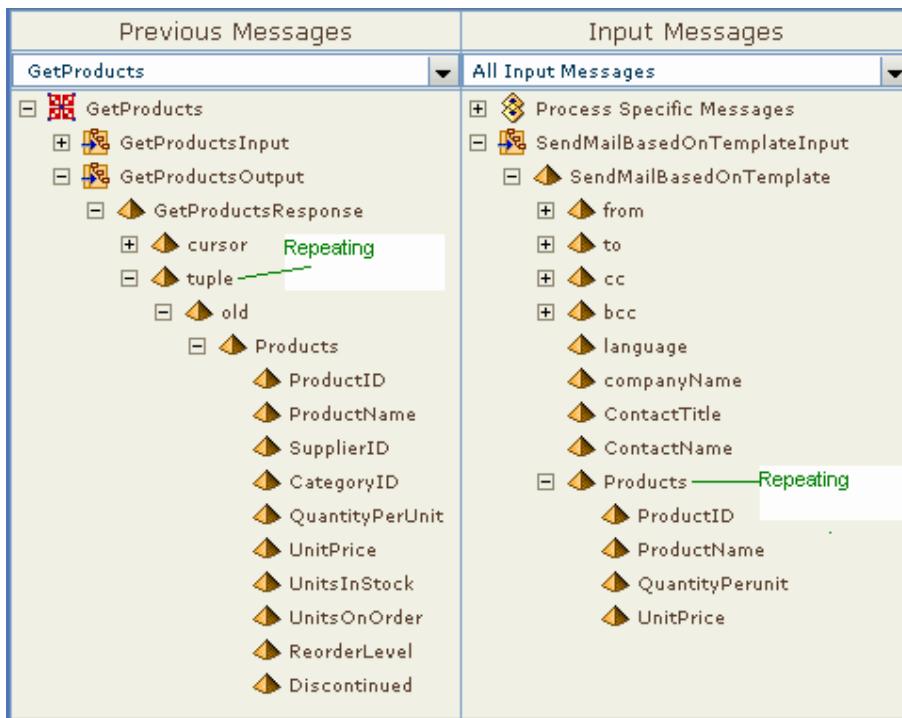
If you have to select those nodes which have 'scale' attribute (along with values) that originate from `http://hello.com`, the following expression selects those nodes.

```
"//@scale[../namespace::*[local-name(.) = substring-before(../@scale, ':') and . = 'http://hello.com']]//.."
```

Changing the XPath

The path of the group variables in the general file should be changed to the same as that of an output message of the previous activity to make the assignment for group variables. The following codeblock and figure display examples of a general file and the corresponding message map.

```
<general xmlns="">
<defaultLanguage>en</defaultLanguage>
<bodyTypeIsHTML>true</bodyTypeIsHTML>
<elements>
<variable id="companyName" path="companyName"/>
<variable id="ContactTitle" path="ContactTitle"/>
<variable id="ContactName" path="ContactName"/>
<group id="Products" path="Products"/>
<variable id="ProductID" path="Products/ProductID"/>
<variable id="ProductName" path="Products/ProductName"/>
<variable id="QuantityPerunit" path="Products/QuantityPerunit"/>
<variable id="UnitPrice" path="Products/UnitPrice"/>
</elements>
</general>
```



- In Previous Messages, the tuple element is repeated (exists multiple times) while in Input Messages, the products element is repeated. But tuple cannot be mapped to products because the underlying XPath is different.
- In the response, the tuple element will be repeated multiple times. When you map the tuple element, only the first occurrence of tuple is mapped while the remaining tuples

will be ignored. However, this issue can be offset by mapping GetProductsResponse (parent of the tuple node) and selecting all the children of the GetProductsResponse element. But an additional element is required in Input Messages to add these children. Therefore, the XPath in the general file should be changed.

To change the XPath:

1. Go to *%AppWorks_Platform_installdir%\WCP\XMLStore%Organization%\collection\templates\emails*
The general files and language files of the email services are displayed.
2. Open the general file in a text editor.
3. Change the path of the group in the general file, and start with a non-repeating element of your choice. For example, 'Marketing'.
4. The XPath of the non-repeating element is changed to<group id='Products' path='/Marketing/tuple'>
5. Change all group variables in the general file as follows:

```
<general>
  <defaultLanguage>en</defaultLanguage>
  <bodyTypeIsHTML>true</bodyTypeIsHTML>
  <elements>
    <variable id="companyName" path="companyName"/>
    <variable id="contactTitle" path="contactTitle"/>
    <variable id="contactName" path="contactName"/>
    <group id="Products" path="/Marketing/tuple"/>
    <variable id="ProductID" path="/Marketing/tuple/old/Products/ProductID"/>
    <variable id="ProductName" path="/Marketing/tuple/old/Products/ProductName"/>
    <variable id="UnitPrice" path="/Marketing/tuple/old/Products/UnitPrice"/>
  </elements>
</general>
```

The general file can contain other tags in addition to the elements tag. The element defaultLanguage represents the language code of the email (refer the Language Support section). The element bodyTypeIsHTML represents the body type of the email (this can be HTML or plain text). Thus, if the bodyTypeIsHTML element is set to 'true', the email will be sent in HTML; else it will be sent as plain text.

The XPath of the repeating element is changed in the general file.

After you complete this task:

- After editing the general file, you must regenerate the WSDL.

Constructs used in a case model

Case constructs are graphical representations used for designing a case model.

Start Case	Use Start Case  event to create a case without using states. You cannot use Initial State if you are using the Start Case event.
Activity	An activity is a step that is performed as part of case model execution. There are three different types of activities: <ul style="list-style-type: none"> ■ Human task activity ■ Business process activity ■ Sub-case activity To add an activity to the case model, click on the case model toolbox.
Activity Cluster	Activity cluster logically groups a set of activities. It is used only as a visual representation and does not have any runtime relevance. <ul style="list-style-type: none"> ■ Click  on the case model toolbox to visually group activities.
Case Model	To add a sub-case to the current case model: <ul style="list-style-type: none"> ■ Click on the case model toolbox. See Modeling Cases for more information.
State	Select  to represent a state within a case model.
State Entry Event	Represents an event that is automatically raised when entering a state. This can be used only within a state.
Initial State	To indicate the beginning or start point for states: <ul style="list-style-type: none"> ■ Click  on the case model toolbox. Start Case cannot be used when using the Initial State event. Activities or group of activities within a State are in the Initial State.
Final State	To indicate the final or end point for states: <ul style="list-style-type: none"> ■ Click  on the case model toolbox. Represents the final state (end) of a case model. When all activities or groups of activities within a State are completed they are said to be in Final State. Upon reaching this state, the case model instances are automatically closed. All case instances must have Final State. This construct can be used only when modeling states within a case model.
Connectors	Select required connector type to link activities and states appropriately from the connector list. You can align the connector text to the connector direction by right-clicking it and selecting Align Text. <ul style="list-style-type: none"> ■ Automatic follow-up connector: Link activities that can be performed automatically one after another in a sequential order. For example, when activity A is completed, activity B is

<p>connector</p> <ul style="list-style-type: none"> ■ Transition/State Transition <p>Text</p> <p>Text Annotation</p> <p>Transparent Text</p>	<p>automatically released to the inbox of the case worker. By default, the Automatic follow-up connector is applied to link activities and states.</p> <ul style="list-style-type: none"> ■ Manual follow-up connector: Link activities that need human intervention. For example, when Activity A is completed, the case worker selects Activity B for release to the case inbox. ■ Intermediate follow-up connector: Select this connector when you have linked activity or activities that need to be completed only after which the current activity can resume to completion. For example, while activity A is active, the case worker can release Activity B and Activity A is automatically suspended. Activity A automatically resumes only after completion of Activity B. ■ Transition/State Transition: Move from one state to another. In the Transition properties, you can specify the situations or conditions, based on which the transition from one state to the next state can occur. This can only be used when modeling states within a case model. <p>You cannot associate the activities that are defined after a subprocess by linking them through any follow-up other than the automatic follow-up. Such activities are released only after the subprocess is complete.</p> <p>Add an annotation with a border and a background color to the model. You can use the annotation to add and display information specific to the model.</p> <ul style="list-style-type: none"> ■ To set the properties of the annotation, right-click it and select Properties to open the property sheet of the annotation, and set the text to be displayed and the font size of the text. <p>Add an annotation that can be linked to a construct. You can use the annotation to add and display information specific to the construct. The annotation comprises a border and a connector that links it to the construct.</p> <ul style="list-style-type: none"> ■ To set the properties of the annotation, right-click it and select Properties to open the property sheet of the annotation, and set the text to be displayed and the font size of the text. <p>Add a transparent annotation to the model. You can use the annotation to add and display information specific to the model.</p> <ul style="list-style-type: none"> ■ The annotation does not contain a border or background color. ■ To set the properties of the annotation, right-click it and select Properties to open the property sheet of the annotation, and set the text to be displayed and the font size of the text.
---	--

Document Received Event	Represents an event that occurs when a certain document type event is received. It is fired when a document of a certain attachment type is attached to the case instance during execution. This will trigger all the associated activities and assign them to the appropriate targets.
Close Case	<p>Denotes the end of a case model.</p> <ul style="list-style-type: none"> ■ Click  on the case toolbox to add the construct. <p>This construct can be used only in combination with Start case construct.</p>

State properties interface

General tab

Applicability Service	<p>Often, the case workers or knowledge workers need to take decisions to choose the most applicable sub-set from next set of follow-up activities, based on the context and data available for a Case. In such scenario, it becomes necessary for a case worker to consider all the possible options and exceptions available for executing a task.</p> <p>The Applicability Service is a feature that helps a case worker to accomplish tasks easily, while considering all the possible options and exceptions by implementing the required business logic on a Web service. When the Applicability Service feature is selected for the State, it becomes applicable for all the activities within the State. When the activities within the State are executed, based on the work option or exception the Applicability Service is triggered. See Using the Applicability Service in a Case Model for more details regarding configuration and related usage.</p> <ol style="list-style-type: none"> 1. Select Business Process or Web Service from the Applicability Service list. 2. Click  to browse and select the related business process or Web service. <ul style="list-style-type: none"> ■ If Business Process is selected, then only the process models that implement the contract shared are displayed. Implementation of such business processes are based on Contract First Development option with the WSDL that is shared. ■ If Web Service is selected, then only the services that implement the WSDL are considered.
Description	Provide the description of the State

Duration tab

The duration at the state level allows to specify the due date and define escalations when the due date expires. It is applicable for all the activities within the state. If all the planned activities of the state are not completed within the time specified in the due date field, the task will be escalated to the appropriate users based on the configuration.

Use Business Calendar		While specifying the duration for the State, you may use a business calendar to ensure that the duration for the State is considered based on the working and non-working days that you have defined in the business calendar. <ul style="list-style-type: none">■ Select Use Business Calendar and click  to browse and select required business calendar from the Select business calendar dialog box.
Duration	Days	Provide the duration for the current State in number of days.
	Hours	Provide the duration for the current State in number of hours.
	Minutes	Provide the duration for the current State in number of minutes.
	Escalate to	Select a User from the Escalate to list, who must be notified if the activities within the State are aborted. <ul style="list-style-type: none">■ Manager of the User - Select this option if you want to notify the manager of the current user of the Case.■ User defined by Case variable - Select this option if you want to notify the case worker or user whose name is defined through the Case variable. When you select this option, the User field is enabled. Click to browse and select the required Case variable referring to a user from the Select a Case Variable dialog box.■ User who created the case - Select this option if you want to notify the user who instantiated the Case instance.

Annotation tab

Provide any important information about the State, which you may want to reuse later.

Transition properties interface

General tab - Group box

Description	Provide a description of the Transition.
Annotation	Provide additional notes or information that you may like to record for the

	Transition.
General tab	
Trigger transition when	<p>Select this option to enable a set of options, which you may use as a condition to trigger the state transition. The options are:</p> <ul style="list-style-type: none"> ■ All planned activities in the source state completed ■ The following activity completed ■ The event was raised from the activity ■ The following event was raised <p>This option remains selected by default, and the All planned activities in the source state completed option remains selected as default transition condition. You can skip this set of options by clearing the check box, and all the options under this section will be disabled.</p>
All planned activities in the source state completed	<p>This option remains selected by default. When you select this option, the transition from the source state to the target state is triggered only after all the planned activities within the source state are either completed or skipped.</p> <p>If you use a sub-process, either a Business Process Model or a Case model as an activity, and if you terminate that sub-process, state transition will not trigger.</p>
The following activity completed	<p>Select this option if you want to trigger the transition upon completion of a specific activity. When you select this option, the corresponding Activity list is enabled, and displays a list of activities that comprise the current state. Select the required activity from the Activity list.</p>
The event was raised from the activity	<p>Select this option if you want to trigger the transition upon completion of a specific event associated with an activity. When you select this option, the corresponding Activity and Event lists are enabled. Select the required activity from the Activity list and select the corresponding event from the Event list. When the Event is raised during execution, the activities in the source state are marked Obsolete and the transition to the target state is considered.</p>
The following event was raised	<p>Select this option if you want to trigger the transition upon completion of a specific event associated with the Case model. When you select this option, the corresponding Event list is enabled and displays a list of events that are associated to the Case model. Select the required event from the Event list. When the Event is raised during execution, the activities in the source state are marked Obsolete and the transition to the target state is considered.</p>
Trigger transition when the condition is	<p>Select this option if you want the state transition to consider a specific condition. This property is useful when you want to specify an explicit guard condition for the state transition. This option remains clear by</p>

satisfied	<p>default. You can use this option independently or along with one of the options defined with Trigger transition when.</p> <p>Click  to open XPath editor, which provides you an option to use an XPath expression as a condition for the state transition.</p>
Trigger state entry event	<p>Select this option if you want to trigger the state entry event of the target state. In this case, all the activities that are immediately followed by one entry construct are released. This property is useful when there is a requirement to configure the multiple transition paths to a state, and only one state entry point must trigger the state entry event.</p>

Note

- You can select either Trigger transition when or Trigger transition when the condition is satisfied for the state transition, or can select both to specify the state transition. You can select any possible combination for the state transition.
- If you select Trigger transition when the condition is satisfied and do not provide the XPath condition, an error is displayed as, Provide the state transition condition, while validating the Case model and in the run time no conditional selection is considered.
- If you clear both Trigger transition when and Trigger transition when the condition is satisfied options, an error is displayed as, Provide at least one of the state transition conditions, while validating the Case model, and the models will not have any means of the state transition.

Permitted activities in a transaction

You can use a short-lived process in the transaction of another business process model. That is, a short-lived process is called as a synchronous sub-process from the transaction of the main process. You can also include non-transactional activities within a group of transactional activities. For example, you can model a one-way notification that invokes a log web service.

The following activities are permitted in a transaction:

- WS-AppServer web services: By default, a WS-AppServer web service participates in a transaction. If the **Participate in Transaction** check box is cleared and the activities are a part of the transaction block, then these activities do not display transactional behavior as long as the service containers handling the web services attached to these activities run outside TomEE. If the **Participate in Transaction** check box is selected, these web services are executed regardless of where the WS-AppServer service container is running. For example, assume that a transaction has four activities - Activities A, B, C and D, out of which only Activities A and B are marked as **Participate in Transaction** and always behave as a part of the transaction. If an exception occurs during execution of Activity D, then Activities A and B are rolled back. However, Activity C, although a part of the transaction, executes as a normal web service and completes the activity if the service containers handling Activity C do not run inside TomEE.

The following table displays the rollback status of a WS-AppServer Activity when the

transaction aborts:

Service Containers Inside TomEE	Participate in Transaction	Rollback Status
✓	Any	✓
✗	✓	✓
✗	✗	✗

- In a short-lived process, the reply property in the input message for a WS-AppServer database insert/update/delete activity enables you to define whether a response is required for the SOAP request. If the reply property is set to no, the response message of this web service is empty in the message map. By default, this property is set to yes.
- Application web services: You can define non-WS-AppServer services as part of a transaction. However, these services do not display transactional behavior. All entity based web services belong to this category. An example of a non-transactional web service is obtaining information about the weather from an external source.

Note: For the application web services to participate in a transaction, ensure that the related service containers reside inside TomEE.

- Asynchronous sub-processes containing manual activities: The process is executed independently and in parallel with another process. Only the asynchronous sub-process is triggered and it runs outside the transaction.
- Synchronous sub-processes (where the sub-process is a short-lived process): In this case, the process waits for the response of the process that executes before it. You must model the synchronous sub-process within the constraints of a transaction.
- Manual activities of type Info: You can include manual activities of type Info (for example, notifications) within a transaction.

The following activities cannot be defined within a transaction:

- Manual activities of type task.
- Receive Messages.
- Delay Events.
- Synchronous sub-processes with manual tasks, Delay and Receive Message events.
- Nested transactions.

Service container properties interface

General tab

Name	Name of the service container. <ul style="list-style-type: none"> ■ The name of the application connector is the default name of
------	---

	<p>the service container. You can change it, if required.</p> <ul style="list-style-type: none"> ■ The selected application connector you are configuring is displayed in the Application Connectors group box.
Computer Name	Name of the computer on which the service container is running or on which it must be created.
Assign OS Process	<p>Enables the new service container to run in the existing JVM. When this option is selected, the Select OS Process list is populated with the names of the existing OS processes. For information about OS processes, see Configuring OS Processes for a Service Container in the <i>AppWorks Platform Administrator's Guide</i>.</p> <p>Note: If any application connector selected in the first step cannot share a JVM, this option is disabled.</p> <p>Caution: If you are configuring a WS-AppServer Service, ensure that you do not configure more than one WS-AppServer Service to a single OS Process. Doing so will result in configuration errors.</p>
Startup Type	This denotes if the service container must be started automatically or manually, when the OpenText AppWorks Platform (<instance name>) has started and when the service container stops.
Number of Attempts for Auto Start	<p>The Number of Attempts for Auto Start group box is enabled only when the startup type is Automatic.</p> <p>Determines the number of times the service container must be restarted if it becomes unresponsive. The options are as follows:</p> <ul style="list-style-type: none"> ■ Default - OpenText AppWorks Platform (<instance name>) automatically starts a service container for three times. This includes the first time it starts and two subsequent stoppages. ■ Infinite - The service container is restarted automatically whenever it becomes unresponsive. ■ Custom - In the text box, you can type the number of times you want the OpenText AppWorks Platform (<instance name>) to automatically restart the service container.
Request Notification Timeout	The time period (in milliseconds) that the service container must wait for the response.
Preference	<p>The order of preference of the service container under the Service Group to which it belongs.</p> <p>This field is displayed only when Simple Failover is chosen as the routing algorithm while configuring the Service Group.</p>
Use Default Log Settings	You can choose to continue with the default log settings. To set the logging options, clear the selection of this check box and configure the log settings on the Log Settings tab that appears. For more information on this tab, see the Log Settings table shown below.

JRE Configuration tab

The JVM settings of the **Monitor** service container are read-only because they are obtained from the `wcp.properties` file directly. Therefore, to modify the values of the JVM, you must make the relevant changes in the `wcp.properties` file.

Classpath	Specify the additional directories or JAR files for the service container classpath to start the service container successfully. Separate these entries by the OS-specific classpath delimiters. The runtime classpath for a service container is the <classpath specified on the JRE Configuration tab> + <classpath specified in application connectors configured for this service container> + <classpath specified in SYSTEM environment variable>
Set the Parameters	Click  to add the JVM Options, which will be executed when the service container is started. An invalid property may stop the service container. You can delete the JVM options by selecting their corresponding check box and clicking  . The JVM properties defined here with -D take precedence over properties defined in the <code>wcp.properties</code> file available in the <AppWorks Platform_installdir>/config folder.

Log Settings tab

Enable Logging	This option allows you to enable the log categories and severities.
Use System Policy	This option disables all the log categories and severities because the system-level configuration is applied. On selecting this option, the <code>Log4jConfiguration.xml</code> file in the <AppWorks Platform_installdir> is applied to a particular service container. By default, the Use System Policy check box is selected and other fields on the page are unavailable. To specify Logger Severities and Logger Consumers details, clear the Use System Policy check box.
Logger Severities	Root severity - Turns the log option on for all the severity levels, including those that are not listed on the screen. The severity levels selected here apply to all the categories. Tip: Enabling the lowest severity level automatically enables the remaining severity levels. However, enabling or selecting the highest severity level enables only that severity. Category-wise severities - Sets the severity for a specific category. If Root severity is selected, it turns the Category-wise severities option on for all the categories.
Logger Consumers This option is enabled only when the Use	AppWorks Platform supports various kinds of log consumers, including those from Log4j for consuming log messages. One or more consumers can be selected for publishing messages. The

System Policy option is not selected.	appropriate log consumer must be selected during the service container configuration. The following are the available options. <ul style="list-style-type: none">■ Publish to File - The log messages are written to the file specified during log configuration. The default path for this file is AppWorks Platform Installation Directory\Logs\<filename>.xml. In the absence of any file name specification, log messages are published to the console. Note: Use a double-slash for folder path specification, for example, <Drive>:\\folder1\\testlog.txt.<ul style="list-style-type: none">• Do not roll - Select this option if you do not want to roll a file.• Roll File every - Rolls over files based on the user defined time intervals, which can be monthly, weekly, daily, hourly, and for every minute.• Roll File for every - Writes the logs to a file. When the log file exceeds a specified file size, it is rolled over with a filename.n. Here, 'n' is the number of times the file has exceeded the specified size.• Roll File for every - Writes the logs to a file. When the log file exceeds a specified file size, it is rolled over with a filename.n. Here, 'n' is the number of times the file has exceeded the specified size.■ Publish to Database - The messages are written to the database.■ Publish to Remote Host - Publishes the log messages to the specified remote host. You must specify the Host Name and Port Number.■ Publish to OS based event log - Publishes log messages to either the Windows Event Log for Windows OS or Syslog with facility LOCAL7 for Linux OS.
---------------------------------------	---

Application Connector tab

Depending upon the connector configured during the creation of the Service Container, the associated connector appears on a separate tab. Click the tab to change the details.

Auditing service connections parameters interface

This topic describes the fields of the Auditing connector.

To create a database configuration:

1. From Select Database Configuration, select **New Database Configuration**.
The New Database Configuration dialog box opens.

2. Provide the required information.
3. From Advanced Options, expand the group box and provide the necessary [details](#).

To continue with an existing Database Configuration:

1. From Select Database Configuration, select **Existing Database Configuration**. The associated fields are automatically filled.
2. From Advanced Options, expand the group box and provide the necessary [details](#).

Note: Only create a Service Group for the Audit Service in a System organization.

Advanced properties

The following properties apply to JDBC connectivity.

Set Precedence to NULL	Optional. If this option is selected, the database value is set to null. In an update request, if the NULL attribute is set to 'True' and data is specified, the database value will still be null.
Support Special Characters in XML	<p>Optional. Select this option if your table names or column names contain characters that are not valid XML tag names.</p> <p>AppWorks Platform supports @, #, \$ and space as special character in table names and column names.</p> <p>The encoding or decoding of special characters in XML takes more time and this may slow down the performance of your computer. In case the database does not contain any table or field with the above four XML special characters, you can clear this check box for improved performance.</p> <p>AppWorks Platform support only @, #, \$, and space as part of table names and column names. If the table has any other special characters, which are invalid in XML names (such as / and), then the XML becomes invalid.</p>
Read Empty String as NULL	<p>Optional. This property provides you an option to read the empty strings in the table as NULL character. If you set the value true, both empty strings and NULL will be returned as NULL in the result. The following XQY XML format appears when the property is set true:</p> <pre><value null="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/></pre> <p>By default, if the property is false, AppWorks Platform distinguishes between NULL and empty strings while reading the data.</p> <p>For NULL, the XML response is: <code><value null="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/></code>.</p> <p>For empty strings, the XML response is: <code><value/></code>.</p>

Write Empty String as NULL	<p>Optional. Using this property, you can insert NULL for empty strings automatically while passing the parameter request.</p> <p>If the property is set true, while inserting or updating the database, the NULL and empty strings are considered to be same.</p> <p>If the insert/update request contains <code><value/></code> or <code><value null="true" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/></code>, NULL will be inserted or updated.</p> <p>By default, if the property is false, AppWorks Platform differentiates between NULL and empty strings while writing the data.</p>
Share Connection Pool	Optional. If this option is selected, , the same connection pool can be shared by different application connectors, which are attached to the same Service Container and use the same database configuration.
Cursor Cache	<p>Optional.</p> <ul style="list-style-type: none">■ Size - The number of cursors to be cached.■ Refresh Interval - The rate at which the cursor cache is refreshed. The interval is specified in seconds. At every interval, the cache is refreshed and all unused cursors since the previous refresh are removed from the cache.
Query Cache	<p>Optional.</p> <ul style="list-style-type: none">■ Size - The number of queries (requests sent to the back-end) to be cached.■ Refresh Interval - The rate at which the query cache is refreshed. The interval is specified in seconds. When a new request arrives and it is not available in the cache, the connector tries to allocate memory to the request from the existing cache. To accommodate this memory requirement, the cache is refreshed at every interval, and all unused queries since the previous refresh are removed from the cache. <p>The cache determines the memory requirements of the connector, and hence must be prescribed appropriately. The cache details are applicable to every connection that the user creates.</p>
Database Connections	<p>Optional.</p> <ul style="list-style-type: none">■ Read - The values in fields Minimum and Maximum indicate the minimum and maximum number of Read connections to be used by the connector to accommodate multiple Read requests.■ Update - The values in fields Minimum and Maximum indicate the minimum and maximum number of Update connections to be used by the connector to accommodate multiple Update requests. <p>To accommodate several Read and Update requests at a time, the number of connections increase until the specified maximum value.</p>

-
- **Refresh Interval** - The rate at which the Database Connections cache is refreshed. The interval is specified in seconds. At every interval, the connection pool is refreshed and all unused and idle connections since the previous refresh are removed from the cache. At any point in time, the connection pool will maintain the minimum number of connections, as specified.
-

Service group configuration interface

Description	Meaningful description of the service group.
Name	<p>Name of the service group.</p> <p>The names of the MDM Publisher and MDM Service groups must not have multibyte characters.</p>
Payload Trim	<p>White spaces in a SOAP request are considered significant and this might impact the custom application connectors with compatibility issues. Select this check box to trim all the white spaces in a SOAP request.</p> <p>Note: This feature does not work with SAML authentication because it can invalidate the cryptographic signature of the assertion. This feature is now deprecated.</p>
Request Validation	<p>Validates the SOAP request against the following schemas:</p> <ul style="list-style-type: none"> ■ SOAP schema ■ AppWorks Platform header schema ■ WSDL schema <p>Validation occurs through the following options:</p> <ul style="list-style-type: none"> ■ Protocol - The request is validated against the SOAP and AppWorks Platform protocol schemas. ■ Payload - The request is validated against the WSDL schema of the AppWorks Platform Web service operation. <p>The request is validated by selecting either or both the options.</p>
Routing Algorithm	<p>Routing algorithm for handling messages sent to the service group. The following algorithms are available by default:</p> <ul style="list-style-type: none"> ■ Simple load balancing - Requests are routed to the started service container under a service group in a round robin way. ■ Simple Failover - Requests are routed to the first started service container under a service group based on their preference number. ■ Classic - Requests are routed to the service containers under a service group randomly without checking their state. ■ Custom Algorithm - Requests are routed based on the logic

	<p>implemented in the custom algorithm.</p> <ul style="list-style-type: none"> ■ If you modify the routing algorithm, you must ensure to restart all the service containers in the service group. ■ Do not modify the routing algorithm of the single sign-on service group. ■ The Custom Algorithm currently is used only on the single sign-on service group. Adding solution-specific algorithms is not supported.
Web service Interfaces	<p>Depending on the selected application connector, the corresponding Web service interfaces are displayed along with the application package or organization, connector name (Type), and the associated namespace details.</p> <p>To attach other Web service interfaces, see Attaching Service Group to a Web Service Interface.</p>

Connection point configuration interface

The Connection Point configuration interface has the following properties.

Name	Denotes the name of the Connection Point.
Computer	Select the computer on which the service container is running.
Port	Enter the port number on which the connection point is to be created. If the port number specified is already in use by some other connection point, you are prompted to provide another port number. When the port number is specified as 0 (zero), the service container generates a valid port number when it starts. If a non zero number is specified, ensure that it falls in the correct port range. Refer to Port Ranges within AppWorks Platform for more information
Use Compression	Select this option if zipsocket is used instead of normal socket.

Service container configuration interface

General tab

Name	<p>Name of the service container.</p> <ul style="list-style-type: none"> ■ The name of the application connector is the default name of the service container. You can change it, if required. ■ The selected application connector you are configuring is displayed in the Application Connectors group box.
Computer Name	Name of the computer on which the service container is running or

	on which it must be created.
Assign OS Process	<p>Enables the new service container to run in the existing JVM. When this option is selected, the Select OS Process list is populated with the names of the existing OS processes.</p> <p>If any application connector selected in the first step cannot share a JVM, this option is disabled.</p> <p>Caution: If you are configuring a WS-AppServer Service, ensure that you do not configure more than one WS-AppServer Service to a single OS Process. Doing so will result in configuration errors.</p>
Startup Type	This denotes if the service container must be started automatically or manually, when the OpenText AppWorks Platform (<instance name>) has started and when the service container stops.
Number of Attempts for Auto Start	<p>The Number of Attempts for Auto Start group box is enabled only when the startup type is Automatic.</p> <p>Determines the number of times the service container must be restarted if it becomes unresponsive. The options are as follows:</p> <ul style="list-style-type: none"> ■ Default - OpenText AppWorks Platform (<instance name>) automatically starts a service container for three times. This includes the first time it starts and two subsequent stoppages. ■ Infinite - The service container is restarted automatically whenever it becomes unresponsive. ■ Custom - In the text box, you can type the number of times you want the OpenText AppWorks Platform (<instance name>) to automatically restart the service container.
Request Notification Timeout	The time (in milliseconds) that the service container must wait for the response.
Disable container managed transactions	<p>Since AppWorks Platform 16, it is possible to run service containers inside TomEE. To ensure transactional consistency of the various components running together in TomEE, Java EE Container Managed Transactions are used. For consistency, this same approach is used for service containers that run outside TomEE. As some older applications are incompatible with this new behavior, container managed transactions can be disabled through this check box.</p> <p>Important: Do not disable container managed transactions for service containers with application connectors that come with OpenText AppWorks Platform, with the exception of WS-AppServer.</p>
Preference	<p>The order of preference of the service container under the Service Group to which it belongs.</p> <p>This field is displayed only when Simple Failover is chosen as the routing algorithm while configuring the Service Group.</p>

Use Default Log Settings	You can choose to continue with the default log settings. To set the logging options, clear the selection of this check box and configure the log settings on the Log Settings tab that appears. For more information on this tab, see the Log Settings table shown below.
--------------------------	---

JRE Configuration tab

Note

- The JVM settings of the Monitor service container are read-only because they are obtained from the `wcp.properties` file directly. Therefore, to modify the values of the JVM, you must make the relevant changes in the `wcp.properties` file.
- The JVM settings of service containers assigned to the Application Server OS process are read-only because they are obtained from the system properties in the `CATALINA_BASE/conf/catalina.properties` file.
- The JVM settings of service containers assigned to a shared OS Process are read-only because they are obtained from the JVM Arguments as defined in the Manage OS Processes form.
- The JVM settings of service containers which are not assigned to a shared OS Process will override the default setting `bus.vm.options` as defined in `wcp.properties`.

Classpath	Specify the additional directories or JAR files for the service container classpath to start the service container successfully. Separate these entries by the OS-specific classpath delimiters. The runtime classpath for a service container is the <classpath specified on the JRE Configuration tab> + <classpath specified in application connectors configured for this service container> + <classpath specified in SYSTEM environment variable>
Set the Parameters	Click  to add the JVM Options, which will be executed when the service container is started. An invalid property may stop the service container. You can delete the JVM options by selecting their corresponding check box and clicking  . The JVM properties defined here with -D take precedence over properties defined in the <code>wcp.properties</code> file available in the <AppWorks Platform_installdir>/config folder.
Policy file name	This specifies the name of the security policy file (with location path), which contains the permissions that are granted. The file has the <code>.policy</code> extension.

Log Settings tab

Enable Logging	This option allows you to enable the log categories and severities.
Use System Policy	This option disables all the log categories and severities because the system-level configuration is applied. On selecting this option, the <code>Log4jConfiguration.xml</code> file in the <AppWorks Platform_

<p>Logger Severities</p> <p>This option is enabled only when the Use System Policy option is not selected.</p>	<p><code>installdir></code> is applied to a particular service container.</p> <p>By default, the Use System Policy check box is selected and other fields on the page are unavailable. To specify Logger Severities and Logger Consumers details, clear the Use System Policy check box.</p> <p>Root severity - Turns the log option on for all the severity levels, including those that are not listed on the screen. The severity levels selected here apply to all the categories.</p> <p>Tip: Enabling the lowest severity level automatically enables the remaining severity levels. However, enabling or selecting the highest severity level enables only that severity.</p> <p>Category-wise severities - Sets the severity for a specific category. If Root severity is selected, it turns the Category-wise severities option on for all the categories.</p> <p>AppWorks Platform supports various kinds of log consumers, including those from Log4j for consuming log messages. One or more consumers can be selected for publishing messages. The appropriate log consumer must be selected during the service container configuration. The following are the available options.</p> <ul style="list-style-type: none"> ■ Publish to File - The log messages are written to the file specified during log configuration. The default path for this file is <code>AppWorks_Platform_installdir\Logs\<filename>.xml</code>. In the absence of any file name specification, log messages are published to the console. <p>Use a double-slash for folder path specification, for example, <code><Drive>:\\folder1\\testlog.txt</code>.</p> <ul style="list-style-type: none"> • Do not roll - Select this option if you do not want to roll a file. • Roll File every - Rolls over files based on the user defined time intervals, which can be monthly, weekly, daily, hourly, and for every minute. • Roll File for every - Writes the logs to a file. When the log file exceeds a specified file size, it is rolled over with a <code>filename.n</code>. Here, 'n' is the number of times the file has exceeded the specified size. • Roll File for every - Writes the logs to a file. When the log file exceeds a specified file size, it is rolled over with a <code>filename.n</code>. Here, 'n' is the number of times the file has exceeded the specified size. ■ Publish to Database - The messages are written to the database.
--	---

- | | |
|--|--|
| | <ul style="list-style-type: none">■ Publish to Remote Host - Publishes the log messages to the specified remote host. You must specify the Host Name and Port Number.■ Publish to OS based event log - Publishes log messages to either the Windows Event Log for Windows OS or Syslog with facility LOCAL7 for Linux OS. |
|--|--|

- **Publish to Remote Host** - Publishes the log messages to the specified remote host. You must specify the **Host Name** and **Port Number**.
- **Publish to OS based event log** - Publishes log messages to either the Windows Event Log for Windows OS or Syslog with facility LOCAL7 for Linux OS.

Clone service container configuration interface

Name	Name of the new Service container
Computer Name	Computer on which the new Service container must be created.
Assign OS Process	Makes the new Service container run in the existing JVM. When this option is selected, the Select OS Process list gets populated with the names of the existing OS processes.
Port	Port number used to connect to a connection point.
Preference	The order of preference of the processor under the Service Group to which it belongs. This field is displayed only when Failover is chosen as the routing algorithm while configuring the Service Group.

Working with process event listeners

Process event listeners enable developers to write custom business logic to persist business process information in custom repositories at various events of the business process execution.

Business process information is published at the following events:

- OnProcessStart
- OnProcessComplete
- OProcess Abort
- OnActivityStart
- OnActivityComplete
- OnActivityAbort

Developers must implement the custom business logic as part of the event listener class and register the event listener class as part of the business process properties.

Process event listeners can be configured for a specific process. There can be different listener implementations for different process models or a single event listener for all the process models.

Creating process event listeners

To create a process event listener:

1. Add the required logic in a Java class implementing the `com.cordys.bpm.event.ProcessEventListener` interface available in the `bpmengine.jar` found at `<cordysInstallDir>/components/bpmengine`. See the following `ProcessEventListener` contract.

```

package com.cordys.bpm.event;
/**
 * Use this contract to execute your custom logic when a process event occurs for a
process in AppWorks Platform.
 * Configure the FQN of the class implementing this interface in the process
properties of the process model.
 * You can provide the implementation for the events that occur within a process
instance life cycle.
 *
 */
public interface ProcessEventListener
{
    /**
     * This method is invoked when a process instance is created.
     */
    void onProcessStart(ProcessInformation processInfo);

    /**
     * This method is invoked when the process instance is complete.
     */
    void onProcessComplete(ProcessInformation processInfo);

    /**
     * This method is invoked when an activity instance goes to an error state
during its execution.
     */
    void onProcessAbort(ProcessInformation processInfo);

    /**
     * This method is invoked when an activity instance of a process is created.
     */
    void onActivityStart(ActivityInformation activityInfo);

    /**
     * This method is invoked when an activity instance of a process is complete.
     */
    void onActivityComplete(ActivityInformation activityInfo);

    /**
     * This method is invoked when an activity instance results in an error during
execution.
     */
    void onActivityAbort(ActivityInformation activityInfo);
}

```

Parameter classes

```
package com.cordys.bpm.event;

import javax.transaction.SystemException;
import javax.transaction.Transaction;
import org.w3c.dom.Element;
import com.eibus.localization.exception.custom.LocalizableIllegalStateException;
/***
 * This contract provides the APIs that are available to the event listeners as
parameters.
 */
public interface ProcessInformation
{
    /**
     * This method returns the ID of the current instance.
     */
    String getInstanceID();

    /**
     * This method returns the parent instance from where the current instance is
created.
     */
    String getParentID();

    /**
     * This method returns the root instance ID from where the current instance is
created.
     */
    String getRootID();

    /**
     * This method returns the process model ID from where the current instance is
created.
     */
    String getProcessModelID();

    /**
     * This method returns the organization DN under which the current instance is
created.
     */
    String getOrganization();

    /**
     * This method returns the process model name whose instance is created.
     */
    String getProcess modelName();
}
```

```
/**
 * This method returns the status of the current instance.
 */
String getStatus();

/**
 * This method returns the user DN of the user who last worked on the current
instance.
 */
String getCurrentOwner();

/**
 * This method returns the start time of the current instance in milliseconds.
 */
long getProcessStartTime();

/**
 * This method returns the last updated time of the current instance in
milliseconds.
 */
long getProcessLastUpdatedTime();

/**
 * This method returns the message map element of the current instance, which is
updated with the input and output message of each activity in the process model.
 */
Element getMessageMap();

/**
 * This method returns the error text of the current instance if an error
occurs.
 */
String getErrorText();
}

package com.cordys.bpm.event;
/**
 * This contract provides the APIs that are available to the event listeners as
parameters.
 */
public interface ActivityInformation
{
    /**
     * This method returns the activity ID.
     */
    String getActivityID();
    /**
     * This method returns the iteration count of the activity execution.
     */
}
```

```

 * This method returns the execution userDN of the current activity. This value is
available only after the activity is complete.
 */
String getExecutedBy();
/**
 * This method returns the {{IProcessInformation}} object.
 */
ProcessInformation getProcessInfo();
}

```

The ProcessInformation and ActivityInformation class instances are provided as inputs to the listener methods that hold the process and activity details.

2. Create a Java archive of the process event listener implementation.

When configuring and implementing process event listeners, ensure the following:

- The event listener implementation neither influences process engine execution nor modifies the business process runtime data. The exceptions thrown from the listener implementations are ignored by the process engine.
- The event listener implementation must avoid resource intensive processing logic and reading from the Platform repositories to ensure there is no performance drop in the process execution.
- Check for memory leaks, thread safe issues, errors, and so on in the event listener implementation before configuring them for the process models.

Configuring process event listeners

To configure a process event listener:

1. Add the absolute path of the JAR file containing the process event listener implementation classes to the Business Process Management service container class path. For more information, see the JRE section in the Service Container Properties Interface topic.
2. On the **Properties** tab of the process model, type the fully qualified name of the event listener implementation class.
3. Publish the business process model to associate the process event listener to the process model.

Example

See the following sample code that demonstrates how to implement a Java class using ProcessEventListener and specify the custom logic in the methods. It uses Active MQ for its logic.

TestProcessListener.java

```
package com.test;

import java.util.Properties;
import javax.jms.JMSEException;
import javax.jms.MessageProducer;
import javax.jms.TextMessage;
import javax.jms.XAConnection;
import javax.jms.XAConnectionFactory;
import javax.jms.XASession;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.transaction.NotSupportedException;
import javax.transaction.RollbackException;
import javax.transaction.SystemException;
import javax.transaction.Transaction;
import javax.transaction.xa.XAResource;
import org.apache.activemq.command.ActiveMQQueue;
import com.cordys.bpm.event.ActivityInformation;
import com.cordys.bpm.event.Transactional;
import com.cordys.bpm.event.ProcessEventListener;
import com.cordys.bpm.event.ProcessInformation;

public class TestProcessListener implements ProcessEventListener
{
    private ActiveMQQueue queue;
    private MessageProducer listenerQueue;
    private Context context;
    private XAConnectionFactory connectionFactory;
    private XASession session;

    private static final String TRANSACTION_MANAGER_FACTORY_CLASS =
"com.cordys.xatransaction.atomikos.AtomikosTransactionManagerFactory";
    private static final String JNDI_FILE_PROPERTY_NAME =
"com.cordys.transport.jms.jndifile";
    private static final String JNDI_FILE_PATH = System.getenv("CORDYS_HOME") +
"\\"config\\reliableMessaging.properties";

    private static final String CONTEXT_FACTORY_CLASS =
"org.apache.activemq.jndi.ActiveMQInitialContextFactory";
    private static final String TRANSACTION_PROVIDER_URL = "tcp://localhost:61616";
    private static final String CONNECTION_FACTORY_NAME = "QueueConnectionFactory";
    private static final String LISTENER_QUEUE_NAME = "ListenerProcessQueue";

    public TestProcessListener() throws NotSupportedException, SystemException,
```

```

NamingException, JMSEException, IllegalStateException, RollbackException
{
    // Setup transaction properties
    System.setProperty(TransactionManagerFactory.class.getCanonicalName(),
TRANSACTION_MANAGER_FACTORY_CLASS);
    System.setProperty(JNDI_FILE_PROPERTY_NAME, JNDI_FILE_PATH);
}

private void enlistQueueResource() throws NotSupportedException,
SystemException, NamingException, JMSEException, IllegalStateException,
RollbackException
{
    // Create a JNDI API InitialContext object
    Properties props = new Properties();
    props.setProperty(Context.INITIAL_CONTEXT_FACTORY, CONTEXT_FACTORY_CLASS);
    props.setProperty(Context.PROVIDER_URL, TRANSACTION_PROVIDER_URL);

    context = new InitialContext(props);
    queue = new ActiveMQQueue(LISTENER_QUEUE_NAME);

    connectionFactory = (ConnectionFactory) context.lookup(LOCATION_FACTORY_
NAME);
    Connection connection = connectionFactory.createXAConnection();
    connection.start();

    session = connection.createSession();
    listenerQueue = session.createProducer(queue);
}

private void addMessageToTransaction(String messageText)
{
    try
    {
        if (listenerQueue == null)
        {
            enlistQueueResource();
        }
        TextMessage message = session.createTextMessage(messageText);
        listenerQueue.send(message);
    }
    catch (JMSEException | IllegalStateException | NotSupportedException |
SystemException | NamingException | RollbackException e)
    {
        throw new RuntimeException(e);
    }
}

@Override

```

```

mException sysException)
{
    throw new RuntimeException(sysException);
}

public static void main(String[] args) throws Exception
{
    TestProcessListener listenerTest = new TestProcessListener();
    listenerTest.addMessageToTransaction("Hello World!");
}
}

```

Business object instance interface

There may be situations where you want to modify the values of an instance and then manually trigger an event. You may edit the business object instances and manually trigger an event through the Business Object Instance interface. The interface lists the business object instances in a particular state. It displays attributes of the instance under **Column**, **Value**, and **Original Value** columns.

The fields on the Business Object Instance Interface are as follows.

 (Refresh)	Click to refresh the Business Object Instance window.
<Instance name (row)>	<ol style="list-style-type: none"> Select an instance (row) to edit or modify attribute values of the business object instance. Under the Value column, edit the attribute value.
Available Operations -> <button that represents the event>	<ol style="list-style-type: none"> Click the button, which represents the event that you want to trigger. All the possible transitions, that you have defined while designing a state model appear under the Available Operations section. The event is fired on the business object instance with the attribute values from the Value column. If you do not edit or modify the attribute value under the Value column, the event is fired on the business object instance with the attribute values from the Original Value column. The business object instance with the new values moves into a state that is a result of the event triggered. A message appears that the event is triggered successfully. The next events, if they exist, appear as buttons.

Business object instance state history interface

An instance can go into any state depending on the events that have occurred. The State History interface displays the state history of a business object instance.

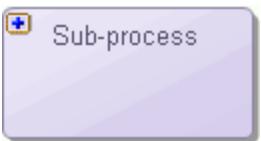
The Tool bar Operations on the Business Object Instance State History Interface are as follows:

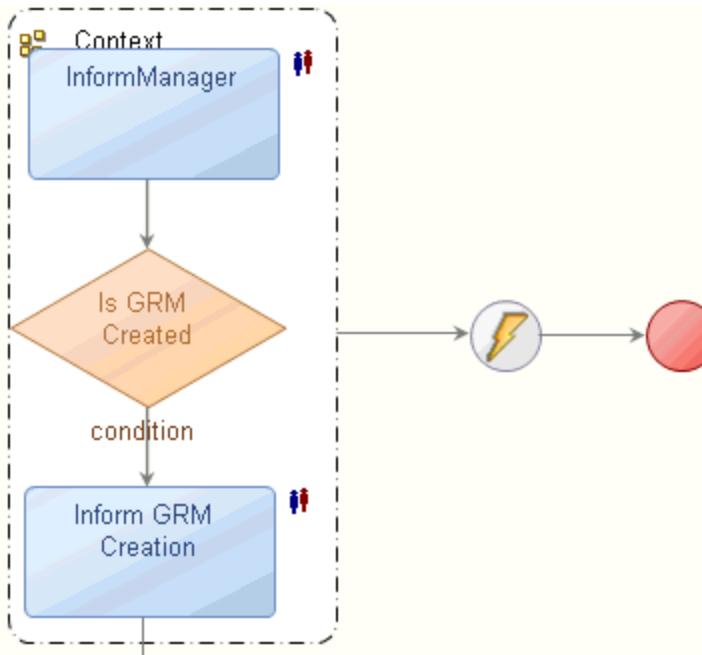
 (Refresh)	Click to refresh the State History window.
 (Compare)	<ol style="list-style-type: none"> Select the check boxes corresponding to two or more instances (rows) and click  (Compare). The State Differences View window appears. When an instance goes into a state, the values of the instance may change. The State Differences View window displays the modifications of the instances made by the states selected. <p>The changes made by a particular state are marked in red.</p>

Mapping of BPML to BPMN constructs

The following table describes how each BPML construct maps to the corresponding AppWorks Platform BPMN construct.

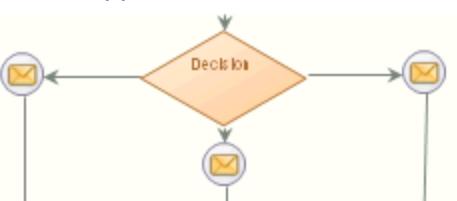
BPML	BPMN construct
Catch (Fault Handler)	Each Catch in the fault handler maps to Exception in BPMN. 
Empty	Empty maps to empty activity in BPMN. 
Invoke	<ol style="list-style-type: none"> If Invoke is used to call a Web service, it maps to an activity of type Web service in BPMN.  If the Port Type and Operations match with the Receive activity with create instance property set to 'yes', then it is modeled as end with message in BPMN.

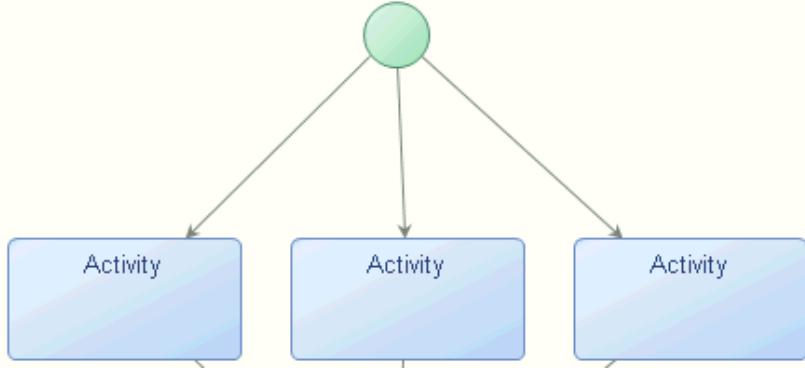
BPML	BPMN construct
	 3. Else, it maps to Send Message Activity of BPMN or it could be a sub-process call.  or 
Receive	1. If the BPEL starts with a Receive activity, with the create instance property set to yes, then it will map to start with trigger type message in BPMN.  2. Any subsequent usage of the Receive activity in the BPEL is interpreted as a receive message in BPMN. 
Reply	<p>If the 'create instance' property is set to yes, then it should be the first activity in the BPEL.</p> <p>A Reply activity is used to send a response to a request previously accepted through a receive activity. Such responses are meaningful only in synchronous interactions.</p> <ol style="list-style-type: none"> 1. The Reply activity exists if the process is being modeled as a synchronous process. 2. If the Port Type and Operations match with the receive activity having the property 'create instance' set to yes, then it is modeled as end with message in BPMN. 3. Else, it maps to Send Message event of BPMN. 

BPML	BPMN construct
	 <p>The Port Type and Operations are generated automatically during the BPML generation, so it is not possible to have the same port type and operation modeled in the BPEL Reply activity.</p>
Scope	<p>Scope maps to Context in BPMN.</p> 
Sequence	<p>Sequence maps to Connector in BPMN.</p> <p>↓</p>
Switch	<p>Switch maps to Decision in BPMN.</p> 
Terminate	<p>Terminate maps to End event of type none.</p> 
Throw	<p>Throw maps to End event of type error.</p>

BPML	BPMN construct
Variables	 Variables are mapped to Process Specific Messages. 
Wait	Wait maps to delay event in BPMN. 
While	Only Duration is supported and should be of type expression. While maps to While loop in BPMN. 

The following constructs, if found in the source BPEL, will not be imported by the Content Transfer Utility. A log file is generated containing the details of the constructs not imported.

BPML	BPMN Construct
Compensate	Compensate is mapped to end of type rollback in BPMN. 
Compensate Handler	Compensate is mapped to Compensate in BPMN.
Partner Link	No mapping exists.
Pick	Pick is mapped to Choice in BPMN. 

BPML	BPMN Construct
Flow	<p>Flow is mapped to activities modeled in parallel in BPMN.</p>  <pre> graph TD Start(()) --> Activity1[Activity] Start --> Activity2[Activity] Start --> Activity3[Activity] </pre>
Assign	<p>Assign is mapped to Assign in BPMN. A business analyst has to manually create the equivalent message mapping in the business process modeling environment.</p>

Modeling atomic and open-ended transactions

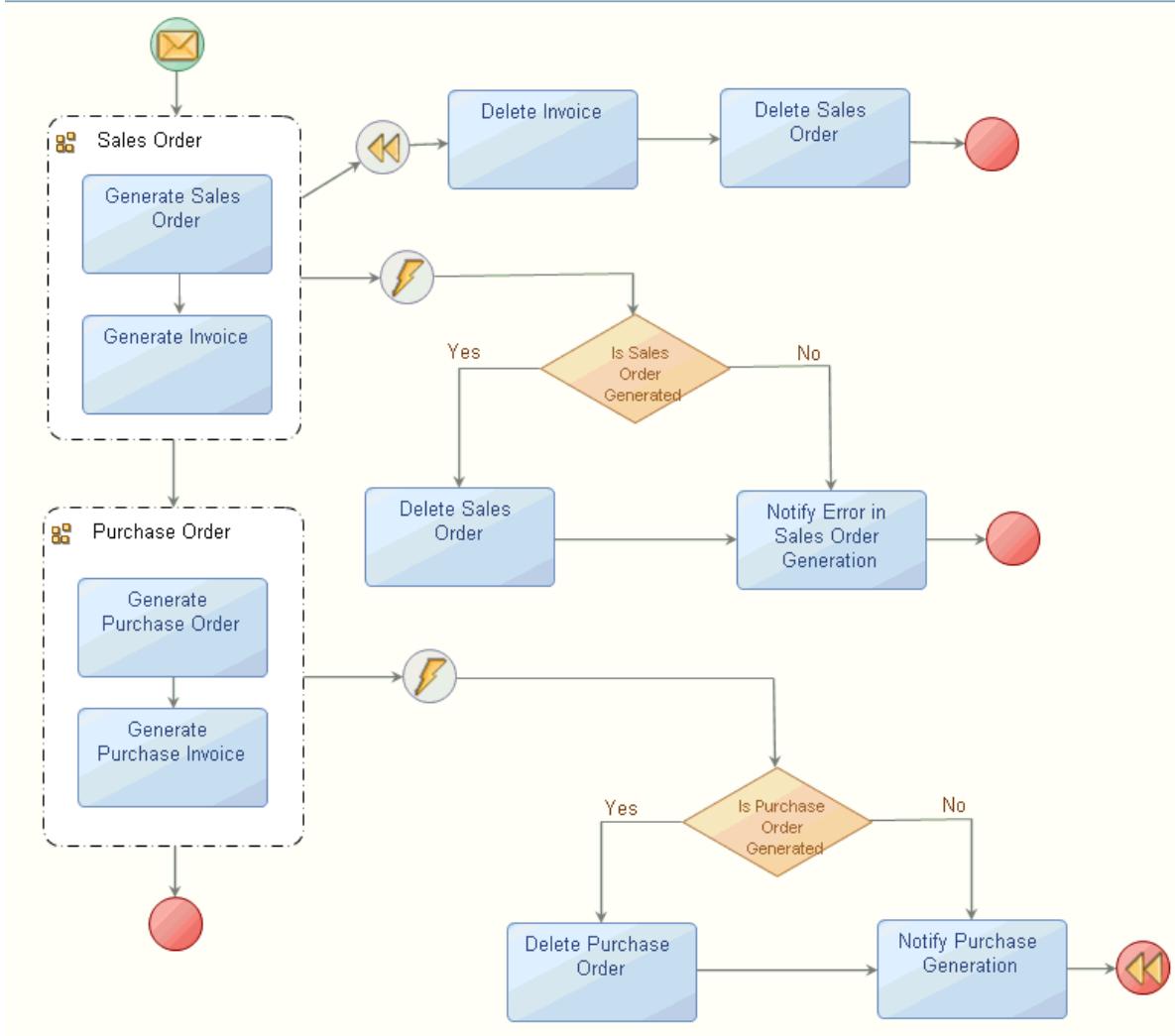
Transactions can be modeled as atomic transactions or open-ended transactions in a business process.

An atomic transaction is an explicit transaction using the Transaction group construct. It must be modeled using the WS-AppServer or application web services. This is especially useful when the duration of the process is very short. All activities in an atomic transaction will be rolled-back which participate in the transaction.

An open-ended transaction can be modeled in a long-lived process using the Compensate and Exception events to bring a transaction flavor to the business process. Long running transactions are often used where the process requires human intervention and where interactions span major organizational boundaries.

You can implement transactional behavior in long-lived processes using crash recovery, and Exception and Compensate events. In a short-lived process or process fragments (processes that do not have human tasks or Receive Message events, you can group activities into a transaction group with rollback recovery. Any exception occurring within the transaction context automatically aborts the transaction. You can attach Exception events to a transaction context to notify the user of the failure of the transaction.

Exception handling and recovery of a consistent transaction state is explicitly modeled in a business process diagram using Exception and Compensate events.



For example, when an order is received, the order confirmation (sales order) and invoice are generated for the customer, and then a purchase order is issued to the supplier.

A Context group construct groups the Generate Sales Order and Generate Invoice steps, to which Exception and Compensate events are attached. An exception occurring in either step is caught by the Exception event, which manages the fault handling logic, and the process is terminated.

A similar type of fault handling occurs in the Purchase Order context, except that instead of terminating the process, the fault handling flow ends in a compensate end event. This is because the fault handling logic did not undo (compensate) the completed sales order steps. The compensate end event is caught by the compensate event in the sales order context, which then executes the modeled compensation flow. Thus, the entire order is reverted irrespective of where the unrecoverable error occurs.

Model-View-Controller

The Model-View-Controller (MVC) is an architectural design pattern that defines separate functional components for a program. It is based on the functional division of a program by data store, presentation, and execution logic.

The MVC design components are defined as follows:

- **Model:** Manages the data store and responds to inputs from the View and Controller components based on the business logic; seldom changes
- **View:** Comprises the user interface layout and presentation details; may change frequently
- **Controller:** Manages and synchronizes model and view.

The execution of a web application seldom changes as it is based on the business logic. In a typical web application, even a minute change in the user interface may bring about changes in its business logic, making it necessary to retest the application.

Using the MVC design pattern, you can divide an application program into standalone components with minimal interdependency. Such a division ensures that changes in a component has minimal impact on the other components, and enables the individual testing of each component. It is especially useful for web applications, where the user interface of a page may change frequently, as compared to the data store and execution logic.

Web applications based on MVC are also easier to customize for multiple devices such as mobiles, palm tops, and desktop computers, as the customization requires changes in the View component only. Thus, the development effort for the MVC components can also be divided based on specialized skill sets.

It is also possible to base different Views on the same Model so that a common set of data is used in multiple pages of a web application.

Chapter 24

Collaborative Workspace

Collaborative Workspace (CWS) enables you to develop business solutions on the AppWorks Platform. CWS offers you a collaborative, Web-based, and model-driven environment enabling business and IT to jointly create, deliver, and deploy business solutions rapidly and incrementally. Additionally, CWS provides a single view on all content of your business solution through workspace isolation enabling you to test changes before sharing these with your team.

You can do the following using CWS:

- [Validate a project](#)
- [Publish a project](#)
- [Synchronize workspaces](#) and [resolve synchronization conflicts](#)
- [Share your changes with others](#), [revert your changes](#), and [update your workspace with changes of others](#)
- [Create and download application packages](#)
- [Set properties of an application package](#)
- [View and modify the properties of a workspace](#), [solution](#), and a [project](#)
- [Reload runtime references](#)

When you modify a document from CWS, the document is locked in the source control management system. Actions such as validating or publishing documents or synchronizing workspaces always produce log files. These action-specific log files are stored in the <AppWorks Platform_installdir> in the folder `Logs/cws` and are automatically cleaned up after four days. Any error that is identified during any of these actions is also presented in the progress dialog box in the browser. Other system errors that occur during such actions, which are not caused by invalidity of application sources are reported in the log file of the Collaborative Workspace service container. This log file can be found in the folder `Logs` and is not be cleaned up automatically.

Validating a project

A project may contain several documents that are required by an application. Validating is a way of checking the entire design-time content in a project to determine its suitability to be used at run time. By validating a project, you generate the build output (run-time content)

for the project and its contents. Whenever you make changes to the project, you need to validate it again so that its contents are available for use at run time.

At any point, if you want to get a fresh build of a project, you can clean the build of the workspace, and validate the project afresh. You can do this using the  (Clean Build Folder) option on the Workspace Documents toolbar.

To validate a project:

1. In the **Workspace Documents**, open **<solution> > <project>**.
2. Right-click the **<project>** and select **Validate**.
The validation process starts in a separate window, which also displays the validation status.
If you validate the project in Classic View, the process status is displayed in a pane to the right of Workspace Documents.
A successful validation is indicated by  whereas errors are indicated by .
3. To know the validation details, click **Details**.
The details appear in the Details tab.
4. After you make corrections, click **Restart** to validate the project again.

Adding existing projects to a solution

Projects available within a workspace are reusable. It means, you can use the same project in any number of solutions that you create in a workspace. You can use an existing project irrespective of whether you or some other developer in the team created it. Reusing projects minimizes the development effort.

To reuse a project, you add it to the solution you are creating:

1. On the toolbar of Workspace Documents (Explorer), click  and select the solution to which you want to add a project.
The selected solution and its contents appear in the workspace.
2. Choose one of the following ways to add a project:
 - a. Right-click the **<solution>** and select **Add > Existing Projects**.
The Solution window appears displaying the list of projects that you can add to the solution. The list does not display the projects that are already a part of the current solution.
 - b. Select the check box next to the project(s) that you want to add and click **Finish**.
 - c. On the toolbar of Workspace Documents (Explorer), click  > Project List.
The Project window opens displaying the list of projects available in the workspace.
 - d. Select the check box next to the project(s) that you want to add.
The  icon on the toolbar of the Project window is enabled.
 - e. Click  (Add to current solution).
The window displaying the list of projects closes.

The selected projects are added to the solution you are developing. You may modify the existing contents of that project or add new documents to it.

Validating a document

Before a document is published, the platform validates whether it satisfies certain criteria. The validation of a document is triggered automatically when you publish the document. You can also trigger the validation of a document yourself. When validating a document, the platform ensures that all other documents on which the document depends are also validated.

You can validate a document in one of the following ways.

To validate a document from Workspace Documents:

1. In **Workspace Documents**, open <solution> > <project> > <document>.
2. Right-click the <document> and select **Validate**.

The validate process starts in a separate window, which also displays the validate status.

If you validate the document in Classic View, the validate process status is displayed in a pane to the right of Workspace Documents.

To validate a document using the Quick Access Menu on an editor:

1. On the top-left corner of the editor, click  (Quick Access Menu) > Validate.
A link (Click here to validate) appears in the space to the right of the Quick Access Menu.
2. Click the hyperlink.
The validate process starts and the status is displayed at the same location.
A successful validation is indicated by  whereas errors are indicated by .
3. To know the validation details, click **Details**.
The details appear in a Details tab.
4. After you make the corrections, click **Restart** to validate the document again.

Inserting related documents and exploring workspace contents

While working on a document, it may be necessary to insert or attach another related or dependent document available in the same workspace. For example, while creating a business process model, you may want to use a Web service operation in it to provide a specific functionality or attach a role to a specific activity. You may also want to browse the workspace to look for some other document or to determine the location where you want to save the current document. In any of these cases, you will need a conveniently placed interface that provides an easy option for document integration and provides a quick view of the workspace contents. To support this required functionality across AppWorks Platform,

the Insert and Workspace AppPalettes are introduced in modelers. The AppPalettes (represented by tabs) appear within a frame, to the left of each modeler or editor.

The Insert AppPalette displays the relevant documents that you could use in the current document. The documents would be grouped per document type, for easy identification and insert. You can drag and drop the document on to the modeler or editor.

The Workspace AppPalette provides a tree view of the entire workspace. It renders the same view as seen in Workspace Documents, but within a slender frame. You could browse the tree in search of projects and folders. Positioning this AppPalette within all modelers/editors is likely to reduce the effort you spend in switching back and forth between the modeler or editor and the Workspace Documents, whenever you need to look at the Workspace contents.

You cannot create new documents in the Insert or Workspace AppPalettes. To create new documents, you have to use the Quick Access Menu. For details about using these AppPalettes in your modeler or editor, see the topics associated with it.

If you happen to close the tabs and want them to re-appear in the frame:

- Click  on the toolbar, and select the name of the AppPalette to display it.

For 1-to-1 relations between documents, modelers often provide a zoom button that opens a generic lookup window. This window lists all the documents that could be candidates for the relation. By default, the documents are filtered by document type. In specific cases, only certain instances of the respective document types can be candidates for the relation. In that case, the title of the generic lookup window includes the postfix suffix "(Filter Applied)". An example in which an additional filter is applied is when looking up a role to which a human activity of a business process is to be assigned. In that case, only functional roles are listed; non-functional roles are filtered out.

Locating an existing document in the workspace

If you are working in the Workspace Documents (Explorer) view, you clearly see all the projects and documents the way they are structured. However, if you are working in the Workspace Documents (My Recent Documents) or Workspace Documents (Tag Search) view, you will see all the documents available in the workspace but cannot determine where they are located and how they are organized in the workspace.

The Locate in Explorer option helps you to see where a document is positioned in the workspace. When you use this option, the view changes from Workspace Documents (My Recent Documents) or Workspace Documents (Tag Search) to Workspace Documents (Explorer).

To locate an existing document in the workspace:

1. On the Workspace Documents (My Recent Documents) or Workspace Documents (Tag Search) view, point to the specific document and click . A menu list appears.
2. Click **Actions > Locate in Explorer**. The Workspace Documents switches to Explorer view, highlighting the document in the tree structure.

Managing an application package

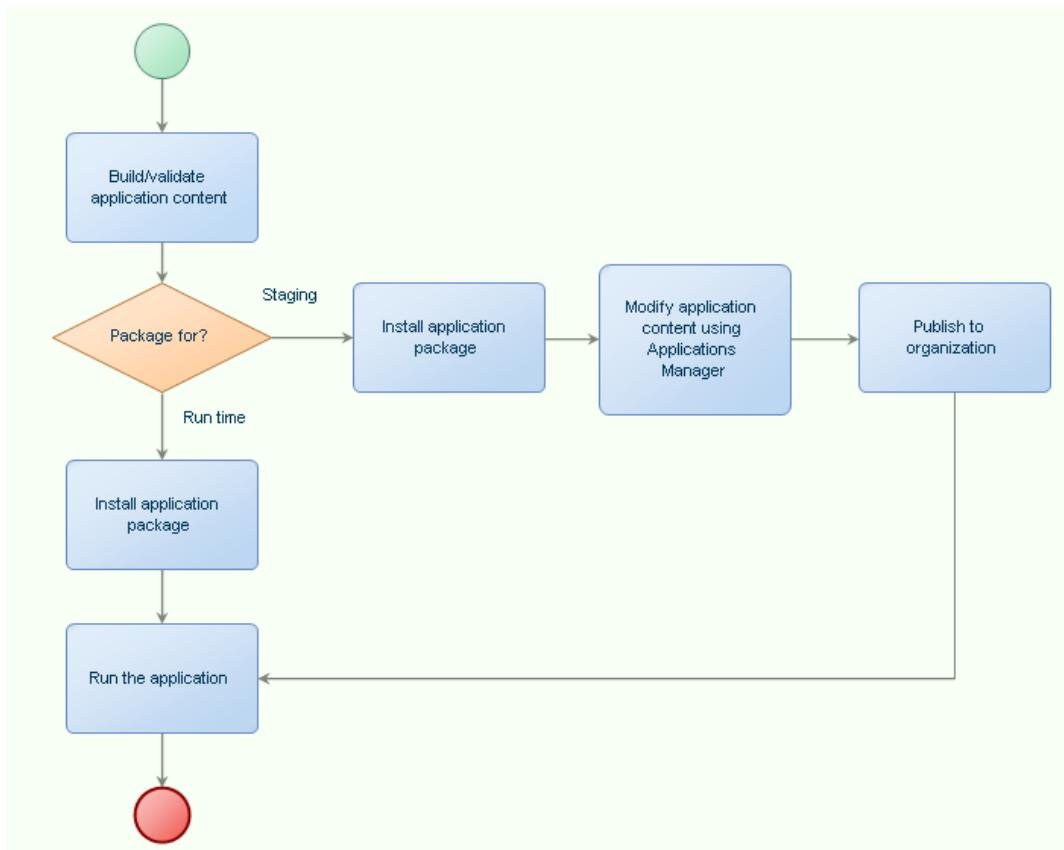
AppWorks Platform provides dual options to package application content. You may either package application content that could be directly run as an application or package it such that minor modifications can be made to it at run time, before publishing it for actual use. Depending upon your choice, you can create the package by selecting the relevant option in the properties of the Application Package. This topic provides information on the application package that can be modified at run time.

You may need to modify application content during application deployment, depending upon the production requirements. To enable modifications, AppWorks Platform provides a feature called Applications Manager. Using Applications Manager, you can modify the contents of existing documents or add new documents (configure your application) within the organization where the application is installed. You can also delete the newly added documents, if necessary. However, you cannot do the following:

- Modify the original structure of project by moving documents or folders
- Rename any folder or document that belongs to the original application package
- Delete any folder or document that belongs to the original application package

This restriction is to retain the original application content because that serves as the core of an application. Applications Manager reuses the editors for viewing and editing documents in their respective editors and republishes the modified versions to run time.

The following illustration takes you through the process of packaging content (options available in [Package Properties](#)), and how the packages are handled in both scenarios.



Modifying the contents of an installed application package

An application package for staging enables you to modify its content in an organization before it is published to run time. Usually, this activity happens on a test or production server where the installed content can be modified to a certain extent before it is used. For example, modifying a business calendar, adding calendar exceptions, changing the organization model, modifying the contents of a decision table, adding or modifying BAM KPIs and business measures.

As part of modifications, you can edit the contents of existing documents or add new documents. You can also delete the newly added documents, if necessary. However, you cannot do the following:

- Modify the original structure of project by moving documents or folders
- Rename any folder or document that belongs to the original application package
- Delete any folder or document that belongs to the application package

Before you begin:

- You must have created the application package for staging, bearing the file name (<owner><project name><version><build>.staging.isvp) and it should have been

loaded in AppWorks Platform.

- You must have the role of CWS User to modify the contents and the role of CWS Application Administrator to publish the content after modifications.

To modify the contents of an installed application package:

1. On My Applications, click  (Applications Manager).
The Applications Manager window opens.
2. Create a solution and add the loaded application to it as a project.
All the loaded staging application packages will appear as existing projects. The staging application package is added as a project to the solution.
3. Modify the contents of the project in the following ways:
 - Edit existing documents in the respective editors.
 - Create new folders and new documents and organize them in the project. The project is modified.
4. Click .

The loaded application package is modified.

Publishing a modified application to run time

After you update an installed application, you must publish it to run time so that it can be used. Without publishing, the changes are not available to the end-user.

Before you begin:

You must have the role of CWS Application Administrator.

To publish a modified application to run time:

1. On My Applications, click  (Applications Manager).
The Applications Manager window opens.
2. Open <solution>, right-click a <project> and select **Publish to Organization**.
You can also right-click <solution> and select Publish Projects to publish all the projects to organization. The publish process starts in a separate window, which also displays the publish status.
If you publish the project in Classic View, the publish process status is displayed in a pane to the right of the Applications Manager.
A successful publish is indicated by  whereas, errors are indicated by .
3. To know details of the published project, click **Details**.
The details appear in a Details tab.
4. After making corrections, click **Restart** to publish the project again.

Managing dependencies between documents

A project can contain several documents such as Business Process Models, Worklists, Web services, User Interfaces, Roles, Users, Organizations Units, and so on. For an application to run successfully, these documents need to function coherently and have clearly defined connections. For example, a Business Process Model can support integrating Roles, Tasks, XForms, and Web services in itself. When the Business Process Model is executed, all these different documents are invoked and displayed to the user for necessary user interaction and completion of an activity. Therefore, it is important to understand the dependencies between documents and link them appropriately. Thus, you can ensure that the application is complete in all respects and functions as per the business need.

Establish dependencies in two ways:

- Using the [Quick Access Menu > Insert](#) option on a document.
- Dragging and dropping one document on to another.

For example, see how [tasks are associated with roles](#). Similarly, you can associate a [Business Process Model with XForms](#), [Web service with a Business Process Model](#), or [Task with a Case Model](#), and create several such combinations depending upon your business requirement. It is important to keep a track of these dependencies and update the related documents whenever you modify or update a document.

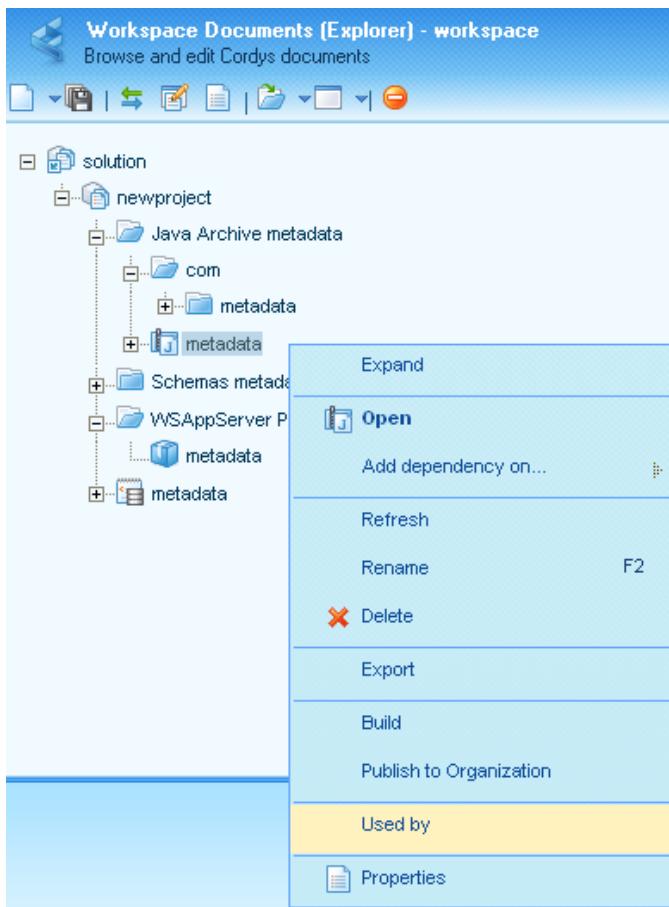
Note: Failing to update dependent documents may affect the functionality of the application. Changing a child document and publishing it will not publish the associated documents where it is used. However, it may not always be possible to remember the connections and dependencies. To help you with such information, CWS provides the Used By feature for each document that you create. It displays the documents being used or referred from, along with their locations. You can use this option either from the Quick Access Menu on the document's editor or in Workspace Documents (Explorer), right-click a document in and select Used By. You will be able to see the dependencies.

Example

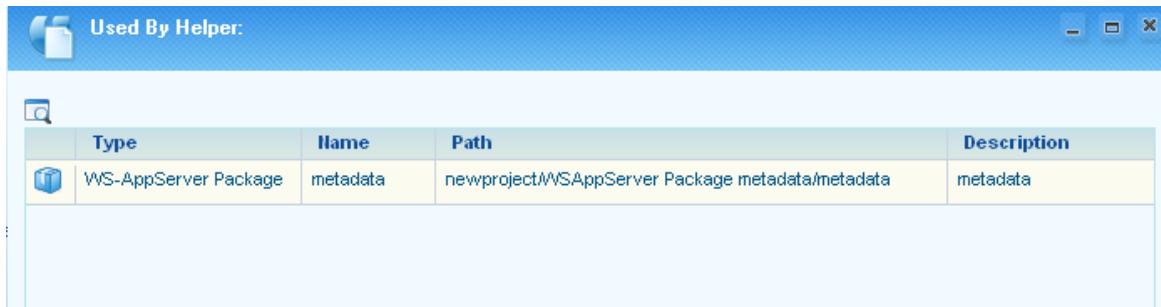
You have created a WS-AppServer Package, and in that process, you have created a JAR. When you use this option on the JAR to see where it is used, it shows you the name of the WS-AppServer package.

The following images illustrate the example.

1. Right-click  (Java Archive Definition) and select Used by.



The Used By Helper window opens displaying the name of the WS-AppServer Package.



2. If you want to locate the displayed document in the Workspace Documents (Explorer), select the document and click .

The tree in Workspace Documents (Explorer) expands to the related level and displays the document with its name highlighted.

Performing miscellaneous common tasks on application content

AppWorks Platform provides options to perform some basic tasks that are common to any document, project, or a solution. You will be able to perform these tasks or view the details from Workspace Documents. The following table lists the options through which these tasks can be accomplished.

Option / Feature	Purpose / Description	Available in Workspace Documents (Explorer)?	Available in Workspace Documents (My Recent Documents)?	Available in Workspace Documents (Tag Search)?
Expand	Expands a project or folder and displays contents.	YES	NO	NO
Collapse	Collapses an open folder or a project and hides the content	YES	NO	NO
Refresh	Refreshes the selected item to reflect any recent modifications to it or contents.	YES	YES	YES
Rename	Renames the selected item.	YES	YES	YES
Delete	Deletes the selected item.	YES	YES	YES
Remove from Solution	Removes the project from the solution.	YES	NO	NO
Used By	Provides information on the dependency that any other document has on this document. This is available only at the document level.	YES	NO	NO
Properties	Displays the properties of the selected item, some of which can be edited.	YES	NO	NO

To perform these tasks in the Workspace Documents (Explorer):

- Right-click at any level - the **<solution>**, **<project>**, or a **<document>** - and click the required option.

To perform these tasks in the Workspace Documents (My Recent Documents) or Workspace Documents (Tag Search):

- Place mouse pointer over the particular document, click  and click the required option.
All the options may not be available when you are in the My Recent Documents or Tag Search view.

The selected application content is viewed, modified, or deleted.

Publishing a project to an organization

When you publish a project, all its documents are validated and then published to the run time. By default, documents are published to the run time of the organization in which you develop the application. If required, you can change it in Workspace Properties.

If you experience a slow publish activity

If you experience a slow publish activity, it may be because AppWorks Platform internally keeps track of all the publish activity which you can use for later reference, for example, changes done on a business process model on every publish.

To enhance performance you can instruct AppWorks Platform not to preserve the information of each publish activity:

- Set the property `development.publish.to.psl.enabled` to **false** (default value is **true**) by including the following line in the `cws.properties` file located at `<AppWorks Platform_installdir>\<instance name>\components\cws\config`:

```
development.publish.to.psl.enabled=false
```

When you set the property to false, AppWorks Platform will not have information related to instances created on older versions of the respective documents, such as process instances of business process models. As a result, the instance information and the contents of the recently published document will not tally.

- To restore the publish mode, set the value of the `development.publish.to.psl.enabled` property to **true**.

To publish a project to an organization:

- In the **Workspace Documents**, open **<solution> > <project>**.
- Right-click the **<project>** and select **Publish to Organization**. Alternatively, you can also right-click **<solution>** and select Publish Projects to publish all the projects within a solution.

The publish process starts in a separate window, which also displays the publish status.

If you publish the project in Classic View, the publish process status is displayed in a pane to the right of the Workspace Documents.

A successful publish is indicated by whereas errors are indicated by .

3. To see details of the published project, click **Details**.
The details appear in a Details tab.
4. After making corrections, click **Restart** to publish the project again.

Publishing a document to an organization

Publishing a document first triggers the synchronization process, validates the document, and then publishes it. Only after a successful publish, the document is available at run time to be packaged as an application. If required, you can publish without synchronization and validation by [modifying the Workspace Properties](#).

You can publish a document in one of the following ways:

To publish a document using Workspace Documents:

- In the **Workspace Documents > <project> > <document>**, right-click the <document> and select **Publish to Organization**.
The publish process starts in a separate window, which also displays the publish status.

To publish a document using Quick Access Toolbar in an editor:

- In the toolbar, click **Publish**.
The publish process starts in a separate window, and shows the publish status.

To publish a document using Quick Access Menu on an editor, do the following:

1. On the top-left corner of the editor, click (Quick Access Menu) > Publish.
The link Click here to publish is displayed in the space to the right of the Quick Access Menu.
2. Click the hyperlink.
The publish process starts in a separate window, which shows the publish status.
A successful publish is indicated by , whereas errors are indicated by .
3. To know details of the published document, click **Details**.
The details appear in a Details tab.
4. After making corrections, click **Restart** to publish the document again.

Using the qualified name

Qualified Name (QN) is used to make the contents of an application unique at run time. When several applications from different vendors are installed on the same computer, it is possible that the names of some applications may have the same name and cause a conflict. To avoid any conflict in the names, AppWorks Platform provides you with the feature to set QN on the application contents so that the name of the published content is distinct.

You can set the QN start point only on folders. The folder on which you set the QN start point becomes the root of the published runtime content. As a result, only the contents within that folder are published in the destination directory, and not the folder itself.

You can also remove the QN start point if you do not require it.

To set the start point of a QN:

1. In the **Workspace Documents**, open the required <project>.
2. Right-click the required <folder> and select  (Set Start Point of Qualified Name).
The folder view changes from  to  indicating that the folder is marked as the start point of QN. The start point of QN is set on the selected folder. When you publish the entire project to runtime, the contents within this folder will appear according to the start point setting.

To remove the start point of a QN:

1. In the **Workspace Documents**, open the required <project>.
2. Right-click the <folder> on which you had set the start point and select  (Remove Start Point of Qualified Name).
The folder view changes from  to  indicating that the folder is not marked as the start point of QN. The Start Point of QN is removed from the selected folder.
You have used the QN feature to determine the way application content is published to run time.

For more information on which artifacts use QN, and their run-time references, see [Document storage and run-time reference information](#).

Document storage and run-time reference information

Document type	Identified by	Revision support	Allows duplicate names	Supports fallback	Responsible application connector	UI to view data
Access Control List	DN	No	No	No	part of platform core services	LDAP explorer
Action Template	QName	No	No	Yes	XML Store (Read)	XML Store Explorer
Application Connector	XML Store Key	No	No	Yes	XML Store	XML Store Explorer
Business Identifier	GUID	Yes	Yes	Yes	Business Process Management (Write), PIM (Read)	Process Instance Manager (PIM)
Business Process	QName	Yes	No	Yes	Business Process Management (Write), PIM (Read)	Deployed Process Models for Process-models, Process Instance Manager (PIM) for Process-

Document type	Identified by	Revision support	Allows duplicate names	Supports fallback	Responsible application connector	UI to view data
						instances
Translation	Source String & Language Code & Source Type	No	No	NA	Notification	Inbox in Runtime
Business Calendar	GUID or QName	No	No	Yes	Collaborative Workspace	NA
Business Calendar Exception	GUID or QName	No	No	NA	Collaborative Workspace	NA
Business event response	GUID	No	Yes	Yes	BAM Connector	NA
Business Measure	GUID	No	Yes	Yes	BAM Connector	Dashboard created from XForms
Case Model	QName	Yes	No	Yes	Business Process Management	Case Instance Manager
Composite Control	QName	No	No	Yes	Repository Service Container	XMLStore Explorer and CMC
Condition Template	QName	No	No	Yes	XML Store (Read)	XML Store Explorer
Configured Task	QName	No	No	Yes	Task Service (Inside Repository Service Container)	Repository Browser (via CMC)
Content Map	QName	No	Yes	Yes	CoBOC	CoBOC Browser
Run-time graphical models	GUID	No	No	No		Business Process designer from the PIM
Staging Content	GUID	No	No	No		Application Manager
XMLStore content	QName	No	No	Yes	XMLStore	XMLStore Explorer
Data Quality Plan	QName	No	Yes	Yes	NA	NA
Data Transformation	QName	No	Yes	Yes	Data Transformation	CoBOC Browser
Decision Table	GUID	No	Yes	Yes	Rule Management	Deployed Rules for Model Definitions, PIM for instances created through BPM

Document type	Identified by	Revision support	Allows duplicate names	Supports fallback	Responsible application connector	UI to view data
Dispatch Algorithm	GUID or QName	No	Yes	No	Notification	NA
Email Model	GUID	No	Yes	Yes	Notification	NA
File	Fully Qualified Name	No	No	No	NA	NA
Inbox Model	GUID	No	Yes	Yes	Notification	NA
Java Archive	Fully Qualified Name	No	No	No	All	NA
KPI	GUID	No	Yes	Yes	BAM Connector	Dashboard created from XForms
Language Pack	QName	No	No	NA	Collaborative Workspace	NA
MDM Model	QName and GUID	No	Yes	Yes	RCORMapper (com.cordys.mdm.connector) and PublisherMapper (com.cordys.mdm.publisher.connector)	MDM Data steward cockpit
Message Bundle	Fully Qualified Name	No	No	No	NA	NA
Multi-Language Mapper	QName	No	No	Yes	Repository Service Container	XMLStore Explorer and CMC
Operations Intelligence Model	NA	No	No	Yes	Operations Intelligence Service	NA
Organizational Unit	GUID or QName	No	No	Yes	Collaborative Workspace	User Manager
Process Monitoring Object	GUID	No	Yes	Yes	BAM Connector	Dashboard created from XForms with MOView composite control
Role	DN	No	No	No	part of platform core services	User manager/LDAP explorer
Rule	GUID	No	Yes	Yes	Rule Management	Deployed Rules for Model Definitions

Document type	Identified by	Revision support	Allows duplicate names	Supports fallback	Responsible application connector	UI to view data
Rule Group	GUID	No	Yes	Yes	Rule Management	Not Available
Schedule	GUID	No	Yes	Yes	Scheduler	Schedule Manager
Calculation Excel file	NA	No	No	Yes	Operations Intelligence Service	SharePoint
Visualization HTML page	NA	No	No	Yes	Operations Intelligence Service	SharePoint
Tag	GUID	No	No	Yes	Tag Service (In Repository Service Container)	Repository Browser (via CMC)
UITask	GUID	No	Yes	Yes	Task Service (Inside Repository Service Container)	Repository Browser (via CMC)
Web File Artifact	URI	No	No	Yes	NA	NA
Web Service Source XMLStore Content	Qname	No	No	No	Repository Service Container	XML Store explorer (both from Cordys Explorer and CMC)
Web service interface	LDAP dn	No	No	Yes	Whoever implements its type	Webservices Explorer
Delivery Model	GUID	No	Yes	Yes	Notification	NA
Work List	GUID or QName	No	Yes	Yes	Notification	NA
WS-AppServer properties	Qname	No	No	No	NA	text editor
WS-AppServer XMLStore Content (.cmx files)	Qname	No	No	Yes	Repository Service Container	XML Store editor (both from Cordys Explorer and CMC)
User Interface	QName	No	No	Yes	Repository Service Container	XMLStore Explorer and CMC

Reverting changes made in the workspace

At any time during development, you have the option to revert your local changes. Reverting your changes brings the workspace to the state of last synchronization with the repository (for example, the time of creating the workspace, last time of incorporate changes or last time of making changes available).

AppWorks Platform provides a single-click option to revert all the changes in the workspace.

Caution: Once you revert, you will lose all the changes that you made. If you need the modifications for later reference, create a copy of those documents that you were working on, before reverting.

Before you begin:

- The workspace should be configured to a Source Control Management (SCM) system, such as SVN.

To revert changes made in the workspace:

1. Switch to the workspace where you want to revert the changes.
2. On the toolbar of Workspace Documents, click (Revert All Workspace Changes).
 - a. If there are no changes to be reverted, a message appears informing you about this.
 - b. If there are changes to be reverted, the Revert Changes window appears, displaying all the changes that have occurred since the last time of synchronization with the repository.
 - Click **Revert**.
 A message appears informing you that changes have been reverted.
3. After reverting your local changes, all locks originating from the current workspace are released.

You have successfully reverted the changes made in the workspace.

Setting access control on XML store definition

While building an application, you may grant certain roles complete access or partial access to the XML Store content, or deny access to a particular XML Store content.

The Security feature helps you set access control on the XML documents within the XML Store content folder defined in the XML Store Definition. All the XML documents within the XML Store content folder have the same access control. You can set control at four levels: Read, Insert, Update, and Delete. The extent to which a user (bearing that role) is able to work with the XML Store content depends upon the level at which access control is set. For instance, only Read access allows the user to retrieve data from the XML Store, but does not allow the user to write data in it. Whereas, the Delete access allows the user to view, modify, insert, and delete the content.

Before you begin:

- Create an XML Store Definition.
- Create roles and publish them to the organization.

To set access control on XML store definitions:

1. In Workspace Documents (Explorer), open <solution> > <project>, right-click  (XML Store Definition), and select Define Runtime Security.
The Security Editor window opens displaying the name of the XML Store Definition on its

title bar.

2. In the Roles pane, click .
The Select Role dialog box opens displaying the roles that you created.
3. Select the role for which you want to set access control.
The selected role appears in the Roles pane.
4. In the **Permissions** pane, select **Read**, **Update**, **Insert**, or **Delete**.
5. Click .
The access control is set on the XML Store content and associated to a specific role.

After you complete this task:

To know how to view the access controls set for a particular role/user, see the topic *Viewing Access Permissions Granted for a Role* in the *AppWorks Platform Administrator's Guide*.

Sharing application content of a workspace with other developers

Developers can develop AppWorks Platform applications in project teams through a collaborative approach. Team members can carry out their development activities in isolated workspaces and share changes with other team members only when they are confident about the quality of their changes. All changes are stored in a central SCM repository to which all development workspaces are configured.

The collaborative application development environment of AppWorks Platform enables a project team to develop business solutions in a collaborative way. In this setup, developers might develop different types of documents, for example, Web service interfaces, XForms, business process models, or business rules. They may also simultaneously resolve identified development issues. In such a setup, it is very important for all the developers to continuously share updated content in the repository.

AppWorks Platform provides a single-click option to share changes with others by making those changes available in the SCM repository. All updates made from the workspace to the SCM repository are tracked instantly and versions are maintained. AppWorks Platform handles tracking automatically like any other configuration management tool. There is an option for users to provide comments while making updates. This makes tracking and retrieval of content easier at a later point.

Other members connected to the workspace can see the updates after they incorporate these changes in to their workspace. Updates are always done at the workspace level.

Before you begin:

- The workspace must be configured to a Source Control Management (SCM) repository, for example, SVN.

To share application content of a workspace with other developers:

1. Switch to the workspace from which you want to share changes with others.
2. On the toolbar of Workspace Documents, click  (Make Changes Available).
 - a. If there are no changes to share, a message appears indicating that there are no changes.
 - b. If there are changes to be made available, the Make Changes Available window opens.
 - i. If there are no changes to be incorporated, enter a comment describing the changes that you are committing, and then click **Make Available**. A message appears indicating completion of the activity.
 - ii. If there are changes to be incorporated, a second tab page lists the incoming changes.
Do one of the following:
 - To only incorporate the latest changes, click **Incorporate**.
 - To incorporate the latest changes and automatically continue with making your changes available, click **Incorporate & Make Available**.
In both scenarios, a message appears indicating completion of the activity.
3. After making your local changes available, all locks originating from the current workspace are released.

You have made your changes available to the SCM repository.

Switching between workspaces

You may need to switch between workspaces if you are developing applications in different workspaces or if you are a member of different teams that develop applications in separate workspaces.

You can switch workspace from the Workspace Documents window.

To switch between workspaces:

1. On the toolbar of the Workspace Documents window, next to  (Workspace), click . A list appears displaying all the workspaces available in that organization, with a tick mark against the current workspace.
2. Click the Workspace to which you want to switch.
The Workspace Documents window changes the view to the selected Workspace, and displays its contents.
You have just switched to a different workspace.
3. Alternatively, you can click  on the toolbar.
The Organize Workspaces window opens, listing all the available workspaces in that organization.
4. Select a Workspace and click **Open Workspace**.

Synchronizing workspace and file system

As you develop content in AppWorks Platform, it gets stored in a folder on the file system. By default, workspaces are synchronized to the following location on the server's file system: <AppWorks Platform_installdir>\<instance name>\cws\sync\<organization>\<workspace name>.

This is the default path. However, a System Administrator can modify it by changing the synchronize.folder variable in the cws.properties file located at <AppWorks Platform_installdir>\<instance name>\components\cws\config.

All the documents that you develop in workspace will be synchronized to this location when you run the Synchronize option. Similarly, if you develop some content directly or add some content at this location on the file system and want that to reflect in the workspace, run the Synchronize option.

Synchronization is important to keep the workspace content in sync with the content on the file system. It is to ensure that the most recent or the latest application sources are validated and published. That is why synchronization is a prerequisite for validate and publish activities. However, it becomes a hurdle especially when you want to validate or publish a document just to preview it, for example, previewing an XForm or running a Business Process Model interactively. Such dependency is likely to delay the development. To address this, an option is provided to turn off synchronization during validate and publish, whenever it is not needed. You can set this in Workspace Properties.

Caution: You may temporarily stop synchronization during validate and publish. However, remember to activate it before the final commit or packaging, as it is important to keep the workspace and the file system always in sync.

To synchronize workspace and file system:

1. Switch to the workspace that you want to synchronize.
2. On the toolbar of Workspace Documents, click  (Synchronize).

The synchronization process starts and the status is displayed in a Synchronize window.

If you synchronize the workspace in Classic View, the synchronization process status is displayed in a pane to the right of Workspace Documents.

A successful synchronization is indicated by  whereas errors are indicated by .

3. To know the synchronization details, click **Details**.
The details appear in a Details tab.
4. After making corrections or [resolving synchronization conflicts](#), click **Restart** to synchronize the workspace again.

To synchronize at the folder or project level:

1. To synchronize document(s) within a folder, right-click the folder and select  (Synchronize).

- To synchronize folders within a project, use this option at the project level.

Using this option, you can choose to synchronize contents of a specific folder or folders and documents within a project, without synchronizing the entire Workspace.

For example, if you modified a Java, JavaScript, or a HTML document on the file system and want to synchronize the modifications of only that document, you can synchronize the folder containing that document. In this manner, you could save lot of time during application development. However, this type of synchronization applies only to modified or updated documents. The entire Workspace is synchronized automatically to avoid conflicts if you add or delete a document in a folder on the file system, or move a document from one folder to another on the file system.

The following table explains the various scenarios when a partial synchronization (of the folder alone) or a Workspace synchronization happens.

Partial synchronization of the parent folder	Partial synchronization of the parent folder followed by complete synchronization of the workspace
When you modify a document within the folder on the Workspace (browser)	When you add a document to the folder or its child folders on the file system
When you modify a document within the folder on the file system	When you delete a document from the folder or its child folders on the file system
When you move a document from one child folder to another child folder on the Workspace	When you move a document from one child folder to another child folder on the file system
When you move a document from one child folder to another child folder on the file system	When you move a document from one parent folder or its child folder to another parent folder or its child folder on the Workspace or the file system
When you delete a document from the folder on the Workspace	When you rename the project or folder containing the parent folder on the Workspace
When you add a document to the folder on the Workspace	-

Resolving synchronization conflicts overwrites the workspace content with the content in the file system

The synchronization function ensures that the shared content in the workspace repository and the file system is consistent in all respects. As the framework offers collaborative development environment, users are likely to make changes to the content either in the workspace or in the file system or at both places. Using the synchronizer, you can keep the content synchronized to the maximum extent possible. However, at times it fails because of simultaneous changes to the same content at both locations. When this happens,

synchronization fails indicating a conflict. This is represented by the Edit Conflicts button that appears on the Synchronization window.



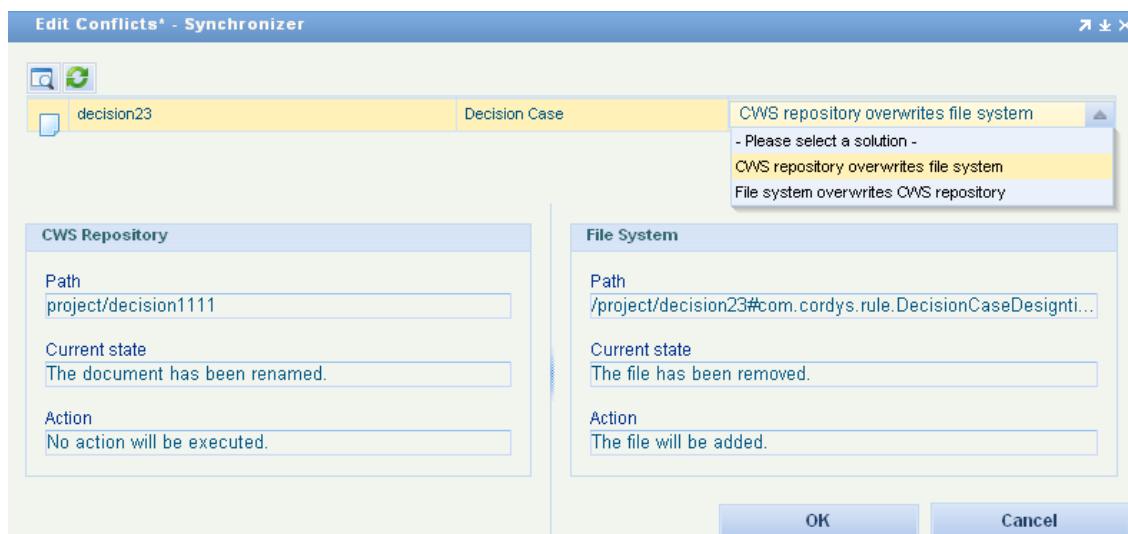
As AppWorks Platform is not equipped with automatic conflict resolution, conflicts need to be resolved manually. Using the Edit Conflicts option, users can resolve conflicts and synchronize contents. Conflicts can be resolved in one of the following ways:

- Overwrite the content in the file system
- Overwrite the content in the workspace

The interface displays the status for both workspace as well as the file system and lets you choose which one to overwrite.

To resolve synchronization conflicts:

1. On the Synchronizer window, click **Edit Conflicts**.
The Edit Conflicts - Synchronizer dialog box opens.
2. Analyze the conflict details and select one of the options in the list.



CWS repository overwrites file system

Overwrites the content in file system with the content in workspace

File system overwrites CWS repository	Overwrites the workspace content with the content in the file system
---------------------------------------	--

The remaining fields are as follows.

Path	Location of the document both in CWS and file system.
Current state	State of the document when the conflict occurred.
Action	Result (how the conflict will be resolved) based on the selected option from the list. The information in the Action field changes when you select an option in list.

3. Click **OK**.

The synchronizer restarts and indicates successful synchronization.

Synchronization conflicts are successfully resolved and the file system or workspace is updated as required.

Updating the workspace with content modified by other developers

The collaborative application development environment of AppWorks Platform permits more than one developer to work on a single workspace at the same time, and develop content therein. However, from a team development perspective, it is recommended that the users create different workspaces that point to the same SCM.

In team development, developers may develop different types of documents, for example, Web service interfaces, XForms, business process models, or business rules or simultaneously resolve identified development issues. In a setup like this, it is very important for all the developers to regularly receive updates done to the workspace contents by other members and keep their workspace up-to-date.

Before you begin:

- The workspace that you wish to update should be configured to a source control management system, for example, SVN.

This option helps you to get those updates from the SCM system. Updates are always received and applied at the workspace level.

To update the workspace with content modified by other developers:

1. [Switch to the workspace](#) where you want to receive updates from the SCM system.
2. On the toolbar of Workspace Documents, click  (Incorporate Changes).
 - If there are no changes to be incorporated, a message appears informing you about this

- If there are changes to be incorporated, the Accept Repository Changes window opens. It displays the list of changes made to the repository by other developers. In the Accept Repository Changes window, you can do the following:
 - Click **Incorporate** to accept the changes and get them into your workspace. A message appears informing successful incorporation of changes.
 - Click **Cancel** to close the window without accepting the changes.
 - Click **Show History** to open the view that shows all changes to the SCM repository.

See also [Viewing the Change History of a Workspace and its Contents](#).

The updates from the SCM system are incorporated into the workspace.

Sometimes, when the CWS repository and the file system are not in sync, incorporating changes may result in synchronization conflicts. In such situations, you must [Edit Conflicts](#) before you can continue incorporating the changes.

Uploading a document to the project

AppWorks Platform provides an option to bundle a document (not of AppWorks Platform format) with the project. You will be able to use this document during runtime. Several document formats including image files can be uploaded to the project. You can upload a document to a project or to a folder within a project from your computer.

To upload a document to the project:

1. In the Workspace Documents, right-click the required <project> or <folder> and select  (Upload Document).
The Upload Document Wizard opens.
2. In the **File** field, browse and select the file to upload.
The selected file name along with its extension and the path from where it is being uploaded appears in the field. Simultaneously, the Name and Description fields display the file name and the Extension field displays the file extension.
3. Click **Finish**.
The Upload Document Wizard closes.

The uploaded document is added to the project or the folder.

If you upload an HTM file to the project and want to package it along with the application content, create a [Web Library Definition](#) document to contain the HTM file. Otherwise, the HTM file is not available in the application package.

Using localization bundles in an application

Localization bundles, also known as message bundles are XML files containing application-specific messages that are used by the application at run time to notify the user of certain system activity, errors, or warnings. The localization bundles that are used at run time by

AppWorks Platform or applications running on it are stored in the folder <AppWorks Platform_installdir>\<instance name>\localization. The localization bundles are stored in this directory at the time of installing an application package. Therefore, if you have localization bundles that must be used by your application, you must publish and package them along with the rest of your application content. When you install the application package, the message bundle is extracted and stored at the above-mentioned location.

To use localization bundles in an application:

1. [Switch to the workspace](#) where you developed the application content, and [synchronize](#) it with the file system.
The workspace and the file system are synchronized.
2. On the file system, locate the project folder that contains your application code and create a plain XML document in any folder or sub-folder with the following naming convention and content:
 - The name of the file must be in the following format:
<id of the localization bundle>#MessageBundle#.xml.
For the following example, this must be
PurchaseOrderMessages#MessageBundle#.xml
 - The content of the file must be in the following format.

```
<MessageBundle id="PurchasingOrderMessages" xmlns=""  
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">  
    <Message id="OrderIdAlreadyExists">  
        <MessageText>An order with the given id already exists</MessageText>  
    </Message>  
    <Message id="OrderAcceptedForProduction">  
        <MessageText>The order has been accepted for production.</MessageText>  
    </Message>  
</MessageBundle>
```

For the application to display the related message, you must include a reference to the Message ID in the application logic.

3. [Synchronize the workspace.](#)
The XML document is added to the project or its folder.
4. [Publish the project.](#)
5. [Create the application package.](#)
6. Deploying the application on the server where you want to run it (see the AppWorks Platform Administration Guide).

After successful installation, the XML file containing the localization bundle is stored in the <AppWorks Platform_installdir>\<instance name>\localization folder. The messages in it are used when the application is run on that server.

Viewing and modifying properties of a workspace

The workspace properties contain additional information such as the name and description, the organization to which the content will be published, and the number of recently created or viewed documents to display. You can view or modify the workspace properties from Workspace Documents.

To view and modify properties of a workspace:

1. On the Workspace Documents toolbar, click  (Workspace Properties).
The Workspace window opens, displaying information across four tabs.

Tab	Field	Description	Edit rights
Workspace	Name	Name of the workspace	Read-only
	Description	Description of the workspace	Editable
Annotations		Notes for the workspace	Editable
User Properties	Publish Organization	Organization to which the contents will be published. To effectively publish documents and their dependencies to the new target organization, you must first clean the build output of the workspace.	Editable (Selection)
	Automatic Synchronize during Validate and Publish	When selected, enables automatic synchronization of the workspace whenever a validate or publish command is executed. By default, it is selected. When the option is cleared, validate and publish occurs without synchronization, which results in a faster validation or publish. You may clear the check box temporarily to enable faster validation or publish so that you can test the workspace content that you are developing. However, for the final commit or packaging, it is important to keep the check box selected. Otherwise, the workspace and the file system will not be in sync and can consequently cause issues	Editable (select, clear)

Tab	Field	Description	Edit rights
History	Number of Documents shown in My Recent Documents	Provision to set the number of recently used items that will be displayed at a time in the Workspace Documents (My Recent Documents) view.	Editable
	Created by; Creation date; Modified by; Modified date	History of creation and modification	Read-only

You can view the details and modify them as required.

2. Modify and click **OK** to save the changes.
The Workspace window closes.

For Application Management spaces that are accessed through the Application Manager task on the Welcome page, the Workspace tab will not be displayed because the name and description of the workspace are not relevant to end users.

Viewing and modifying properties of a solution

This feature enables users to logically group their projects that exist in the same workspace, if required.

The solution properties contain information such as the name and description and its maintenance history.

To view and modify solution properties:

1. In the **Workspace Documents**, right-click <**solution**> and select **Properties**.
The Solution window opens, displaying information across three tabs.

Tab	Field/Button	Description	Edit Rights
General	Name	Name of the Solution	Editable
	Description	Solution description	Editable
	Direct Access URL (REST)	A link to access the document directly	Read-only
Annotations		Notes for the solution	Editable
History	Creation	User information and date and time when the solution was created	Read-only
	Last Modification	User information and date and time when the solution was recently modified	Read-only

2. View the details and/or modify the details as required, and click **OK** to save the changes. The Solution window closes.

Viewing and modifying properties of a project

The project properties contain additional information such as the name and description and its maintenance history.

To view and modify project properties:

1. In the Workspace Documents, open <solution>, right-click <project> and select Properties. The Project window appears, displaying information across three tabs.

Tab	Field/Button	Description	Edit Rights
General	Name	Name of the Project	Editable
	Description	Project description	Editable
	Direct Access URL (REST)	A link to access the document directly	Read-only
Annotations		Notes for the project	Editable
History	Creation	User information and date and time when the project was created	Read-only
	Last Modification	User information and date and time when the project was recently modified	Read-only

2. View the details and/or modify the details as required, and click **OK** to save the changes. The Project window closes.

Viewing the change history of a workspace and its contents

When developers connect to an SCM system and frequently modify its contents, they might want to trace and view all the changes made to a project, a folder, or a document. It would help developers monitor the development of the application and retrieve useful information from the past, if necessary.

The Change History feature of CWS provides the facility to view all the changes made to the workspace and its contents. It also displays, the comments that were provided during each "Make Available" action, and the type of action (add, update, and delete) performed on a document. You can also view change history for a specific project, folder, or even a document.

Before you begin:

- The workspace should be configured to a Source Control Management (SCM) system, for example, the SVN.

To view the change history of a workspace:

1. On the toolbar of Workspace Documents, click  (Show Change History). The Change History dialog box opens, displaying the Current Workspace Revision number in a text box and all the earlier revision numbers in a tabular format.
2. Click a revision entry in the table. The comment (if available) appears in the comment box and other details, such as the change action (add, update, or delete), name of the document, and the location on the SCM system appear in another table on the same window.
3. Initially, the Change History dialog box displays only 25 of the most recent changes. To display the next 25 changes, click **Get next 25 changes**.
4. To view all the changes, click **Get all changes**.

To view change history of a project, folder, or a document:

1. In the **Workspace Documents** (Explorer), open <**solution**> (if applicable), right-click <**project**>, <**folder**>, or <**document**> and select **Change History**. The Change History dialog box opens, displaying the Current Workspace Revision number in a text box and all the earlier revision numbers specific to that particular project, folder, or document in a tabular format. The revision number in the table indicates the latest change for that project, folder, or document whereas the revision number in the text box is related to the entire workspace
 2. Click a revision entry in the table. The comment (if available) appears in the comment box along with other details, such as the change action (add, update, delete), name of the document, and the location on the SCM system appear in another table on the same window.
- When you try to view Change History of a document or folder that is not yet available in the SCM system, a notification with the following message is displayed: There is no history available for this document, it has not yet been committed to the repository.

You have successfully viewed the change history for a workspace, project, folder, or a document.

Reloading runtime references

Important: This command is deprecated. Use the `cws reloadruntimereferences` command. For more information, see [reloadruntimereferences](#).

The reuse of application artifacts is supported through runtime references. See Runtime references for more information on which application artifacts can be reused.

Information stored in a runtime reference can become outdated when compared to the deployed version of the referenced artifact due to various reasons. One of the reasons may be that the contract of the referenced artifact has changed since the time of creating or last reloading the runtime reference.

Publishing a project or a single document, which includes a runtime reference that is outdated will fail. It must be reloaded to make the runtime reference up-to-date again.

To reload a runtime reference:

1. In Workspace Documents, expand the project hierarchy up to the folder in which the runtime reference is located.
2. Right-click the <runtime reference> and select **Reload Reference**.

The command-line tool with which all the runtime references in a workspace can be reloaded with a single action is as follows:

```
ReloadRuntimeReferences <organization name> <workspace name> [-execute]
```

The command-line tool accepts the following parameters where:

- <organization name> is the name of the organization
- <workspace name> is the name of the workspace
- -execute is an optional parameter.

This option causes all the outdated runtime references to be reloaded. Without this option, a test run will be performed.

To reload all the runtime references using the command-line tool:

1. Log in to the AppWorks Platform server.
2. Open a command prompt and navigate to the folder <AppWorks Platform_installdir>/components/cws/scripts. In a Windows environment, you must navigate to the subfolder windows; for a Linux environment, you must navigate to the subfolder linux.
3. Run the tool with the proper parameters.

The user who executes the tool must have a AppWorks Platform user account with the user ID set as the user name of the operating system user account. The AppWorks Platform user must be assigned the Developer role in the organization in which the workspace resides.

Examples on how to run the command-line tool, including sample output is as follows.

Example 1: Command to trigger a test run of reloading runtime references in the workspace "My Workspace" in the organization "My Organization"

```
~>ReloadRuntimeReference "My Organization" "My Workspace"  
Starting ReloadRuntimeReferences
```

This is a test run. Runtime references will not be reloaded.

The runtime reference MyProject/RuntimeRoles/Tester has been reloaded.

Verifying runtime references, 100% completed.

This was a test run. No runtime references have been modified. To reload all the runtime references, use the option '-execute'.

All runtime references have been processed.

Example 2: Command to reload all the outdated runtime references in the workspace "My Workspace" in the organization "My Organization"

```
~>ReloadRuntimeReferences "My Organization" "My Workspace" -execute
Starting ReloadRuntimeReferences

The runtime reference MyProject/RuntimeRoles/Tester has been reloaded

Verifying runtime references, 100% completed.

All runtime references have been processed.
```

Using the CWS command-line tools

You can use command-line tools for the actions that can be performed in CWS, such as building, publishing, packaging, and synchronizing.

These tools are available in the form of:

- Generic CWS command with sub commands
- Specific CWS commands

Before executing any of the CWS command-line tools, ensure the following:

- If the operating system is Windows, an environment variable CORDYS_HOME must exist and its value must be set to the installation directory of the AppWorks Platform environment.
- Ensure that the user executing the tool is a user in AppWorks Platform. There must be a user with the User ID set as the User name of the operating system user account.
- The user executing the tool is authorized to do so in the context of a specific organization. The user must be assigned the Developer role in that organization.

Generic CWS command with subcommands

To execute a subcommand with the generic command:

1. Log in to the AppWorks Platform server.
2. Open a command prompt and navigate to the <AppWorks Platform_installdir>/bin/ folder.
3. In Windows, run the command prompt as Administrator.
4. Run the command `cws` with the relevant subcommand and arguments as follows.

Subcommands

- [PruneObsoleteAdministration](#)
- [CreateSVNWorkspace](#)
- [RemovePSL](#)
- [validate](#)
- [publish](#)
- [package](#)
- [synchronize](#)
- [resetsynchronizer](#)
- [removeworkspace](#)
- [movesynchronizelocation](#)
- [reloadruntimerefrences](#)
- [upgradesvnworkingcopy](#)

Help on subcommands

You can use the following to get help related to the subcommands:

Operating system	All subcommands	Specific subcommand
Windows	<code>cws help</code>	<code>cws help <subcommand></code>
Linux	<code>./cws.sh help</code>	<code>./cws.sh help <subcommand></code>

Arguments

Each subcommand has its own arguments. Optionally, you can add one or more of these tuning arguments.

Setting	Setting	Default value	Example	Remark
Heap	<code>-MaxHeap xxx</code>	Default JVM	<code>-MaxHeap 2G</code>	The value appended with

Setting	Setting	Default value	Example	Remark
		heap size		a unit, indicated with a letter. Append the letter 'k' or 'K' to indicate kilobytes, 'm' or 'M' to indicate megabytes, and 'g' or 'G' to indicate gigabytes. You must enter one of the above units. There is no default unit.
NOM	-MaxNOM xxx	Default NOM size	-MaxNOM 4G	The size in megabytes. Append the letter 'm' or 'M' to indicate megabytes, and 'g' or 'G' to indicate gigabytes. The default unit is M.
Classpath	-cp pathToJar	empty	-cp ./extraCommand.jar	
Language	-lang xx_XX	empty	-lang en_US	If empty, the settings from wcp.properties are used

Specific CWS commands

To execute a specific CWS command:

1. Log in to the AppWorks Platform server.
2. Open a command prompt and navigate to the <AppWorks Platform_installdir>/components/cws/scripts folder. In Windows, run the command prompt as an administrator
 - In a Linux environment, navigate to the subfolder linux.
 - In a Windows environment, navigate to the subfolder windows.
3. Run the command with correct arguments.

Commands

For more information on a command and its arguments, see these topics:

- [CWSBuild](#)
- [CWSDeploy](#)

- [CWSOverview](#)
- [CWSPackage](#)
- [CWSSynchronizer](#)
- [CWSWorkspaceRemovalTool](#)
- [MoveSynchronizerLocation](#)
- [ReloadRuntimeReferences](#)
- [ResetSynchronizer](#)
- [UpgradeSVNWorkingCopy](#)

CWS sub command: CreateSVNWorkspace

The CreateSVNWorkspace sub command creates a workspace in CWS based on an SVN repository. The command can be used as:

```
cws CreateSVNWorkspace <parameterlist>
```

Parameters

The CreateSVNWorkspace sub command creates a workspace in SVN with the following parameters.

<code>-description <workspace description></code>	: Description of the workspace.
<code>-organization <organization name></code>	: Name of the organization.
<code>-proxyhost <proxy host></code>	: Host name of the proxy server.
<code>-proxypassword <proxy password></code>	: Password to access the proxy server.
<code>-proxyport <proxy port></code>	: Port number of the proxy server.
<code>-proxyusername <proxy username></code>	: User name to access the proxy server.
<code>-svnpassword <svn username></code>	: Password to access the SVN repository.
<code>-svnurl <svn url></code>	: Location of the SVN repository.
<code>-svnusername <svn username></code>	: User name to access the SVN repository.
<code>-workspace <workspace name></code>	: Name of the workspace.

Example

To create a MyWorkspace workspace in the MyOrganization organization with the SVN repository details, perform the following:

```
cws CreateSVNWorkspace -organization MyOrganization -workspace MyWorkspace -  
description MyWorkspaceDescription -svnurl http://<url>/repository/MyProject/trunk -  
svnusername <user> -svnpassword <pass>
```

If a parameter contains white space characters, enclose it in double quotes as shown:

```
cws CreateSVNWorkspace -organization "My Organization" -workspace "My Workspace" -
```

```
description "My Workspace Description" -svnurl "http://<url>/repository/My
Project/trunk" -svnusername <user> -svnpassword <pass>
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: PruneObsoleteAdministration

The obsolete administration data consists of:

- CWS database objects
- CWS help files on the file system

The prune mechanism consists of the following background tasks running within the CWS service container:

- Object Pruner - Deletes all the database records that are marked for deletion. By default, it is scheduled to run daily at 5:00, 12:00, and 20:00 hours.
- File Pruner - Deletes all the log and help files that are created by CWS and older than 1 to 3 days, depending on the file type. By default, it is scheduled to run daily after 3 minutes of the start of the CWS container.

Note

- You can change the time schedule of the Object Pruner, but the time schedule of the File Pruner cannot be changed.
- See [Defining a custom schedule for object pruning](#) for definition of a time schedule for the Object Pruner task.

When a considerable number of resources are used from the CWS service container and developer performance is impacted, run the Object Pruner task in a separate JVM from the command line using the following command:

```
cws PruneObsoleteAdministration
```

Parameters

This sub command does not accept additional parameters. However, it may require memory configuration using the `-MaxHeap` and `-MaxNom` parameters.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: RemovePSL

A PSL is used to graphically view business process models after deployment and enables process administrators to graphically monitor and debug the execution of instances of these business process models. To align with the versioning capabilities on business process models, every package contains a unique PSL and does not overwrite the PSLs of others versions of the same package or other packages.

Caution: Running the RemovePSL tool removes the inactive Publish Source Layers (PSLs) from the system and consequently improves the migration performance. The process of identifying PSLs eligible for removal involves querying various database tables that might contain several records, for example, PROCESS_INSTANCE table. Running the tool during business hours severely impacts performance. Therefore, ensure this tool is run only during non-business hours.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#). For performance recommendations, see the Performance Configuration Guidelines in the *AppWorks Platform Performance and Tuning Guide*. Also, ensure the following:

- The appropriate database driver must be added to the classpath of the Collaborative Workspace Service Container, for example, `sqljdbc.jar` for MSSQL Server.
- The user executing the tool must have read and write permissions on the `<AppWorks Platform_installdir>`.

You can use the RemovePSL sub command as follows:

```
cws RemovePSL <parameterlist>
```

Running the command without parameters displays the inactive PSLs.

Parameters

The following parameters define the behavior of the RemovePSL sub command.

<code>-execute (-x)</code>	: Removes the inactive PSLs from the environment.
<code>-history (-h) <number of days></code>	: Indicates the period a PSL must be inactive before it is eligible for removal.
<code>-verbose (-v)</code>	: Enables verbose logging and provides more information on the timings.

For example, to remove PSLs that are inactive for 10 days or more, use the following command in a Windows environment.

```
cws RemovePSL -execute -history 10
```

A PSL is active with respect to a Business Process Model if one of the following conditions are true.

The deployed business process model referring to the PSL is:

- still active
- inactive but active process instances exist for the given business process model revision
- inactive and only completed or terminated process instances exist for the given business process model revision (older than what is shown in the history window)

A PSL is active with respect to a Decision Table if one of the following conditions is true.

The deployed decision table referring to the PSL is:

- still active
- inactive but active process instances exist in the decision table
- inactive and only completed or terminated process instances exist in the decision table (older than what is shown in the history window)

In all other cases, the PSLs are considered to be inactive and qualified for deletion.

When the history window displays and the PSLs are deleted, you cannot open the graphical view of the process instances available outside the history window. However, you can still work with the text or grid view in the Process Instance Manager (PIM).

The deployed MDM model referring to the PSL is when the Model is published to run time.

CWS command-line tool: CWSBuild

Deprecated: This command is deprecated. Use the `cws validate` command. For more information, see [validate](#).

The command-line tool CWSBuild can be used to build a Collaborative Workspace (CWS) document that resides in a development workspace. Running this tool for a single document, folder, or project has the same result as validating the same document from the browser using the context menu item Validate.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSBuild accepts three parameters:

```
CWSBuild <ORGANIZATION_NAME> <WORKSPACE_NAME> <DOCUMENT_NAME>
```

Where,

- <ORGANIZATION_NAME> is the name of the organization in which the document resides
- <WORKSPACE_NAME> is the name of the workspace in which the document resides
- <DOCUMENT_NAME> is the name of the document in terms of its path in the workspace

For example, if your organization is called 'MyOrganization', you have a workspace called 'MyWorkspace', and that workspace contains a document with the path 'MyApplication/Roles/MyRole', building the document can be done using the following command:

```
CWSBuild MyOrganization MyWorkspace MyApplication/Roles/MyRole
```

In cases where any of the parameters contain white space characters, you must include them in double quotes as shown:

```
CWSBuild "My Organization" "My Workspace" "My Application/Roles/My Role"
```

CWS command-line tool: CWSDeploy

Deprecated: This command is deprecated. Use the `cws publish` command. For more information, see [publish](#).

The command-line tool CWSDeploy can be used to deploy documents to the corresponding run-time repository.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSDeploy accepts three parameters:

```
CWSDeploy <ORGANIZATION_NAME> <WORKSPACE_NAME> <DOCUMENT_NAME>
```

Where,

- <ORGANIZATION_NAME> is the name of the organization in which the document resides
- <WORKSPACE_NAME> is the name of the workspace in which the document resides
- <DOCUMENT_NAME> is the name of the document in terms of its path in the workspace

For example, if your organization is called 'MyOrganization', your workspace is called 'MyWorkspace', and that workspace contains a document with the path 'MyApplication/Roles/MyRole', deploying that document to the corresponding run-time repository can be done using the following command:

```
CWSDeploy MyOrganization MyWorkspace MyApplication/Roles/MyRole
```

In cases where any of the parameters contain white space characters, you must include them in double quotes as shown:

```
CWSDeploy "My Organization" "My Workspace" "My Application/Role/My Role"
```

CWS command-line tool: CWSOverview

The command-line tool CWSOverview can be used to generate an overview of all workspaces that exist in the AppWorks Platform environment.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSOverview accepts one parameter:

```
CWSOverview [<ORGANIZATION>]
```

Where,

- <ORGANIZATION> is the name of the organization for which the workspaces must be listed. This parameter is optional.

If no parameter is provided, the command-line tool will generate an overview of all workspaces, grouped by organization.

For example, if your organization is called 'MyOrganization', generating an overview of all workspaces in that organization can be done using the following command:

```
CWSOverview MyOrganization
```

In cases where the name of the organization contains white space characters, you must include it in double quotes as shown:

```
CWSOverview "My Organization"
```

The output of the tool may be as shown:

```
~>CWSOverview
Starting CWSOverview

** Organization 'o=development,cn=cordys,cn=defaultInst,o=acme.com' ***
DevelopmentWorkspace: ACME
Created on: Thu Dec 17 11:33:05 CET 2009 [0050568B-184D-71DE-8FEB-EAF759044E98]
DevelopmentWorkspace: Test
Created on: Thu Dec 17 11:32:06 CET 2009 [0050568B-184D-71DE-8FEB-EAF759044EA4]

** Organization 'o=system,cn=cordys,cn=defaultInst,o=acme.com' ***
DevelopmentWorkspace: ACME Test Workspace
Created on: Thu Dec 17 11:34:12 CET 2009 [0050568B-184D-71DE-8FEB-EAF7A49C859C]
ISVPStagingSpace: __ISVP_Space__
Created on: Thu Dec 17 11:22:49 CET 2009 [0050568B-184D-71DE-8FEB-EAF62005C491]
Metaspacespace: __CWS_Default_Metaspacespace__
```

```
Created on: Thu Dec 17 06:12:50 CET 2009 [0050568B-184D-71DE-A89E-EACACE79EA23]
OrganizationStagingSpace: __Organization Staging__
Created on: Thu Dec 17 11:22:49 CET 2009 [0050568B-184D-71DE-8FEB-EAF62005C4BC]
```

CWS command-line tool: CWSPackage

Deprecated: This command is deprecated. Use the `cws package` command. For more information, see [package](#).

The command-line tool CWSPackage can be used to create application packages from Collaborative Workspace (CWS) projects.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSPackage accepts three parameters:

```
CWSPackage <ORGANIZATION_NAME> <WORKSPACE_NAME> <PROJECT_NAME>
```

Where,

- <ORGANIZATION_NAME> is the name of the organization in which the project resides
- <WORKSPACE_NAME> is the name of the workspace in which the project resides
- <PROJECT_NAME> is the name of the project to be packaged

For example, if your organization is called 'MyOrganization', your workspace is called 'MyWorkspace', and that workspace contains a project named 'MyProject', packaging that project can be done using the following command:

```
CWSPackage MyOrganization MyWorkspace MyProject
```

White space characters in parameters

In cases where any of the parameters contain white space characters, you must include these in double quotes as shown:

```
CWSPackage "My Organization" "My Workspace" "My Project"
```

The created package is stored at the following location: <AppWorks_Platform_installdir>/cws/build/<ORGANIZATION_NAME>/<WORKSPACE_NAME>/package/

CWS command-line tool: CWSSynchronizer

Deprecated: This command is deprecated. Use the `cws synchronize` command. For more information, see [synchronize](#).

The command-line tool CWSSynchronizer can be used to synchronize a particular workspace with the corresponding synchronization directory on the file system of the server. Running this tool has the same result as synchronizing the same workspace from the browser.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSSynchronizer accepts two parameters:

```
CWSSynchronizer <ORGANIZATION_NAME> <WORKSPACE_NAME>
```

Where,

- <ORGANIZATION_NAME> is the name of the organization in which the workspace resides
- <WORKSPACE_NAME> is the name of the workspace to be synchronized

For example, if your organization is called 'MyOrganization', and you have a workspace called 'MyWorkspace', synchronizing that workspace can be done using the following command:

```
CWSSynchronizer MyOrganization MyWorkspace
CWSPackage "My Organization" "My Workspace" "My Project"
```

In cases where any of the parameters contain white space characters, you must include them in double quotes as shown:

```
CWSSynchronizer "My Organization" "My Workspace"
```

CWS command-line tool: CWSWorkspaceRemovalTool

Deprecated: This command is deprecated. Use the `cws removeworkspaces` command. For more information, see [removeworkspaces](#).

The command-line tool CWSWorkspaceRemovalTool can be used to remove specific types of workspaces from a specified organization.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool CWSWorkspaceRemovalTool accepts the following parameters:

```
CWSWorkspaceRemovalTool [ORGANIZATION_NAME|-all-organizations] [-psl] [-staging] [-care] [-development]
```

Where,

- <ORGANIZATION_NAME> is the name of the organization in which the workspaces will be removed. To remove the specific types of workspaces from all organizations, you must provide the option -all-organizations instead of the name of one specific organization.
- -psl is an optional parameter. When this parameter is provided, all PSLs will be removed in the specified organization.
- -staging is an optional parameter. When this parameter is provided, the application management space, also known as staging space, will be removed in the specified organization.
- -care is an optional parameter. When this parameter is provided, all CARE spaces will be removed in the specified organization.
- -development is an optional parameter. When this parameter is provided, all the development workspaces will be removed in the specified organization.

You can provide any combination of the last four parameters. When none of these parameters are provided, nothing will be removed.

For example, to remove all organization-level PSLs from an organization called 'MyOrganization', you must use the following command:

```
CWSWorkspaceRemovalTool "MyOrganization" -psl  
CWSSynchronizer "My Organization" "My Workspace"
```

Removing all organization-level PSLs as well as all development workspaces from all organizations can be done using the following command:

```
CWSWorkspaceRemovalTool -all-organizations -psl -development
```

If the name of the organization contains white space characters, you must include it in double quotes as shown:

```
CWSWorkspaceRemovalTool "My Organization" -psl -care -staging -development
```

CWS command-line tool: MoveSynchronizerLocation

Deprecated: This command is deprecated. Use the `cws movesynchronizelocation` command. For more information, see [movesynchronizelocation](#).

The command line tool MoveSynchronizerLocation is used to change the location of the Collaborative Workspace (CWS) synchronization folder in an automated way. This affects the synchronization location of all the workspaces in the AppWorks Platform instance. When running the tool, first all CWS service containers are stopped to prevent changing the synchronization folder while someone is still developing content in one of the workspaces. There may be multiple CWS service containers in case of load balancing. After running the tool, all the CWS service containers must be restarted manually.

The MoveSynchronizerLocation tool does not change the location of the build folder; if required, you must change it manually.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool MoveSynchronizerLocation accepts two parameters:

```
MoveSynchronizerLocation DESTINATION [-skipConfiguration]
```

Where,

- DESTINATION is the new synchronization directory
- -skipConfiguration is an optional parameter.
When this parameter is provided, the CWS configuration file, located at <install directory>/components/cws/config/cws.properties, is not updated.

If you run the MoveSynchronizerLocation tool with the optional parameter – skipConfiguration, the mentioned configuration file will not be updated. In that case, the old synchronization directory will be used again. To make the changes effective in this case, the configuration file must be updated manually.

For example, changing the synchronization directory to E:\temp\cws\sync can be done by running the following command:

```
MoveSynchronizerLocation E:\temp\cws\sync
```

In case the path of the destination directory contains white space characters, the path must be included in double quotes as shown:

```
MoveSynchronizerLocation "E:\temp\cws\sync folder\"
```

The operating system user account under which the platform operates must have full permissions for the new synchronization directory.

CWS command-line tool: ResetSynchronizer

Deprecated: This command is deprecated. Use the cws resetsynchronizer command. For more information, see [resetsynchronizer](#).

The command-line tool ResetSynchronizer can be used to solve the synchronization problems in case of unrecoverable errors. It can be used for single workspaces only. When running the tool, the current synchronization folder of the respective workspace will be renamed. In the next synchronization, the workspace content will be stored in a freshly created synchronization folder as it is in the CWS repository.

If the workspace previously had SVN-integration, that integration will be lost after running the ResetSynchronizer tool.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool ResetSynchronizer accepts two parameters:

```
ResetSynchronizer <ORGANIZATION NAME> <WORKSPACE NAME>
```

Where,

- <ORGANIZATION NAME> is the name of the organization in which the workspace resides
- <WORKSPACE NAME> is the name of the workspace

For example, if your organization is called 'MyOrganization', and you have a workspace called 'MyWorkspace', resetting the synchronizer for this workspace can be done using the following command:

```
ResetSynchronizer MyOrganization MyWorkspace  
MoveSynchronizerLocation "E:\temp\cws\sync folder\"
```

In cases where the name of the organization or the name of the workspace contain white space characters, you must include them in double quotes as shown:

```
ResetSynchronizer "My Organization" "My Workspace"
```

CWS command-line tool: UpgradeSVNWorkingCopy

Deprecated: This command is deprecated. Use the `cws upgradesvnworkingcopy` command. For more information, see [upgradesvnworkingcopy](#).

The command-line tool UpgradeSVNWorkingCopy is used to upgrade the existing team development workspaces to the configured version of Subversion.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

The command-line tool UpgradeSVNWorkingCopy is used in one of the following ways:

- By upgrading the working copy of a specific workspace
- By upgrading the working copy of all the workspaces in a specific organization
- By upgrading the working copies of all the workspaces in all the organizations

To upgrade the working copy of a specific workspace, run the tool with the following parameters:

```
UpgradeSVNWorkingCopy <organization name> <workspace name>
```

Where,

- <organization name> is the name of the organization.

To upgrade the working copy of all workspaces, run the tool with the following parameters:

```
UpgradeSVNWorkingCopy <organization name>
```

Where,

- <organization name> is the name of the organization and <workspace name> is the name of the workspace of which the working copy must be upgraded.

To upgrade the working copies of all the workspaces in all organizations, run the tool with the following parameters:

```
UpgradeSVNWorkingCopy -all-organizations
```

Examples

If you want to upgrade a single team development workspace in the organization MyOrganization, use the following command:

```
~>UpgradeSVNWorkingCopy MyOrganization MyWorkspace
Starting SVN Working Copy Upgrade
Upgrading workspaces in organization: 'MyOrganization'
The workspace 'MyWorkspace' has been upgraded to the SVN version 1.9.
```

If you want to upgrade all the workspaces in all the organizations, use the following command:

```
~>UpgradeSVNWorkingCopy -all-organizations
Starting SVN Working Copy Upgrade
Upgrading workspaces in organization: 'MyOrganization'
The workspace 'Example Workspace 1' has been upgraded to the SVN version 1.9.
Upgrading workspaces in organization: 'system'
The workspace 'Example Workspace 2' has been upgraded to the SVN version 1.9.
```

In cases where any of the parameters contain whitespace characters, you must include them in double quotes as follows:

```
UpgradeSVNWorkingCopy "My Organization" "My Workspace"
```

CWS sub command: validate

The validate sub command validates and builds a document in CWS. Running this tool for a single document, folder, or project has the same result as validating the same document from the browser using the context menu item Validate. The command can be used as:

```
cws validate <parameterlist>
```

Parameters

Provide the following parameters in the validate sub command:

-organization (-o) <organization name>	: Name of the organization.
-workspace (-w) <workspace name or ID>	: Name or ID of the workspace.
-document (-d) <document qualified name or ID>	: Name or ID of the document

Example

If your organization is called MyOrganization and you have a workspace called MyWorkspace that contains a document with the path MyApplication/Roles/MyRole, then the document can be build using this command:

```
cws validate -organization MyOrganization -workspace MyWorkspace -document  
MyApplication/Roles/MyRole
```

When a parameter contains white space characters, enclose it in double quotes as shown:

```
cws validate -organization "My Organization" -workspace "My Workspace" -document "My  
Application/Roles/My Role"
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: package

The package sub command creates an application package. Depending on the package properties, a model packages from the Collaborative Workspace (CWS) projects. Running this tool has the same result as creating a package of the same project from the browser.

The command can be used as:

```
cws package <parameterlist>
```

Parameters

Provide these parameters in the package sub command:

```
-organization (-o) <organization name> : Name of the organization.
-project (-p) <projectID or name> : Project to be packaged.
-workspace (-w) <workspaceID or name> : Name or ID of the workspace.
```

Example

For example, if your organization is called MyOrganization, your workspace is called MyWorkspace that contains a project called MyProject, then the project can be packaged using this command:

```
cws package -organization MyOrganization -workspace MyWorkspace -project MyProject
```

When a parameter contains white space characters, enclose it in double quotes as shown:

```
cws package -organization MyOrganization -workspace MyWorkspace -project "My Project"
```

The created package is stored at <AppWorks Platform_installdir>/cws/build/MyOrganization/MyWorkspace/package.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: synchronize

The synchronize sub command synchronizes a workspace with the corresponding synchronization directory on the file system of server. Running this tool has the same result as synchronizing the workspace from the browser.

The command can be used as:

```
cws synchronize <parameterlist>
```

Parameters

Provide these parameters in the synchronize sub command:

-organization (-o) <organization name> -workspace (-w) <workspace name or ID>	: Name of the organization. : Name or ID of the workspace.
--	---

Example

If your organization is called MyOrganization, and you have a workspace called MyWorkspace, then the workspace can be synchronized using this command:

```
cws synchronize -organization MyOrganization -workspace MyWorkspace
```

When a parameter contains white space characters, enclose it in double quotes as shown:

```
cws synchronize -organization "My Organization" -workspace "My Workspace"
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: removeworkspaces

The removeworkspaces sub command removes workspaces from the specified organization.

The command can be used as:

```
cws removeworkspaces <parameterlist>
```

Parameters

Provide these parameters in the removeworkspaces sub command:

-organization (-o) <organization name>	: Workspace(s) of a single organization.
-all-organizations	: Workspace(s) of all organizations.
-execute (-x)	: If this parameter is not specified, an overview is produced but workspaces are not removed.
-care	: CARE workspaces.
-development	: Development workspaces.
-psl	: PSL workspaces.
-staging	: Staging workspace.

Provide any combination of these parameters **-care**, **-development**, **-psl**, and **-staging**. No workspaces are removed when none of these parameters are provided.

Provide a specific organization or all organizations. No workspaces are removed when none of these parameters are provided.

For example, to remove all PSLs from an organization called MyOrganization, use this command:

```
cws removeworkspaces -organization MyOrganization -psl -execute
```

To remove all organization-level PSLs and development workspaces from all organizations, use this command:

```
cws removeworkspaces -all-organizations -psl -development -execute
```

If name of the organization contains white space characters, enclose it in double quotes as shown:

```
cws removeworkspaces -o "My Organization" -psl -care -staging -development -x
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: publish

The publish sub command publishes a document in CWS. Running this tool for a single document, folder, or project has the same result as publishing the same document from the browser using the context menu item Publish to Organization. The command can be used as:

```
cws publish <parameterlist>
```

Parameters

Provide these parameters in the publish sub command:

-organization (-o) <organization name>	: Name of the organization.
-workspace (-w) <workspace name or ID>	: Name or ID of the workspace.
-document (-d) <document qualified name or ID>	: Name or ID of the document.

Example

If your organization is called MyOrganization, and you have a workspace called MyWorkspace that contains a document with the path MyApplication/Roles/MyRole, then the document can be published using this command:

```
cws publish -organization MyOrganization -workspace MyWorkspace -document  
MyApplication/Roles/MyRole
```

When a parameter contains white space characters, enclose it in double quotes as shown:

```
cws publish -organization "My Organization" -workspace "My Workspace" -document "My  
Application/Roles/My Role"
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: movesynchronizelocation

The movesynchronizelocation sub command changes the location of the Collaborative Workspace (CWS) synchronization folder in an automated manner. This affects the synchronization location of all workspaces in the AppWorks Platform instance. When running the tool, all the CWS service containers are stopped to prevent any modification to the

synchronization folder while someone is still developing content in one of the workspaces. There can be multiple CWS service containers in case

The movesynchronizelocation tool does not change the location of the build folder; if required, change it manually.

The movesynchronizelocation sub command can be used as:

```
cws movesynchronizelocation <parameterlist>
```

Parameters

Provide these parameters in the movesynchronizelocation sub command:

-destination (-d) <destination>	: New synchronization directory.
-skip-configuration	: Optional. If this parameter is specified, the CWS configuration file located at <install directory>/components/cws/config/cws.properties is not be updated.

If you run the movesynchronizelocation tool with the parameter `-skip-configuration`, the specified configuration file is not updated and the old synchronization directory is reused. To make the changes effective in this case, manually update the configuration file.

For example, change the synchronization directory to `E:\temp\cws\sync` by running this command:

```
cws movesynchronizelocation -destination E:\temp\cws\sync
```

If the path of the destination directory contains whitespace characters, enclose the path in double quotes as shown:

```
cws movesynchronizelocation -destination "E:\temp\cws\sync folder\"
```

The operating system user account of AppWorks Platform must have complete permissions to the new synchronization directory.

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: reloadruntimereferences

The reloadruntimereferences sub command reloads the outdated runtimereferences for a specific workspace in an organization.

The command can be used as:

```
cws reloadruntimereferences <parameterlist>
```

Parameters

Provide these parameters in the reloadruntimereferences sub command:

<pre>-organization (-o) <organization name> -ws (-w) <workspace name or ID> -execute (-x)</pre>	: Name of the organization. : Name or ID of the workspace. : If this parameter is specified, all outdated run-time references are reloaded. If this parameter is not specified, a test run is performed.
---	--

Example

If your organization is called MyOrganization and you have a workspace called MyWorkspace, reloadruntimereferences can be done with this command:

```
cws reloadruntimereferences -organization MyOrganization -ws MyWorkspace -execute
```

If a parameter contains white space characters, enclose it in double quotes as shown:

If a parameter contains whitespace characters, enclose it in double quotes as

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: resetsynchronizer

The resetsynchronizer sub command solves the synchronization problems (in case of unrecoverable errors) in a single workspace. When running the tool, the current synchronization folder of the respective workspace is renamed; in the next synchronization, the workspace content is stored in a newly created synchronization folder as it is in the CWS repository.

The command can be used as:

```
cws resetsynchronizer <parameterlist>
```

Provide these parameters in the resetsynchronizer command:

<pre>-organization (-o) <organization name> -ws (-w) <workspace name or ID></pre>	: Name of the organization. : Name or ID of the workspace.
---	---

Example

If your organization is called MyOrganization, and you have a workspace called MyWorkspace, then the synchronizer can be reset using this command:

```
cws resetsynchronizer -organization MyOrganization -workspace MyWorkspace
```

When a parameter contains white space characters, enclose it in double quotes as shown:

```
cws resetsynchronizer -organization "My Organization" -workspace "My Workspace"
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: upgradesvnworkingcopy

The upgradesvnworkingcopy sub command upgrades the existing team development workspaces to the configured version of Subversion.

The command can be used as:

```
cws upgradesvnworkingcopy <parameterlist>
```

Parameters

Provide these parameters in the upgradesvnworkingcopy sub command:

-all-organizations all organizations.	: Upgrades the available workspaces from
-organization (-o) <organization name>	: Name of the organization.
-workspace (-w) <workspaceID or name>	: Name or ID of the workspace.

The upgradesvnworkingcopy sub command is used in one of these ways:

- Upgrade the working copy of a specific workspace in a specific organization
- Upgrade the working copy of all workspaces in a specific organization
- Upgrade the working copies of all workspaces in all organizations

To upgrade the working copy of a specific workspace in a specific organization, run the tool with the following parameters:

```
cws upgradesvnworkingcopy -organization <organization name> -workspace <workspace name>
```

Where,

- <organization name> is the name of the organization and <workspace name> is the name of the workspace in which the working copy must be upgraded.

To upgrade the working copy of all workspaces in a specific organization, run the tool with the following parameters:

```
cws upgradesvnworkingcopy -organization <organization name>
```

Where,

- <organization name> is the name of the organization.

To upgrade the working copies of all workspaces in all organizations, run the tool with the following parameters:

```
cws upgradesvnworkingcopy -all-organizations
```

Examples

If you want to upgrade a single team development workspace called MyWorkspace in the organization called MyOrganization, use this command:

```
cws upgradesvnworkingcopy -organization MyOrganization -workspace MyWorkspace
```

If a parameter contains whitespace characters, enclose it in double quotes as shown:

```
cws upgradesvnworkingcopy -organization "My Organization" -workspace "My Workspace"
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

CWS sub command: removeworkspace

The removeworkspace sub command removes a single development workspace for a specific organization.

The command can be used as:

```
cws removeworkspace <parameterlist>
```

Parameters

Provide these parameters in the removeworkspace sub command:

-organization (-o) <organization name>	: Name of the organization.
-workspace (-w) <workspace name or ID>	: Name or ID of the workspace.
-execute (-x)	: The tool removes the workspace if this parameter is specified. A test run is performed and nothing is removed if this parameter is not specified.

Example

If your organization is called MyOrganization, and you have a workspace called MyWorkspace, then the workspace can be removed using this command:

```
cws removeworkspace -organization MyOrganization -workspace MyWorkspace -execute
```

When a parameter contains white space characters, enclose it in double quotes as

```
cws removeworkspace -organization "My Organization" -workspace "My Workspace" -execute
```

For general instructions on how to use CWS command-line tools, see [Using the CWS Command-line Tools](#).

Defining a custom schedule for object pruning

The Object Pruner removes database records that are marked as deleted by the CWS service container. Scheduling the Object Pruner at regular time intervals releases computer resources.

If you do not specify a custom time schedule or define an invalid time schedule, the object pruner runs according to the default time schedule: 5:00, 12:00, 20:00.

The Object Pruner runs daily. In your custom time schedule, you must specify time values in a 24-hour format. For example:

- 0 (=0:00)
- 15 (=15:00)
- 8:15
- 23:00

Before you begin:

- You must have the role of a System Administrator to perform this task.
- You must be able to access the configuration file on the server where AppWorks Platform is installed to modify the configuration settings and control the schedule of the Object Pruner.

To define a custom schedule:

1. Open the file <AppWorks_Platform_installdir>\components\cws\config\cws.properties.
2. Add the following code to the file:

```
cleanup.cws.temporary.objects.schedule=<schedule>
```

3. Replace <schedule> with your custom time schedule.
For example, to run the Object Pruner each day at 3:00 and 23:30, add the following code: cleanup.cws.temporary.objects.schedule=3:00,23:30
4. Restart the CWS service container.
 - If an invalid custom schedule is provided, an error displays when the service container starts and the Object Pruner does not run, for example:
cleanup.cws.temporary.objects.schedule=9:00,13:30,19:60
 - If no custom schedule is provided, the default time schedule is used, for example:
cleanup.cws.temporary.objects.schedule=<empty>

Document locking

In team development scenarios, you cannot always avoid merge conflicts. For example, when developing Java code, multiple developers update the same Java class. Merge conflicts can also occur when developing AppWorks Platform applications including business processes, user interfaces, and so on. The merge conflicts on the AppWorks Platform documents are very hard to resolve manually without introducing any inconsistencies. In cases where Subversion (SVN) can resolve the conflict using auto-merge, you cannot be sure if a well-formed XML document is produced with respect to the XML schemas for the various AppWorks Platform document types.

Subversion provides a feature to lock the documents in the SCM repository. When you own a lock on the document, you can ensure that other users cannot commit any changes on the document. If other users try to modify the document that is already locked, an error message containing information about the user who locked the document is displayed. After the lock on the document is removed, the user can make his changes on the updated document.

Before you begin:

- The workspace must be configured to a Source Control Management (SCM) system, for example, the SVN.

Locking documents

When you modify a document from CWS, the document is locked in the SCM repository. In certain types of documents such as CSS, HTML, Java, Javascript, Folders, Text, and MessageBundle files, a lock is not required because problems do not occur when merge conflict arises. The list of locked documents can be viewed.

When you perform an action that modifies a document, CWS queries the SCM repository to check whether the document can be locked. If a lock can be obtained, the document path in the SCM repository is locked and you can modify the document. If the path is already locked, CWS does not allow you to modify the document and an error is displayed containing information about the user who is owning the lock.

In such cases you receive the following error message:

You cannot modify the document '<document>'. It is being modified by '<user name>' in workspace '<workspace>' from organization '<organization>'. To avoid SCM merge conflicts, the document is locked in the SCM system.

Where,

<document>	name of the document
<user name>	name of the user modifying the document
<workspace>	name of the workspace
<organization>	name of the organization

You cannot modify an older version of a document when a new version is in the repository. If you try to modify an older version, you will get the following error message:

You cannot modify the document. The document is out-of-date. Incorporate the latest changes and try again.

As indicated in the error message, you can perform the required changes only after incorporating the latest changes in your workspace. If other users do not lock the document, you can lock the document and make the required changes. Incorporating the latest changes can introduce SVN merge conflicts, which need to be resolved using client-side SVN tools such as TortoiseSVN. In that case, you will get the following error message:

There are SCM conflicts detected which needs to be resolved. Solve the specified conflicts on the file system and try again.

Releasing locks

You can release the locks that you have introduced in the repository in the following ways:

- **Make Changes Available** - When you make your local changes available to others, you no longer have local changes on any of the version controlled documents. Therefore, there is no need to keep the locks on the documents for which changes have been committed. All your locks will be released in the SCM-repository.
- **Revert Local Changes** - Alternatively, you can revert all your local changes. Again, after that you no longer have local changes, which mean that after reverting local changes all locks can safely be released.
- **Delete the Workspace** - Whenever a workspace is deleted, all corresponding locks will be released from the repository.

There can be situations when a document is locked in the repository even though there are no local changes and the user is not aware of owning the lock. Alternatively, project teams can change and there can be locks in the repository owned by the old project members. The following steps explain the actions you can perform in such situations and the consequences:

- **Break Locks** - You can manually break the locks in the repository. Open the Repository Browser by using client-side Subversion tools such as TortoiseSVN. In the context menu of an arbitrary folder (revisioned or unrevisioned), click TortoiseSVN > Repo-browser. In the Repository Browser window, enter the URL of the repository or you can browse to the path where the locks are not released and click Break Lock. You must be aware of the consequences of breaking locks
- **Stealing Locks** - Client-side SVN tools also enable you to steal locks when another user locks documents in the repository. You can break the lock of the other user, and create and own a new lock. The working copy of your document will contain the corresponding lock token. Stealing locks is not recommended. However, CWS can deal with locks in the workspace synchronization folder (in Subversion, this is called the working copy). If you edit a document from the browser that has a lock on the working copy, CWS allows further modifications.

Breaking locks owned by other users is strongly discouraged because the user may experience merge conflicts when working on the document again. In such situations, if the AppWorks Platform documents such as User Interfaces or Business Processes are merged properly, the merged document may contain invalid XML. If the XML is valid, the user may experience Javascript errors when opening documents in the browser.

Using models of other applications

AppWorks Platform provides a rich development environment to design integrated applications that contain entities, user interfaces, business processes, roles, and so on. While developing such applications, often need to use models deployed as part of other applications (developed by others or even by yourself) arises.

For many types of models, using such models of other applications can be done through the runtime references. See [Runtime References](#) for more information, including a list of all the types of models leveraging this approach.

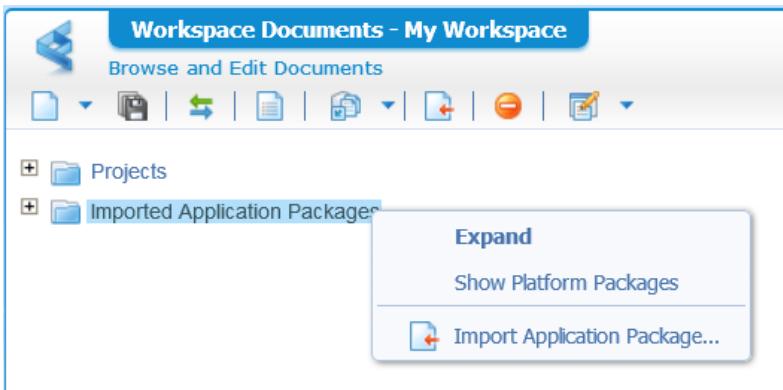
With the introduction of modeling the business domain through entities, there are many scenarios in which one also wants to use entities of other applications. A natural approach is to introduce a new type of runtime reference for entities. However, when using entities of other applications, the runtime reference approach is not rich enough to cover all the requirements. Therefore, an alternative approach is used in which model contracts are delivered as part of the application package. See [Model contracts](#) for more information.

If you want to use entities of other applications, but want to customize them or create subtypes of them to make them suited for your business purpose, you can make use of model packages. See [Model packages](#) for more information.

AppWorks Platform also provides entities that can be used like Model contracts or Model packages. These packages are already installed during AppWorks Platform installation, and are available for all organizations.

To make the packages visible in CWS:

- From the context menu of **Imported Application Packages**, select option **Show Platform Packages**.



Model contracts

With model contracts, AppWorks Platform provides a way to reuse models that exist in other applications.

To provide permission for reuse of a model to other developers:

- Select the Available for use by others check box.
The permission has to be given per model and attribute of the model. For example, for each property and relationship of a specific entity.
- When the application package is created, the check marked attributes are collected in a contract for each model that was selected for reuse.

The model contract gives you an instrument to indicate which parts of a model will be backward compatible. It provides you with freedom to develop better versions of the model and the application in which the model is used.

Before you begin:

- You must have the Developer role.

The characteristics of model contracts are as follows:

- Model contracts are always included in an application package.
- An application package can contain zero or more model contracts.
- A model contract can contain zero or more attributes, for example, properties and relationships for entities.
- Model contracts cannot be imported, updated, or removed individually. All model contracts in the application package are imported, updated, or removed at once.
- It is possible to import zero or more than one application packages in a workspace.

The following model type is available for reuse once imported:

- Entity - To add a relationship to the imported entity for using its available data in other entities.

The flow of using model contract is as follows:

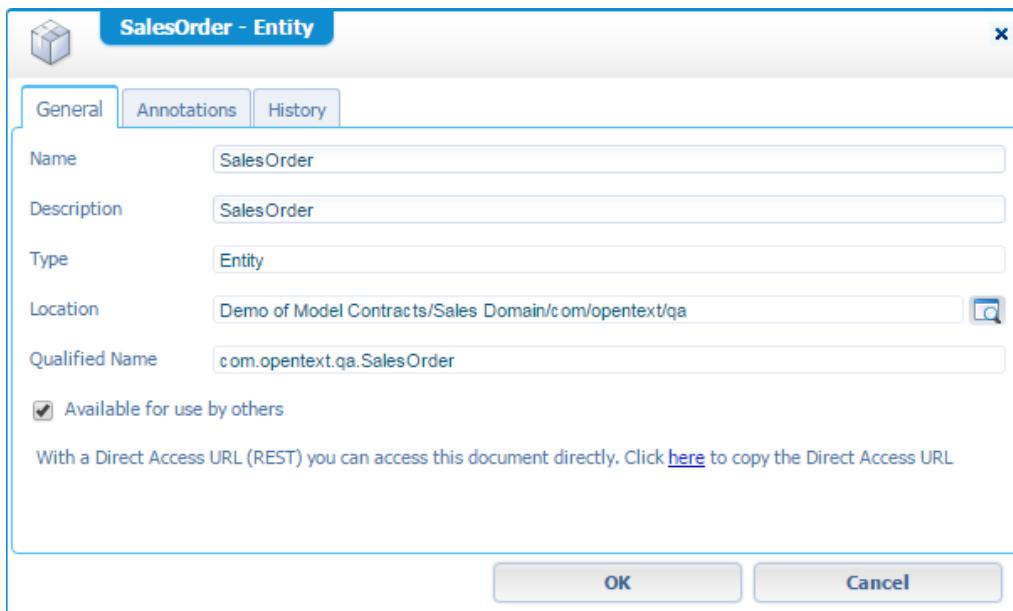
- Selecting models - Developer 1 selects one or more models and their attributes for which a contract must be created so that they become available for use in other applications.
- Creating an application package - Developer 1 generates an application package, containing model contracts for the selected models.
- Importing an application package - Developer 2 imports the application package containing the model contracts in the development workspace where the reuse will be done. The imported data is visible in the development workspace in the Imported Application Packages section. Developer 2's designed projects are visible in the Projects section. When the application package is imported, Developer 2 is able to start creating references to the imported models.
- Managing an imported application package - Developer 2 can open, view, update, or remove the imported application package. Updating includes both upgrading and downgrading to higher or lower versions.

Selecting models for which contracts are to be generated

The models for which contracts are to be generated must be selected.

To select a model for which a contract is to be generated:

1. Select the model.
2. Right-click, select **Properties**.
3. In the **General** tab, select **Available for use by others**.
4. Click **OK** to save.



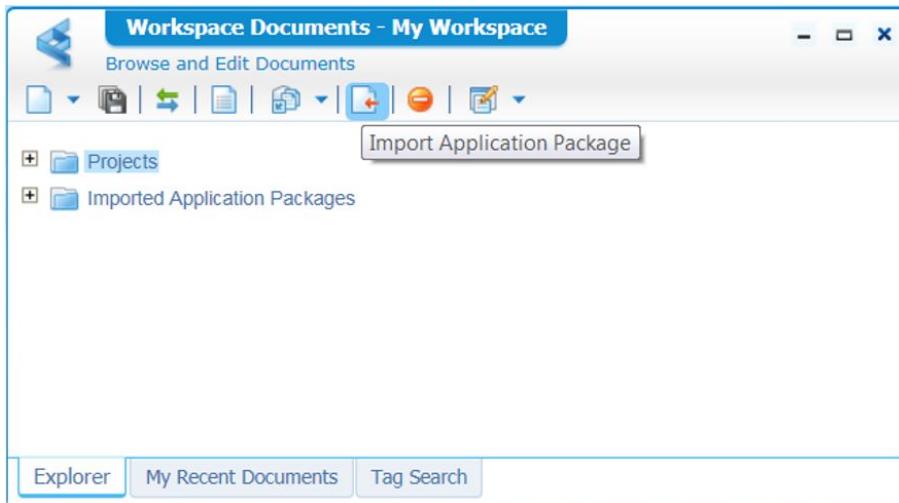
Creating an application package with the model contracts

When a package is created from the project containing the selected models, the model contracts are automatically generated and added to the package. See [Creating and Downloading Application Packages](#) for information on how to create a package.

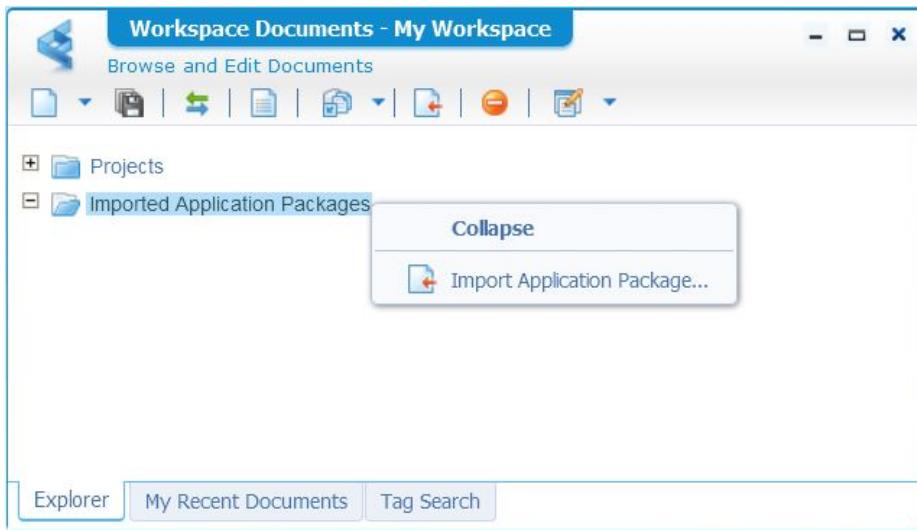
Importing an application package with the model contracts

To import model contracts, import the application package containing them:

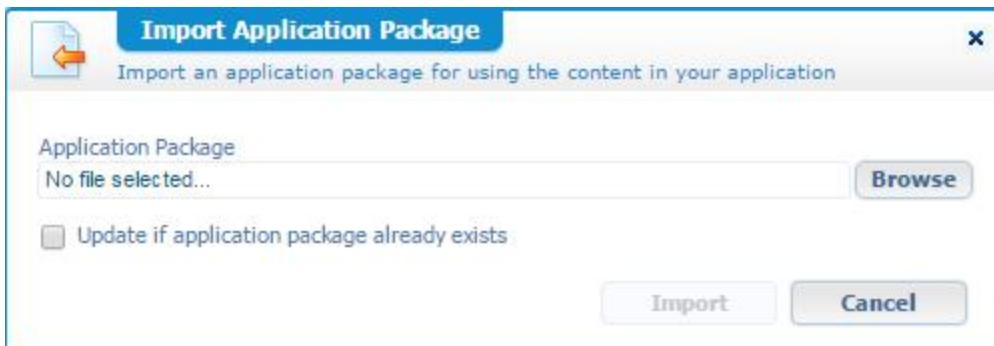
1. Open the import dialog in one of the following ways:
 - Click (Import Application Package) in the toolbar of the Workspace Documents.



- Right-click **Imported Application Packages** and select **Import Application Package**.



- Browse to the package that contains the required documents and import it.



- When the import is finished, the package with its contents appears in the tree section of Imported Application Packages in Workspace Documents.
 - It is possible that the selected package does not contain anything to import. In that case, the user is notified about this.
 - The storage of imported content on either file system (after a synchronize) or SCM (for example, SVN) is same for projects and models.

Model dependencies

It is possible that an imported model package has dependencies to other model packages. In that case, the imported package icon shows a red minus sign. The actual dependencies can be found in the imported model package Properties. The packages that are shown there may have to be imported too. See [Managing an imported model package](#) on how to open the imported model package Properties.

Multiple import situations

There are multiple situations in which you want to have an import functionality in your workspace. There is a distinction made in the UI for the level of import.

- On workspace level, packages can be imported to which you can refer (this page).
- On project level there is a way to import Business Process Model related information that results in a new document in your project ([See Importing an XPDL File](#)).

Managing an imported application package

This topic describes the various ways an imported application package is managed.

Opening a model from an imported application package

A model from an imported package contains information on the application package from which the model originates.

To open a model from an imported application package:

- In the **Workspace Documents**, right-click <model> and select **Open**.
The <Model> window opens, displaying the following information.

Field	Description	Edit Rights
Name	Name of the model	Read only
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only

Viewing properties of an imported application package

The imported application package properties contain information on the application package from which the imported data originates.

To view properties of an imported application package:

- In the Workspace Documents, right-click **Imported Application Packages** and select **Properties**.
The Application Package window opens, displaying the following information.

Field	Description	Edit Rights
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only
Build Number	Build number of the package	Read only

Updating an imported application package

Like importing model contracts by importing the application package where it resides, the update of model contracts implies the update of the complete imported application package.

To update an application package:

1. Follow the procedure as described for [importing an application package](#).
2. In the Import Application Package window, select **Update if application package already exists**.
The update of the selected package is performed.

Updating a model package with a contract package

The update functionality supports the possibility to update a model package with a contract package and vice versa.

Removing an imported application package

Like importing model contracts by importing the application package where it resides, the removal of model contracts implies the removal of the complete imported application package. Before the model contract is removed, the Used By check is performed automatically, which checks if it is used in other places.

If you do not want to delete but to check where the model contract is used, right-click on the application package or the folder or the model and then click Used By for an overview of where the selected part is used.

To remove an imported package:

1. Right-click on an imported application package and select **Delete**.
2. The Used by check is performed to see if the imported application package is in use.
3. If the package
 - a. Is in use, it is never removed
 - b. Is not in use, the removal is finished when the application package disappears from the Imported Application Packages section.

Model packages

With model packages, AppWorks Platform provides a way to adjust models that exist in other applications to your needs. The model package gives you an instrument to share model information. It provides you with freedom to reuse and adjust models from other applications.

Before you begin:

- You must have the Developer role.

The characteristics of model packages are:

- A model package is always generated together with an application package
- A model package can contain zero or more models
- A model package can contain several entities with all their customizable building blocks

- A model package can have dependencies to other model packages
- It is possible to import zero or more model packages in a workspace

The following model type is available for reuse once imported:

- Entity - To customize imported entities
- (Role)

The flow of using a model package is as follows:

- [Creating a model package](#) - Developer 1 generates a model package, containing all model information from a project.
- [Importing a model package](#) - Developer 2 imports the model package in the development workspace where the customizations will be done. The imported data is visible in the development workspace in the Imported Application Packages section. Developer 2's designed projects are visible in the Projects section. When the model package is imported, Developer 2 is able to start customizing the imported models, and using them in his own application.
- [Managing an imported model package](#) - Developer 2 can open, view, update, or remove the imported model package. Updating includes both upgrading and downgrading to higher or lower versions.

Creating a model package

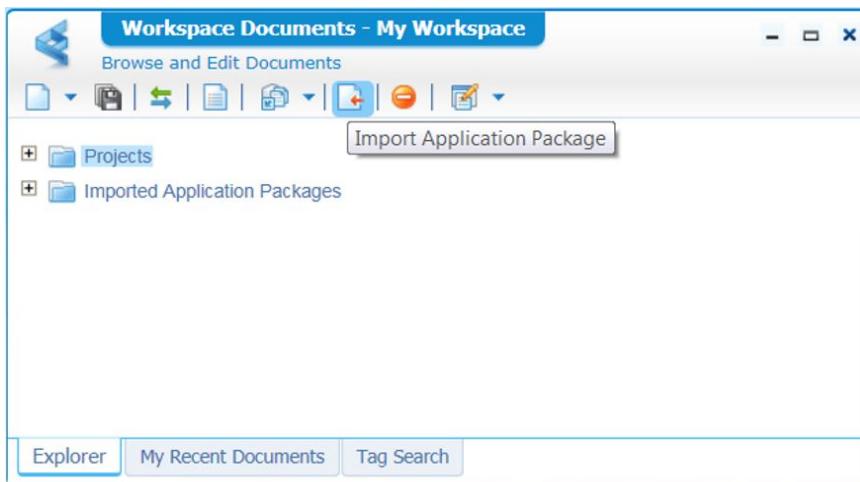
After the package property Create Model Package has been selected as described in [Setting Properties of an Application Package](#), a model package is created and downloaded together with an application package as described in [Creating and Downloading Application Packages](#).

Importing a model package

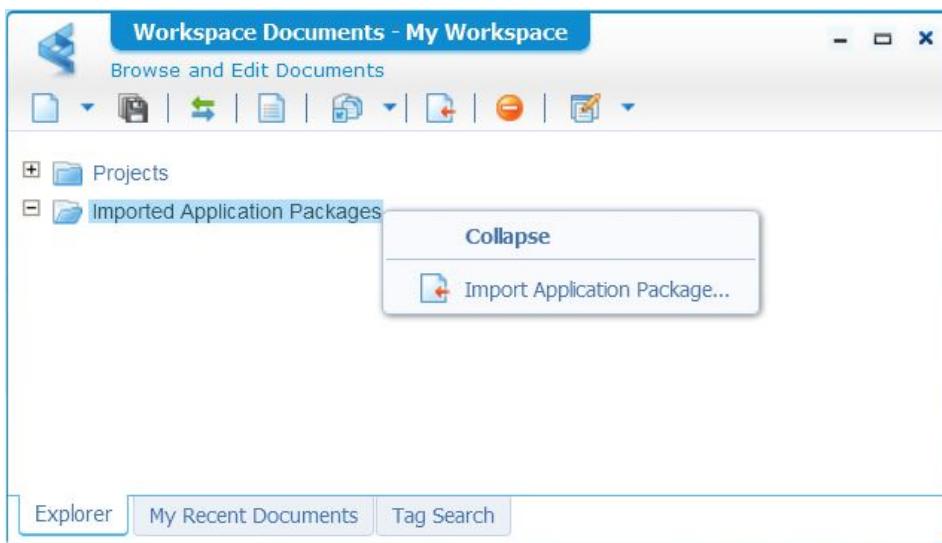
To import models, import the model package:

1. Open the import dialog in one of the following ways:

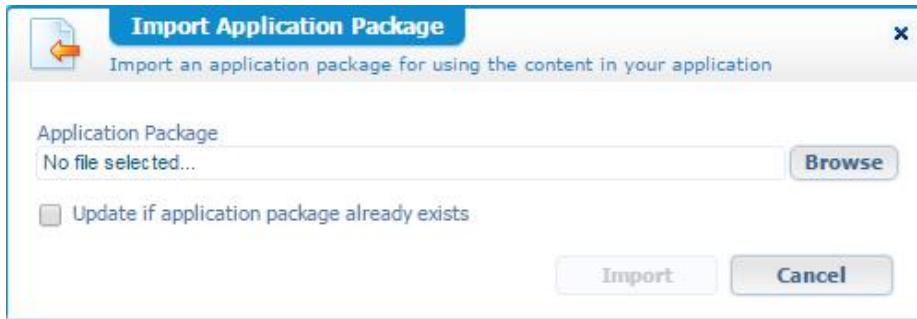
- a. In the toolbar of the Workspace Documents, click  (Import Application Package).



- b. Right-click **Imported Application Packages** and select **Import Application Package**.



2. Browse to the package that contains the required documents and import it.



3. When the import is finished, the package with its contents appears in the tree section of Imported Application Packages in Workspace Documents.
 - a. It is possible that the selected package does not contain anything to import. In that case, the user is notified about this.
 - b. The storage of imported content on either file system (after a synchronize) or SCM (for example, SVN) is same for projects and models.

Model dependencies

It is possible that an imported model package has dependencies to other model packages. In that case the imported package icon shows a red minus sign. The actual dependencies can be found in the imported model package Properties. The packages that are shown there may have to be imported too. See [Managing an imported model package](#) on how to open the imported model package Properties.

Multiple import situations

There are multiple situations in which you want to have an import functionality in your workspace. There is a distinction made in the UI for the level of import.

- On workspace level, packages can be imported to which you can refer (this page).
- On project level there is a way to [import Business Process Model](#) related information that results in a new document in your project.

[Managing an imported model package](#)

This topic describes the various ways of managing the imported model packages.

[Opening a model from an imported model package](#)

A model from an imported package contains information on the package from which the model originates.

To open a model from an imported package:

- In the **Workspace Documents**, right-click <model> and select **Open**. The <Model> window opens, displaying the following information.

Field	Description	Edit Rights
Name	Name of the model	Read only
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only

Viewing properties of an imported model package

The imported model package properties contain information on the package from which the imported data originates.

To view properties of an imported package:

- In the **Workspace Documents**, right-click <imported application package> and select **Properties**.

The Model Package window opens, displaying the following information.

Field	Description	Edit Rights
Package Name	Name of the package	Read only
Package Version	Version of the package	Read only
Build Number	Build number of the package	Read only
Package dependencies	Columns Package name, Version and Build number. Identifier of the model package that contains models that are used by models in this model package. Column Resolved. Indicates if a model package has been imported that contains these models. This may be a package with the specified Name, Version and Build number, but can also be a package with the same Name but different Version and/or Build number.	Read only
Model dependencies	Columns From type, From name. Identifier of an individual model in this package. Column To type, To name. Identifier of an individual model in the selected package in the Package dependencies list, that is used by the (From) model in this package. Column Resolved. Indicates that the (To) model is present in the selected package in the Package dependencies list.	Read only

Updating an imported model package

Like importing models by importing the model package where it resides, the update of models implies the update of the complete imported package.

To update a model package:

1. Follow the procedure as described for [importing a model package](#).
2. In **the Import Application Package** window, select **Update if application package already exists**.
The update of the selected package is performed.

Updating a contract package with a model package

The update functionality supports the possibility to update a [contract package](#) with a model package and vice versa.

Removing an imported model package

The removal of models implies the removal of the complete imported model package. Before the model package is removed, the Used By check is performed automatically, which checks if models from the package are used in other places.

To check where the models from the model package are used:

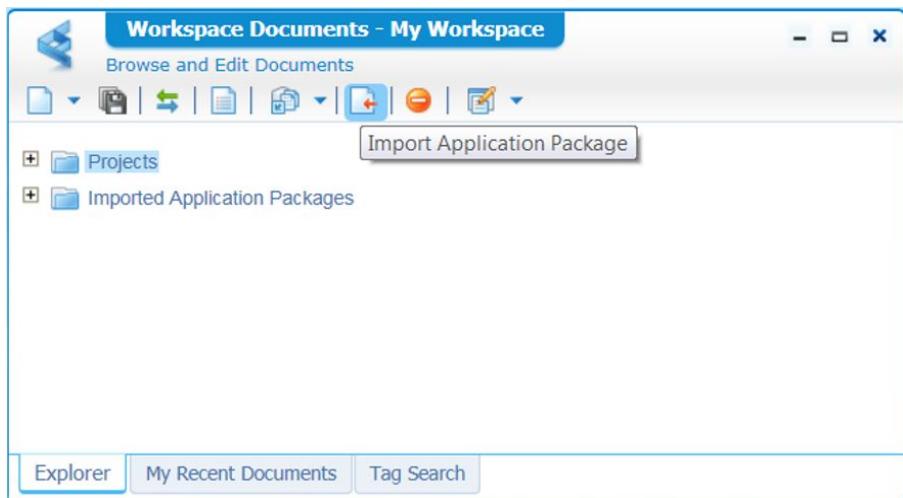
- Right-click on the model package, the folder, or the model and then click **Used By** for an overview.

To remove an imported model package:

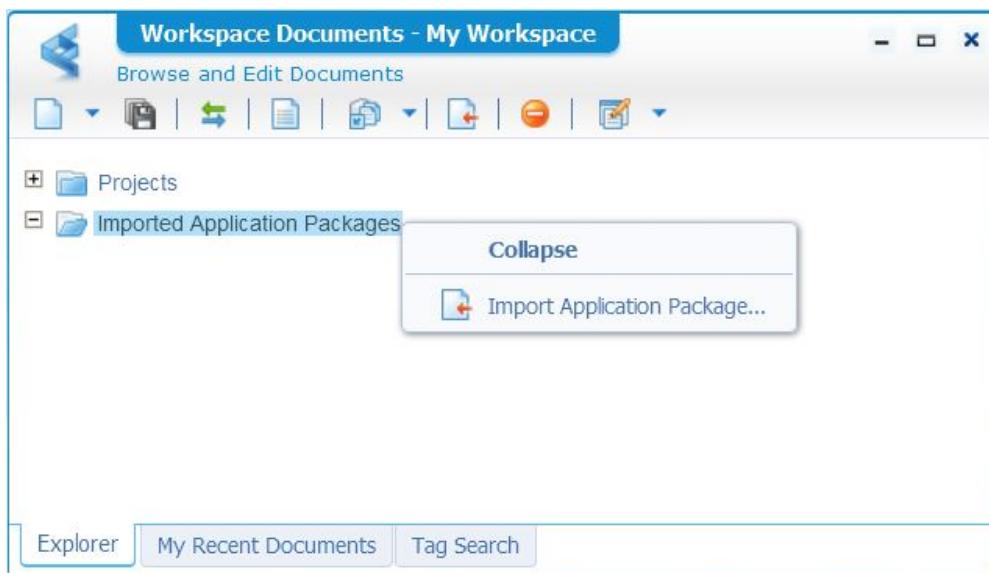
1. Right-click on the imported model package and select **Delete**.
The Used by check is performed to see if models from the imported application package are in use.
2. If models from the package:
 - a. Are in use, the package is not removed.
 - b. Are not in use, the package is removed.
The removal is finished when the model package disappears from the Imported Application Packages section.

Importing a package**To import a package:**

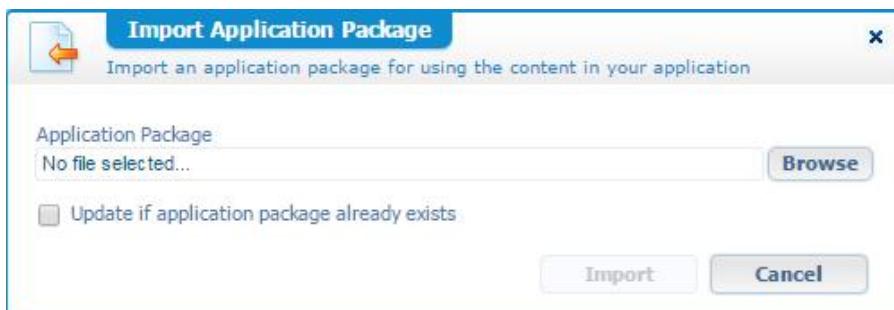
1. Open the import dialog in one of the following ways:
 - a. In the toolbar of the Workspace Documents, click  (Import Application Package).



- b. Right-click **Imported Application Packages** and select **Import Application Package**.



2. Browse to the package that contains the required documents and import it.



3. When the import is finished, the package with its contents appears in the tree section of Imported Application Packages in Workspace Documents.

It is possible that the selected package does not contain anything to import. In that case the user is notified about this.

The storage of imported content on either file system (after a synchronize) or SCM (for example, SVN) is same for projects and models.

Model dependencies

It is possible that an imported model package has dependencies to other model packages. In that case, the imported package icon shows a red minus sign. The actual dependencies can be found in the imported model package Properties. The packages that are shown there may have to be imported too. See [Managing an imported model package](#) on how to open the imported model package Properties.

Multiple import situations

There are multiple situations in which you want to have an import functionality in your workspace. There is a distinction made in the UI for the level of import.

- On workspace level, packages can be imported to which you can refer (this page).
- On project level there is a way to [import Business Process Model related information](#) that results in a new document in your project.

Runtime references

AppWorks Platform provides a rich development environment to design integrated applications that contain user interfaces, business processes, roles, and so on. While developing such applications, you often need to reuse application artifacts deployed as part of other applications (developed by others or even by yourself). To reuse such application artifacts, you need to create a run-time reference in your project. The run-time reference can then be used as a regular design-time document.

Run-time references introduce dependencies on the applications that provide the referenced artifacts. Therefore, to be able to deploy your application, the referenced artifacts must be deployed first.

Before you begin:

- Ensure that the application containing the artifact you want to reuse is installed, and you have the required permissions to access it.

The following types of application artifacts are available for reuse after deployed.

Document	Description
 (Action Template)	Add a run-time Action Template reference, to use it in Rules
 (Application Connector)	Add a run-time Application Connector reference to use it

Document	Description
	for a Web Service
 (Business Calendar)	Add a run-time Business Calendar reference, to use its data in a Business Process Model
 (Business Identifier)	Add a run-time Business Identifier reference, to use it for identification of process instances in the Process Instance Manager.
 (Business Measure)	Add a run-time Business Measure reference to use it as data source for creating KPIs or in a User Interface for performance visualization in dashboards
 (Business Process Model)	Add a run-time Business Process Model reference, to use it directly or to use it in a Business Process Model
 (Case Model)	Add a run-time Case Model reference to manage cases in a Business Process Model
 (Composite Control)	Add a run-time Composite Control reference for use in a User Interface
 (Condition Template)	Add a run-time Condition Template reference to use it in Rules
 (Content Map)	Add a run-time Content Map to be used in a Data Transformation
 (Decision Table)	Add a run-time Decision Table reference to your business process model to reuse the Rules
 (KPI)	Add a run-time KPI reference to reuse the indicators in the dashboards
 (Process Monitoring Object)	Add a run-time Process Monitoring Object reference to use it as a data source in Business Measure
 (Runtime Role)	Add a run-time Role reference to reuse it in your Business Process Model
 (User Interface)	Add a run-time User Interface reference to use it in a Business Process Model
 (Web service Interface)	Add a run-time Web service interface to reuse the Web Services. The run-time Web Services and XML schema run-time documents are received in the schema folder

To reuse existing run-time content instead of recreating the same content at design-time:

1. In the **Workspace Documents** (Explorer), open <solution>, right-click <project> or <folder> and select **Add Runtime Reference > Other**.
The New Runtime Reference dialog box opens, displaying all the runtime documents.
2. Select the document that you want to refer.
A window pertaining to the selected document opens, displaying the run-time information containers.
3. Browse through the containers and select the run-time document that you want to refer.
4. Click **Next**.
The next screen of the window opens with the details of the selected run-time document.
5. In **Additional Instructions**, provide details to install the run-time reference, and click **Finish**.
The window closes.

The run-time document is added to the existing project as a reference.

In team development scenarios, run-time references may be incorporated into your workspace without deploying the providing application on your development environment. In those cases, you can still build and package the enclosing projects. However, while publishing the project, for each run-time reference, the computer will check whether a compatible version of the referenced artifact is deployed. In case the referenced artifact is not available or when its interface contract has changed, publishing will fail.

Reusing XML schema

The XML schemas that are used in Web services can be used in applications also. But, you cannot use the XML schemas that are created and not used in a Web service. For using such schemas, you need to create a Web service with the schemas that you want to reuse.

To reuse XML schema:

1. Create a Web service in any of the following ways:
 - **To generate AppWorks Platform Web service operations for External Web services:**
 - Create WSDL that contains at least one Inline schema that imports or includes all the required schemas. (The schemas that are to be reused in an application).
 - Generate Webservice operations.
 - **To generate Web Service Operations on Custom Logic:**
 - Create [XML Schema](#) that imports or includes all the required schemas and ensure that the schema has at least one schema fragment.
 - a. Generate [Webservice operations](#).
2. Install the application package at the location where you want to reuse the schema.

3. Add a run-time Web service interface:
 - a. In the **Workspace Documents (Explorer)**, right-click <project> or <folder> and select **Add Run-time Reference > Other**.
The New Run-time Reference dialog box opens, displaying all the runtime documents.
 - b. Select the document **Web Service Interface**.
A window pertaining to the selected document opens, displaying the runtime information containers.
 - c. Browse through the containers and select the run-time document that you want to refer.
 - d. Click **Next**.
The next screen of the window opens with the details of the selected run-time document.
 - e. In **Additional Instructions**, provide details to install the run-time reference, and click **Finish**.
The window closes.
 - f. Navigate to the Web service interface document and expand it.
The XML schema runtime documents exist in the schema folder.
4. Reload the existing Web service interface run-time references in a project to get the XML schema run-time documents (which are available in run-time repository).

Analyzing package dependencies among application packages

AppWorks Platform applications often comprise multiple application packages that might be developed from a single source or multiple sources. If a package is dependent on another package, you cannot deploy it on a target environment until all the packages it depends upon are deployed first. This ensures that all the prerequisites for deploying and running the applications are met.

While developing an application, you might create the entire application or reuse the parts created by others. For more information about reusing application artifacts, see [Runtime references](#). When packaging a project, the system gathers all the references to other projects or packages and generates package dependencies for each of these references.

There may be scenarios where one application package has an unexpected or unwanted dependency on another package. For instance, in case of cyclic package dependencies, a package might be dependent on another package and the other package is in turn dependent on the earlier package (directly by itself or indirectly through other package dependencies).

Assume that there are two packages, P1 and P2, and the package P2 has a dependency on package P1. The following procedure enables you to identify the model in the package that is causing such a dependency.

Assume that there are two packages, P1 and P2, and the package P2 has a dependency on package P1. The following procedure enables you to identify the model in the package that is causing such a dependency.

To identify a model in the package:

1. When deploying an application package, the Application Deployer provides a complete list of all the package dependencies. From this list, you can identify the package dependency that you want to analyze.
2. Identify whether the sources of both the packages reside in the same workspace.
3. Based on the location of the packages, do the following:
 - a. If both the packages, P1 and P2, reside in the same workspace, one of the models of P2 might have a reference to one of the models in P1. You can use the Used By functionality to identify the exact dependency between packages.
 - b. If P1 and P2 do not reside in the same workspace, P2 might have a runtime reference to a model from P1. You must navigate through the project hierarchy to identify the runtime reference. If a model uses a runtime reference located in another project in the same workspace, the system generates a package dependency on that project and not on the package of which a model is referenced.
 - c. If you import P1 to P2, you can use the Used By functionality to identify the model of P2 that uses a model of the imported package P1. Importing packages enables you to use entities from other packages.

If you are using Java Archive Definitions (JAD), including the WS-AppServer variant, you can specify the dependencies on other Java Archives (JAR). If it is an external JAR, you must manually provide the file system path. Optionally, you can type the name of the package that delivers the dependent JAR. A package dependency is generated with the package name that you specified. If the package name is left blank, a package dependency is not generated. See Java Archive Definition Interface in the *AppWorks Platform Administrator's Guide*.

Correcting absolute and relative paths in web artifacts

With the support of Organization Level Styling, there is a separation between artifacts that are available for all organizations and those that are available only in a specific organization. The main advantage is that customers can achieve different styling per organization. In addition, developers gain from this functionality as they can now test their own changes to Web artifacts without any impact on the work of other developers or testers who work on the same AppWorks Platform environment.

By making the AppWorks Platform Web front end multitenant, absolute paths to Web artifacts starting with `/cordys/` will be mapped to the shared Web content instead of the organization-specific Web content. While publishing artifacts from a development workspace, these artifacts will be available in the run time of the respective organization

only. If any of these artifacts link to another artifact through an absolute path starting with `/cordys/`, the system may not be able to find the linked Web artifact.

While developing user interfaces with the AppWorks Platform UI designer, you do not need to consider absolute or relative paths to JavaScript (JS) libraries or Cascading Style Sheet (CSS) files that are linked. You only need to include the JavaScript libraries and CSS files using the available controls in the AppWorks Platform UI designer. Then, the system ensures that the paths are correctly generated in the runtime counterpart of that user interface.

Organization-specific Web artifacts such as HTML, CSS, JS libraries, and images must be retrieved from the organization-specific Web location. To achieve this, all references to such libraries must be relative as absolute references starting with `/cordys/` will be mapped to the shared Web location.

To benefit from organization level styling, all absolute paths must be corrected to their relative equivalents.

Correcting absolute paths

The correct way to refer to artifacts on the organization-specific Web server is by using relative paths. When a form or HTML library includes the `application.js`, this results in the `BASE` element to be set. Generic information about the `BASE` element can be found [here](#). In the case of including `application.js`, the `BASE` element is set to the organization URL, `/home/<organization name>`. For Web artifacts loaded through `/cordys` and that come from the Shared space, the `BASE` element is set to `/cordys`. As a result of setting the `BASE` element, references to other Web artifacts must be the same as their qualified names. The following table displays some examples of old absolute paths and their relative counterparts using qualified names (new path).

Source artifact	old path	new path
<code><organization URL>/a/b/c.html</code>	<code>/cordys/a/b/icon.png</code>	<code>a/b/icon.png</code>
<code><organization URL>/a/b/c/mystyles.css</code>	<code>/cordys/a/b/c/background.png</code>	<code>background.png</code>
<code><organization URL>/b/c.html</code>	<code>/cordys/d/e/icon.png</code>	<code>d/e/icon.png</code>
<code><organization URL>/a/b/c.caf</code>	<code>/cordys/a/b/utils.js</code>	<code>a/b/utils.js</code>
<code><organization URL>/a/b/c.html</code>	<code>/cordys/a/b/style.css</code>	<code>a/b/style.css</code>
<code><organization URL>/a/b/c/mystyles.css</code>	<code>/cordys/f/g/h/background.png</code>	<code>../../../../f/g/h/background.png</code>
<code><organization URL>/b/c.caf</code>	<code>/cordys/d/e/style.css</code>	<code>d/e/style.css</code>
<code><organization URL>/b/c.html</code>	<code>/cordys/a/b/utils.js</code>	<code>d/e/utils.</code>

References to Images, Javascript and CSS files from XForms and HTML files need to be handled with base relative urls. This means that the new URL is relative to the BASE url. Note that for references from within a CSS file to for example an image this is different as CSS files do not support base relative urls. From CSS files, url references must be relative to the CSS file itself.

One tricky thing is that the BASE element is set only after including the application.js library. The inclusion of the `application.js` library itself is therefore done without the BASE element being set. As a result, the reference to `application.js` must use the correct relative path, starting from the location where the source artifact is stored. These relative paths consist of two parts:

- The part to navigate to the organization URL
- The qualified name of the linked Web artifact

The following table displays examples of relative paths to include the `application.js`.

From page	Refer to
<organization URL>/a/b/c.htm	../../wcp/application.js
<organization URL>/d.htm	wcp/application.js
<organization URL>/wcp/e/f.htm	../application.js

Correcting invalid relative paths

Application artifacts such as HTML libraries, CSS files, and JavaScript libraries are stored in CWS (Collaborative Workspace) as regular text files. CWS cannot recognize references from such artifacts to other artifacts. This makes it impossible to update such references whenever the relative path changes. As such, relative paths may become invalid. The scenarios in which this occurs are:

- Moving the artifact that includes the relative reference to a different location
- Moving the referenced artifact to a different location

In both scenarios, the developer must manually correct the relative reference to ensure the application functions correctly.

For example, when a CSS file includes an image and you move the CSS file without moving the image, you need to update the image URL in the CSS file to the correct relative path.

Preperation for dynamic base URL and future changes

Currently, AppWorks Platform supports only base-relative URLs. However, the upcoming AppWorks Platform releases may also support the regular relative URLs.

To indicate that a form uses base-relative URLs, the attribute `cordysDynamicallyInsertBase` with the value "true" must be added to the `HTML` tag of each page as shown below. If this

attribute is present, then AppWorks Platform will dynamically (through JavaScript) insert the BASE tag.

Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"> <html cordysDynamicallyInsertBase="true">
<head> </head> <body> <p>Have you seen our <a href="cages/birds.gif">Bird
Cages</a>?</p> </body> </html>
```

Caution: Your pages may work fine in the current release even though the `cordysDynamicallyInsertBase` attribute is not added to the HTML tag. However, the upcoming releases will not add the BASE tag if the `cordysDynamicallyInsertBase` attribute is not present. Therefore, it is recommended that you add the `cordysDynamicallyInsertBase` attribute to the HTML pages for continual support.

Creating and downloading application packages

You can deploy a business solution on a AppWorks Platform environment by creating the required application packages and delivering them to that environment. A business solution typically consists of multiple application packages, which are likely to have dependencies between them. Every project in CWS corresponds to a single application package.

Depending on package property Create Model Package, an application package or an application package + a model package will be created and downloaded. Both have the same file name but have a different file extension.

Before creating the application package, AppWorks Platform first determines whether the project contains validation errors or warnings. If there are validation errors, the validation of the project fails and no application package is created. If there are validation warnings, these are included in the packaging progress details but do not block the application package creation. If the application package creation fails, the packaging progress details include meaningful pointers on the reasons for the failure.

To create and download an application package for a project:

1. In **Workspace Documents**, select the project to be packaged.
2. Right-click the project and select **Packaging > Create Package**.
The packaging process starts in a separate window. The Download Package button is enabled if the project is validated without errors and the application package is created.
3. To download the application package, click **Download Package**.
The application package is downloaded to your local machine.

To download the most recent application package of a project that is already available for download:

In this case, there is no need to create an application package first. This option is available only if there is a package already available with the same Package File Name generated while creating a new package.

1. In **Workspace Documents**, select the project for which you want to download the most recent application package.
2. Right-click the project and select **Packaging > Download Latest Package**.
The latest application package is downloaded to your local computer.

See [Setting Properties of an Application Package](#) for the procedure to modify the properties of an application package.

Deploying application artifacts to the AppWorks Platform_installdir

Although it is possible to deploy application artifacts to the <AppWorks Platform_installdir>, there is no explicit way to exactly specify which artifacts are to be deployed there. It is predefined in the platform about the type of application artifacts that can be deployed to the <AppWorks Platform_installdir> by default. The following is the list of all such application artifacts:

- Localization Bundles.
- text files (*.txt)
- Java archives (*.jar)
- All documents of unknown type. (You can know if a document is of the 'unknown type' by opening the document properties. The 'Type' field will have the 'Type unknown' value.)

Some of the paths in the <AppWorks Platform_installdir> are marked as reserved locations. The platform will not allow any custom application to deploy file artifacts to those locations. The following is the list of reserved locations:

- <AppWorks Platform_installdir>/Web
- <AppWorks Platform_installdir>/web
- <AppWorks Platform_installdir>/webroot

Whenever file artifacts are defined to be deployed in any of the reserved locations, an error is displayed every time you try to publish, package, or deploy such artifacts. If you intend to deploy artifacts to the AppWorks Platform web server, this must be done using a Web Library Definition; see [Deploying artifacts to the AppWorks Platform web server](#).

Deploying artifacts to the AppWorks Platform web server

While installing an application package, AppWorks Platform artifacts such as business processes, user interfaces, roles, and so on are deployed to special runtime repositories. The Web Library Definition is a dedicated document type that is used to deploy artifacts on the AppWorks Platform Web server. While publishing a Web Library Definition, all the artifacts that are in its scope are deployed to the AppWorks Platform Web server. Some

types of application artifacts that are typically deployed to the AppWorks Platform Web server are Web pages (.htm or .html), style sheets (.css), JavaScript libraries (.js), and images (.jpg, .png, .gif, etc.).

See [Creating a Web Library Definition](#) for details on creating a Web Library Definition.

Artifacts are deployed to the AppWorks Platform Web server according to their qualified name. It is recommended to mark the Web-content folder of a Web Library Definition as the starting point of the qualified name of all contained artifacts.

Creating a web library definition

Before you begin:

- You must have uploaded web files into the project.

A Web Library Definition enables you to package and deploy files such as .htm, .html, and .js that are used in the application being created to the AppWorks Platform web server. A Web Library Definition bundles them in a suitable format so that they can be deployed within an application and used at run time.

1. Select a starting point and click  to open the Web Library Definition editor.
2. Enter the following information.

Field Name	Description	User Action
Name	Name of the Web Library Definition file	Provide a name
Description	Description of the Web Library Definition	Provide a brief description
Web-content Folder	The folder from which the web files will be included. These web files will be used in the Application Package. Note: The documents in this folder will be validated and published as per the publish logic of Web Library Definition.	Next to the field, click to browse and select the source content project/folder

3. Click .
- The Save Document dialog box opens.
4. Provide the **Name** and **Description**, and click  to browse and select the location to save the Web Library Definition file. The folder path where the Web Library Definition will be created appears in the **Save in Folder** field.
5. Click **Save**.
- The Save Document dialog box closes.

6. Close the **Web Library Definition** window.

The **Web Library Definition** is created with the provided name and is stored at the selected location. When you package the Web Library Definition, it stores the packaged web files that are in the scope of the Web Library Definition, in the Application Package (**.isvp**) file.

After you complete this task:

- [Publish](#) the Web Library Definition to an organization.

When deploying an application that contains a Web Library Definition (or when publishing such a project), the documents in the corresponding Web content folder will be deployed to the AppWorks Platform web server. The exact location at which the documents will be deployed is based on their qualified name.

Impact analysis across development workspaces

Irrespective of a bug fix or a change to a public API, the developers must determine the impact of their changes. To view the impact, you must know where the changed artifact is used. The usage of an artifact by other artifacts within the same workspace can be identified through the Used By functionality. Whenever an artifact is reused in other applications that are developed in other workspaces through the run-time references, the platform does not provide out-of-the-box functionality to identify these places.

To identify which applications are (re)using a certain document, Collaborative Workspace (CWS) can be configured to generate details of the package dependencies caused by the run-time references. When packaging a project, CWS creates two files: the application package and a package dependency file. The package dependency file contains information about the artifacts that are shipped with the application package, the artifacts that are reused, and where exactly these artifacts are used.

Caution: The impact analysis functionality described here can only be performed on artifacts that are delivered through application packages of the .cap format. The package dependency file will only be generated for run-time packages, not for staging packages.

Steps to enable the generation of package dependency files

Although you can download the application package from the browser immediately after packaging a project, the package dependency file is not available for download. You must manually take it from the following file system location:

```
<CWS_build_directory>/<organization>/<workspace>/<package_file_name>
```

Where,

- <CWS_build_directory> is the location of the CWS build folder
- <organization> is the name of the organization

- <workspace> is the name of the workspace
- <package> is the package file name. See [Setting Properties of an Application Package](#) for details on package file names.

Understanding the contents of the package dependency file

The package dependency file is an XML file consisting of the following high-level structure:

```
<Project buildNumber="1" name="Project X"
    packageID="5e00188d-e06a-4fd6-ad0e-060903cc5b0a"
    packageName="My Company Product X"
    projectID="660dbc18-c604-4c7e-95f1-7edd10eeadf5" version="1.0"
    xmlns="http://schemas.cordys.com/cws/applicationdependencies/1.0">
    <Delivers/>
    <Uses/>
</Project>
```

The `Project` element contains the following attributes:

- **name**: the name of the project
- **projectID**: the design-time identifier of the project
- **packageID**: the run-time identifier of the package
- **packageName**: the name of the package as defined in the package properties of the originating project
- **version**: the version of the package as defined in the package properties of the originating project
- **buildNumber**: the build number of the package as defined in the package properties.

The `Project` element contains the following child elements:

- **Delivers**, containing information (in the form of `Artifact` elements) about the contents of the package itself
- **Uses**, containing information on which artifacts of other packages are used and where.

Delivers element

The `Delivers` section contains one or more occurrences of the following XML:

```
<Artifact documentID="e4c4df5f-6a52-4892-9dc8-8858aa18e8de" name="C"
    path="/Product Y/com/y/C" qname="com.y.C">
    <ArtifactID>f9cf86d4-e074-415b-8d53-dd14199cb148</ArtifactID>
    <ArtifactID>f9cf86d4-e074-415b-8d53-dd14199cb149</ArtifactID>
    <Artifact documentID="123" name="D" path="/ProductY/com/y/C/D"
    qname="com.y.C.D">
        <ArtifactID>f9cf86d4-e074-415b-8d53-dd14199cb150</ArtifactID>
    </Artifact>
</Artifact>
```

The **Artifact** node describes the details of an artifact in the application package. It contains the following information:

- **name**, the name of the artifact as known in CWS
- **qname**, the qualified name of the artifact as configured in CWS
- **path**, the location and name of the artifact as known in CWS
- **documentID**, the ID of the object as known in CWS
- **ArtifactID**, the ID of the corresponding artifacts in the application package. Note that an artifact can have multiple **ArtifactIDs**.

An **Artifact** node can contain child **Artifact** nodes. This can happen when a model in CWS consist of multiple model parts that result in artifacts in an application package. This is the case for example in the Webservice model, where the Webservice Interface can contain multiple Webservice Operations. The general rule that can be applied here is that a certain artifact is referenced when it is used, or when one of its child artifacts are used.

Uses element

The **Uses** section contains zero or more occurrences of the following XML:

```
<Package name="My Company Product X" id="5e00188d-e06a-4fd6-ad0e-060903cc5b0a"  
version="1.0" build="1">  
    <ArtifactDependencies>  
        <ArtifactDependency>  
            <SourceArtifact artifactID="a6f6798c-b3d8-4c9e-9973-1fe3a12f7ee6"/>  
            <TargetArtifact artifactID="3a6b9a5b-fcc9-4425-8842-65d3ee62e218"/>  
        </ArtifactDependency>  
    </ArtifactDependencies>  
</Package>
```

The **Package** node refers to the package on which the current package depends and contains the following information:

- **name**, the name of the application package
- **id**, the ID of the application package
- **version**, the version of the application package
- **build**, the build number of the application package

Within the **Package** node, the details about the dependencies between artifacts are depicted. The **Package** node contains one **ArtifactDependencies** node, which contains one or more **ArtifactDependency** nodes.

The **ArtifactDependency** node contains the following information:

- **SourceArtifact**, referring to the artifact within the main package
 - **artifactID**, containing the ID of the artifact within the main package. This ID corresponds to an artifact in the Delivers section.

- **TargetArtifact**, referring to the artifact within the application package specified in the **Packagenode**.
 - **artifactID**, containing the ID of the artifact within the application package specified in the **Package** node. This ID corresponds to an artifact in the **Delivers** section of the package dependency file of the referred application package.

Package dependency file schema

The file contents will comply with the following schema:

```

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.cordys.com/cws/applicationdependencies/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Project">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Delivers">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Artifact" type="ArtifactType"
maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Uses">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Package" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ArtifactDependencies">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element
name="ArtifactDependency" maxOccurs="unbounded">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element
name="SourceArtifact" type="ArtifactReferenceType"/>
                                <xs:element
name="TargetArtifact" type="ArtifactReferenceType"/>
                              </xs:sequence>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

```

        <xs:attribute type="xs:string" name="id"/>
        <xs:attribute type="xs:float" name="version"/>
        <xs:attribute type="xs:byte" name="build"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute type="xs:string" name="name"/>
<xs:attribute type="xs:string" name="projectId"/>
<xs:attribute type="xs:string" name="packageID"/>
<xs:attribute type="xs:string" name="packageName"/>
<xs:attribute type="xs:float" name="version"/>
<xs:attribute type="xs:byte" name="buildNumber"/>
</xs:complexType>
</xs:element>
<xs:complexType name="ArtifactType">
<xs:sequence>
    <xs:element name="ArtifactID" type="xs:string" maxOccurs="unbounded" />
    <xs:element name="Artifact" type="ArtifactType" maxOccurs="unbounded" />
</xs:sequence>
<xs:attribute type="xs:string" name="name"/>
<xs:attribute type="xs:string" name="qname"/>
<xs:attribute type="xs:string" name="path"/>
<xs:attribute type="xs:string" name="documentID"/>
</xs:complexType>
<xs:complexType name="ArtifactReferenceType">
    <xs:attribute type="xs:string" name="artifactID"/>
</xs:complexType>
</xs:schema>

```

Using the information from the package dependency files

To know the details of which artifacts use a particular artifact, you must bring the package dependency files of the relevant packages together. For instance, you can build up a custom repository in which both Delivers and Uses information is stored. The connecting link between your artifact being reused and the artifacts reusing your artifact is the artifact's run-time identifier. Given an artifact, say A, you must search for all target artifacts having a run-time identifier that is equal to the run-time identifier of A or any of child artifacts of A. From these target artifacts, you must take the corresponding source artifacts. Using the run-time identifiers of the source artifact, you can further retrieve the artifact details, especially the name of the source artifact, by searching for the matching artifact in the Delivers part of that package.

You can then provide the name of the artifact and request the dependency repository to return all the artifacts or child artifacts that are reusing it.

Qualified names versus artifact IDs

Impact analysis is bound to the life cycle of an artifact. If one decides to delete an artifact, that is the moment to determine the impact of that decision. If someone (possibly the same developer) happens to create a new artifact with exactly the same name, this artifact will be treated as being different from the artifact that was removed, because both have different artifact IDs.

During an artifact's lifecycle, the artifact ID does not change. This enables the system to trace an artifact, even when renaming it, moving it to another location in the same package, or even moving it to another package. The qualified name of an artifact is just one of its properties; it can change from time to time. In addition, the qualified name of an artifact must always be unique. There must never be two artifacts of the same type with the same qualified name in the same package, or in different packages that are to be deployed in the same context at the same time.

Changing the qualified name of an artifact is a deliberate action of the developer. The developer must then realize what that means for analyzing the impact of changes. After delivering a new version of the package, the dependency repository may contain multiple artifacts with the same name, but with different qualified names. It is then up to the developer to select the right artifact.

Setting properties of an application package

The properties of an Application Package will determine how you want to package the application content. You may either package application content to run as an application or package it so that it is open for minor modifications at runtime and then publish it for actual use. Depending upon the settings, the output file (Application Package) will differ.

To set the properties of an application package:

1. In **Workspace Documents**, right-click <**project**> and select **Packaging > Package Properties**.
The Application Package Properties window opens, displaying fields across two tabs.
2. Enter the following information on the **General** tab.

Package for	<p>There are two options:</p> <ul style="list-style-type: none"> ■ Runtime - The application package can only be deployed and used as is. ■ Staging - The resulting application package is loaded into the Staging Area. This allows you to align certain application artifacts to production environment specific configurations before publishing the application to the run time. In addition, this enables business operations managers to update business-critical application parameters instantly, that is without requiring engineering
-------------	---

	activities.
Create Model Package	<p>Caution: A package created for staging cannot be upgraded to later versions. Also, if such a package is unloaded, the changes that were made to it will be removed from all the organizations. Therefore, consider these points before choosing to create a staging application package.</p> <p>Only enabled if Package for = Runtime. If selected, the Packaging > Create Package context menu option and the CWS package subcommand will not only create an application package, but also a model package. The model package has the same file name as the application package, but has a different extension (.mpk instead of .cap - kind of acronym of Model Package).</p>
Package Owner	Owner of the Application Package.
Product Name	Name of the product or application that is packaged.
Version	<p>Version of the application.</p> <p>The version can be alphanumeric. While upgrading an ISVP, a comparison of ASCII alphanumeric values is performed. The comparison is not case-sensitive. Upgrade is possible only when the new version has a higher ASCII value compared to the old version. For example, upgrade is possible from version 1.0 to 2.0; while it is not possible to upgrade from the versions 9.0 to 10.0 and abc23.5 to ABC2.87.</p> <p>Note: Version is a free string. The following characters * ~ ` @ # % ^ & () { } [] \ / ? , > < + = " ' : ; are not allowed while naming the version for the package. For example, version can be A1.001.002 or B1.001 or C.002.004 and so on. Also, ensure that while specifying the version for the current package, it is always greater than the older version of the same package; this enables the current package to be considered for upgrade.</p>
Build Number	<p>A unique number representing the build of the application that is packaged.</p> <p>This number helps you to keep a track of successive builds of the application package and is useful while upgrading an Application Package from one build to another. The number given here should always be greater than the one given for the previously created Application Package. Build number allows numerals between 0 and 9 and can be delimited by a dot(.). For example: 123.457.</p> <p>Note: Build is a concatenation of two integers, separated by a dot; the second integer is optional. For example, the build number can be 500.12345 or 204.56795 or just 2356. Also, ensure that while</p>

	<p>specifying the build number for the current package, it is always greater than the older build number of the same package; this enables the current package to be considered for upgrade.</p>
Supported Deployment Spaces	<p>Spaces in which the package is allowed to be deployed. You can deploy packages either in the shared space or in organization spaces. If you select only the 'Shared' space, then this package can only be deployed in the shared space, making the functionality available for all organizations. Similarly, if you select only the 'Organization' space, then this package can only be deployed in organization spaces. If you select both the levels, then this package can be deployed in both the shared spaces and in organization spaces. For more information, see Managing Application Packages in the Organization Space in the <i>AppWorks Platform Administration Guide</i>.</p> <p>Note: You must select at least one of the supported deployment spaces. However, you can deploy the staging packages only in the shared space. As such, it is not possible to select the 'Organization' space when the property 'Package for' is set to 'Staging'.</p>
Package Name	<p>Name of the Application Package. It determines the uniqueness of the application and the value is checked during an upgrade. By default, the package name displays the values entered in Package Owner and Product. Package Name is the name by which the Application Package is stored in LDAP.</p> <p>Caution: Modifying the Package Name may interfere with the upgrade process of the previously installed application package.</p>
Package File Name	<p>A file name given to the Application Package. If you are creating the Application Package for use at run time, the file extension appears as .cap. If you are creating the application package for modifications at run time, the file extension appears as .staging.cap. By default, this displays the information as in the Package Owner and Product Name appended by the Version and Build Number, in that sequence.</p> <p>Note: The name cannot contain any special characters.</p>
Package Description	Description of the Application Package

3. Click the **License Agreements** tab to enter the license agreement information. This is a place where you can specify an End User License Agreement (EULA) along with the application content, if required. The license content is displayed for acceptance while loading the Application Package. The EULA could either be in .txt or .html format and must be uploaded to the project you are packaging.
4. Click  beside the End User Legal Agreement field. The Package Info dialog box opens, displaying all the TEXT and HTML files available in

the Workspace. The selected EULA appears in the output field as a hyperlink.

- To open the linked file, click the hyperlink.
- To remove the EULA, can click .

5. From the list of file names, select the license agreement file that is available, and click **OK**.

The Package Info dialog box closes and the file name of the license agreement along with the project name appears in the End User Legal Agreement field.

6. Add license information of any third party JAR used in the project, in the following manner:

- a. Click .

A row is added to the table.

- b. Under the JAR column, click  to browse and select the JAR available in the project.

The names of the JAR and the project which contains it appear next to .

- c. Under the JAR Description column, provide a description for the selected JAR file.

- d. Under the JAR Legal Text column, click  to browse and select the license agreement file that is available in the project.

The names of the license agreement file and the project which contains it appear next to .

- e. Under the JAR Legal Text Description column, provide a description for the selected JAR license agreement.

7. Click **OK**.

The properties for the Application Package are set.

To modify the properties:

- Modify both General and License Agreements tabs, and click **OK** to save the changes.

Setting the SVN working copy format of team development workspaces

AppWorks Platform supports multiple Subversion (SVN) Working Copy formats. By default, the latest supported format is used.

Before you begin:

- You must have the role of a System Administrator to perform this task.
- You must be able to access the configuration file on the server, where AppWorks Platform is installed, to modify the configuration settings and control the working copy version of the workspace.

To set a specific SVN Working Copy format:

1. Open the file <AppWorks Platform_installdir>\components\cws\config\cws.properties.
2. Add the following line to the file:

```
teamdevelopment.svn.workingcopy.version=<version>
```

You must replace <version> with the specific supported version of Subversion (1.7, 1.8, 1.9, 1.10). For example, if you are using Subversion 1.10, add
teamdevelopment.svn.workingcopy.version=1.10

3. Restart the CWS service container.

The SVN Working Copy format of new team development workspaces is now set to the given value.

Upgrading application development workspaces

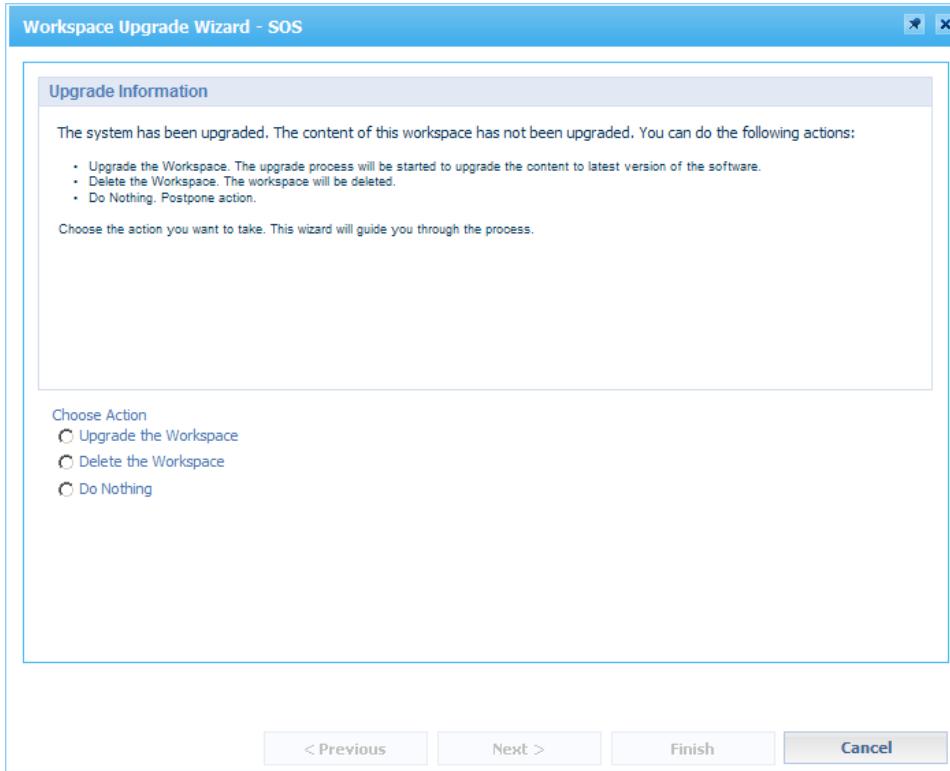
After upgrading a AppWorks Platform development environment, you may need to upgrade the individual application development workspaces one by one. This depends on whether changes have been introduced to the format of any of the document types. If a workspace must be upgraded, the workspace upgrade wizard is started the first time the development workspace is opened. Upgrading a workspace is required at most once after upgrading the AppWorks Platform development environment.

There is an important difference between upgrading standalone development workspaces and team development workspaces. This is because upgrading a team development workspace may require documents to be upgraded to the latest version. Upgrading a document requires the document to be modified. In team development scenarios, this is only possible when no locks exist in the SCM repository to which the workspace is connected.

The remainder of this topic explains the procedure to upgrade standalone development workspaces. Next, the topic discusses the same for team development workspaces.

Upgrading stand-alone development workspaces

When you open a standalone development workspace that must be upgraded, the following window opens.



The wizard provides three options:

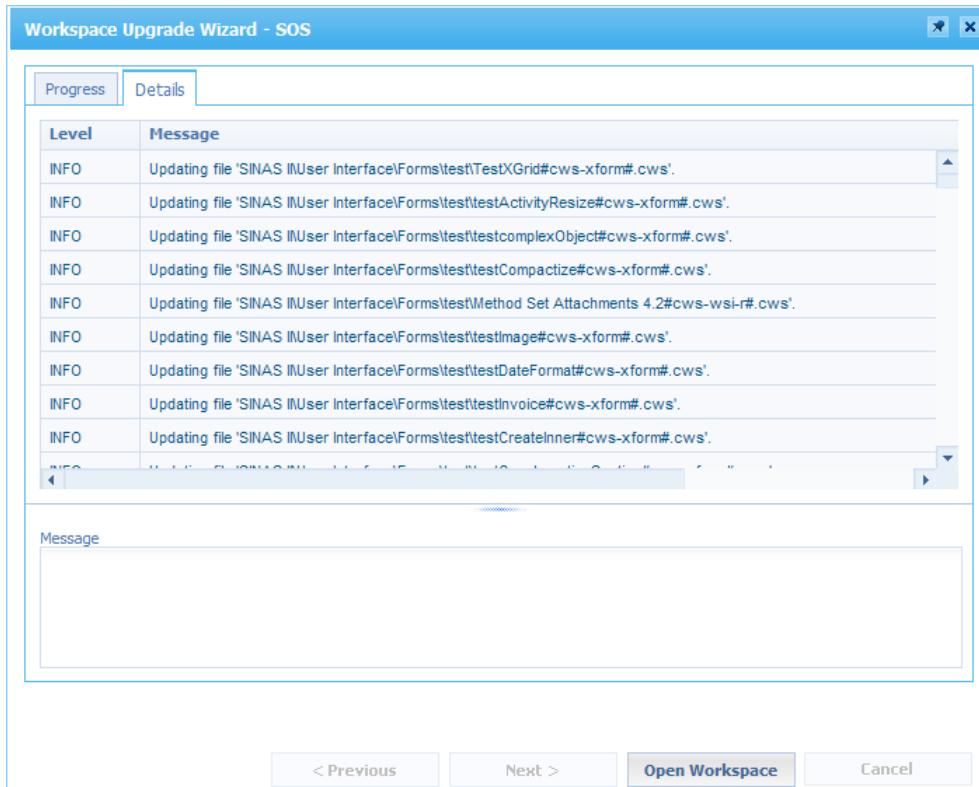
1. Upgrade the Workspace
2. Delete the Workspace
3. Do Nothing

In this section we explain each of the options in more detail.

To upgrade the Workspace:

1. Select **Upgrade the Workspace** and click **Next**.

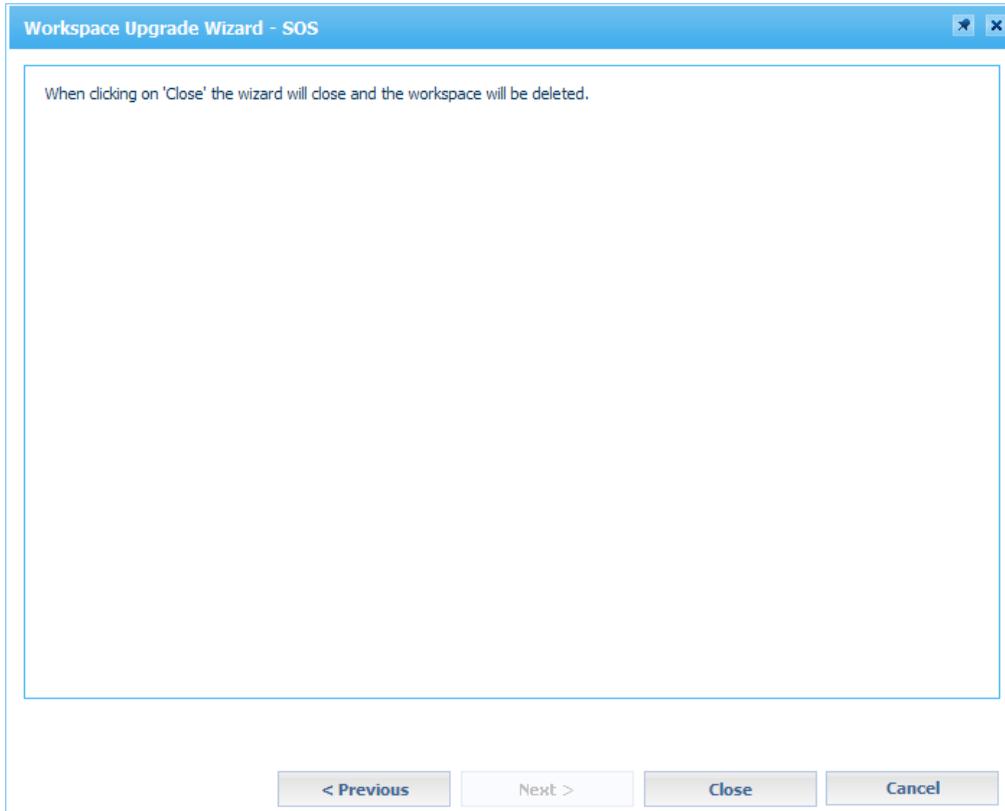
The process of upgrading the workspace starts. As long as the upgrade process continues, progress will be presented. When the workspace is upgraded successfully, the following window opens.



2. Click **Open Workspace**.
- The workspace opens.
3. Continue developing applications in this workspace.

To delete the workspace:

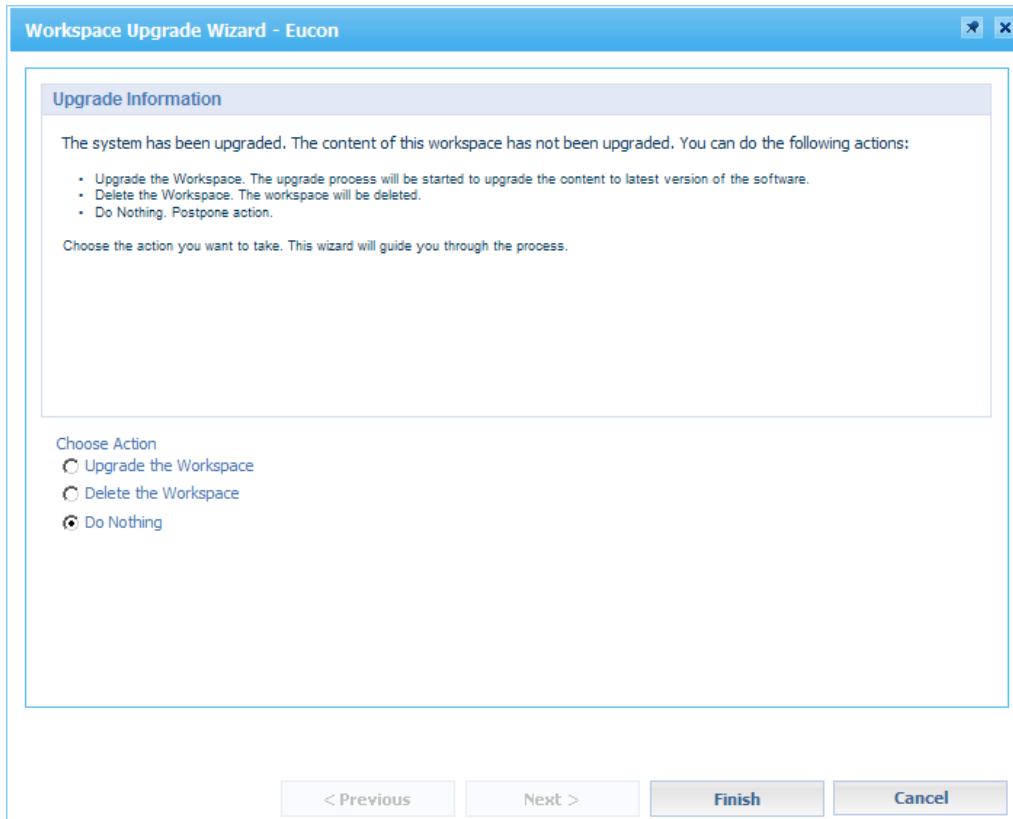
1. Select **Delete the Workspace**.
- The following window opens.



2. Click **Close** to delete the workspace.
- The wizard closes.

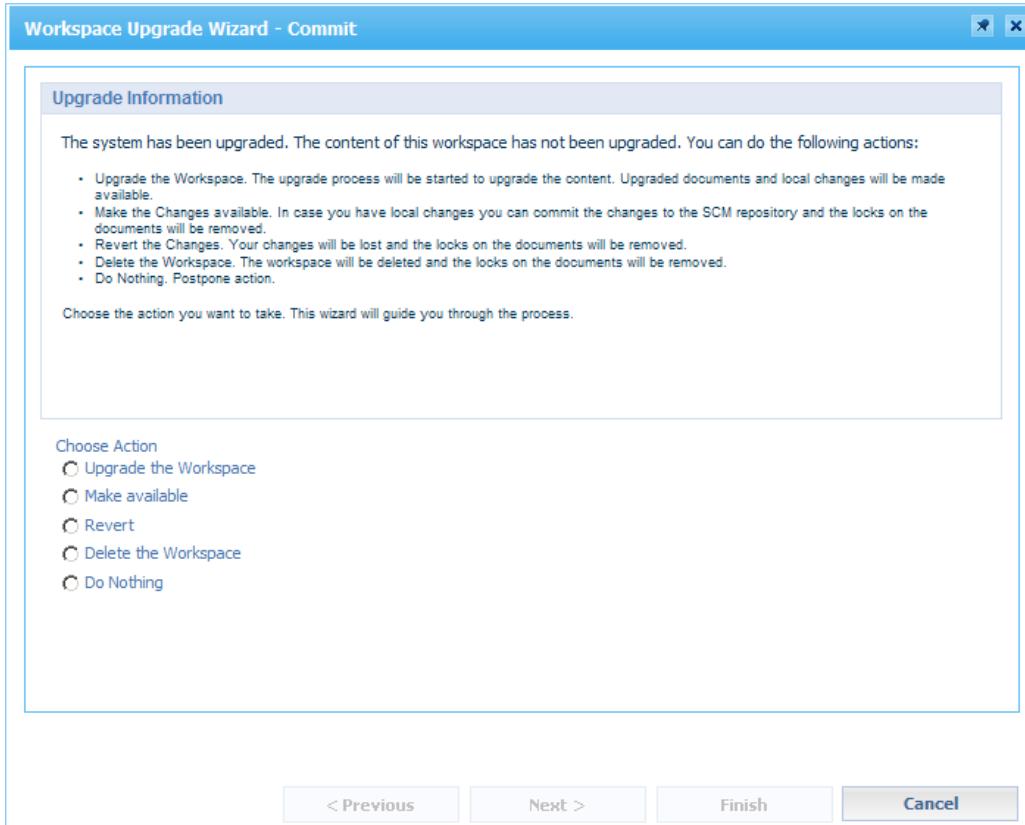
To do nothing:

1. Select **Do Nothing**.
The Finish button is enabled.
2. Click **Finish**.
The wizard closes. The Organize Workspaces window opens.



Upgrading team development workspaces

Upgrading team development workspaces is more complex. When you open a team development workspace that must be upgraded, the following window opens.



Dependent on the state of the workspace, the following options are provided:

1. Upgrade the Workspace
2. Make available
3. Revert
4. Delete the Workspace
5. Do Nothing

In this section, we explain each of the options in more detail.

To upgrade the workspace:

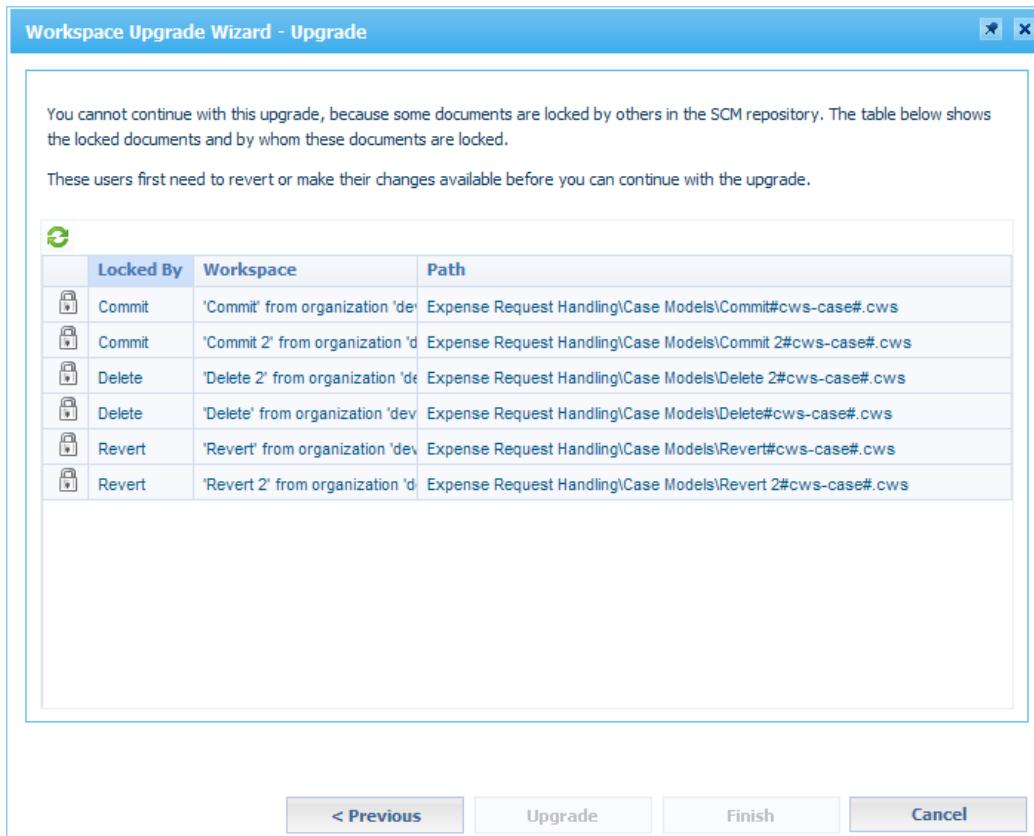
1. Select **Upgrade the Workspace** and click **Next**.

One of two results occurs:

- either the process of upgrading the workspace is started
- or an overview of locks in the repository is shown.

In the former case (if no locks are in the repository), the result is the same as if the workspace would have been a standalone workspace apart from the fact that after upgrading the workspace has completed successfully, all changes are automatically made available to others.

In the latter case (if others have locks in the repository), an overview is presented as follows.



From this overview, you know whom to contact to make sure that they release their locks. Then you can continue with upgrading your workspace. Others can release their locks either by making their changes available, reverting their local changes, or by deleting their workspace. In the remainder of this section, these options are explained.

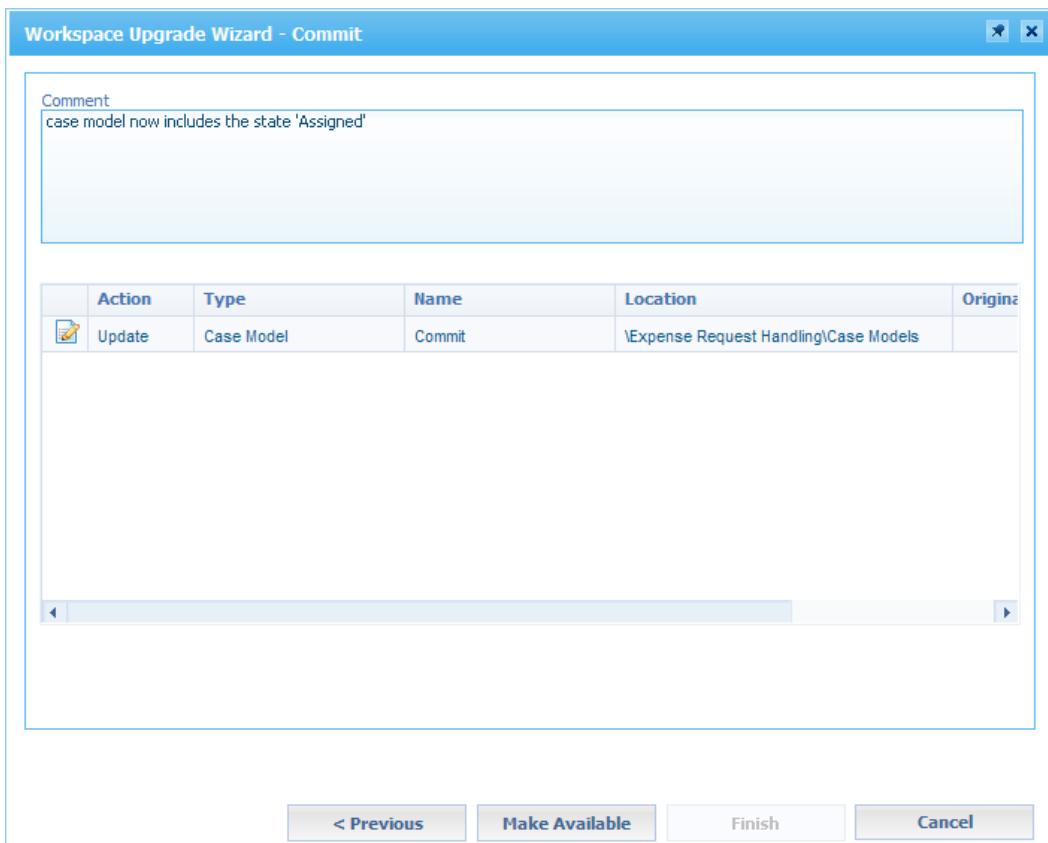
To update the lock overview:

1. Click . As long as there are still locks in the repository, those locks will be shown and **Upgrade** remains disabled.
2. When all locks have been released and refreshing the overview produces an empty list of locks, **Upgrade** is enabled after which you are able to upgrade the workspace. The remainder of the upgrade procedure is the same as a standalone workspace.

To make available:

The option **Make available** becomes enabled if there are local changes in the workspace for which locks reside in the repository.

1. Select **Make available** and click **Next**.



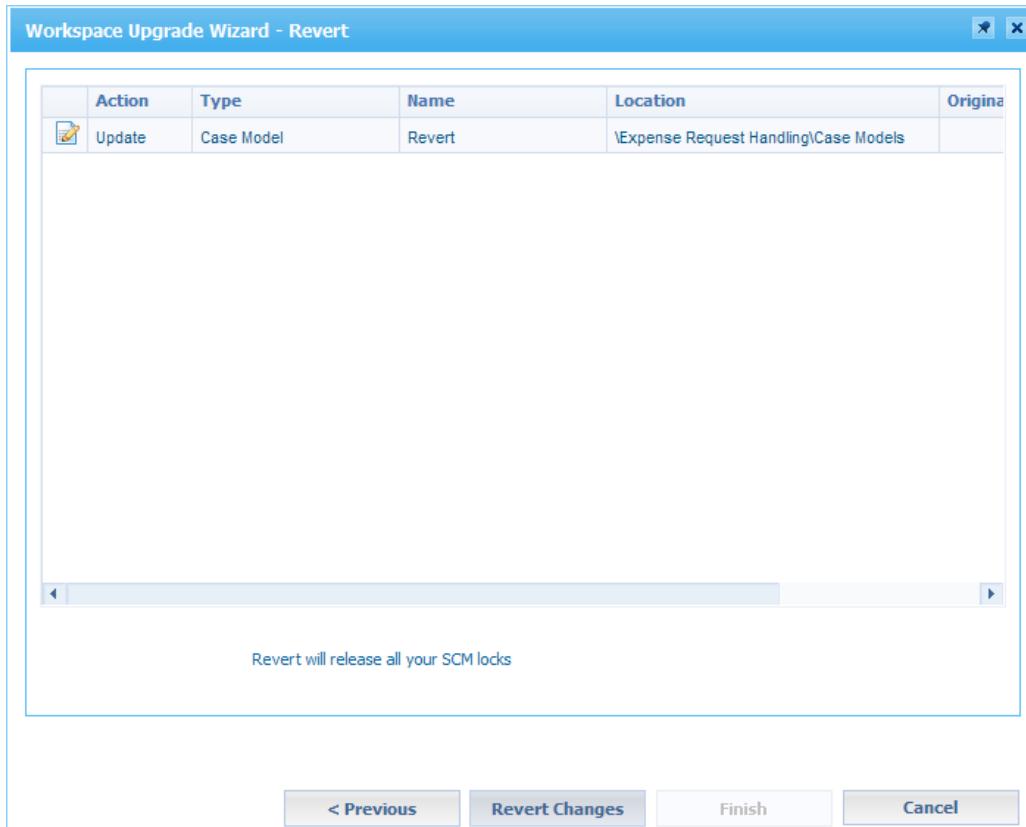
The window provides an overview of the various changes that will be made available. Furthermore, you can add a comment to describe the changes.

When you click **Make Available**, your changes are available to others. The upgrade wizard closes automatically and the Organize Workspaces window opens again.

To revert:

Similar to the previous option, **Revert** will only be enabled if there are local changes in the workspace for which locks reside in the repository.

1. Select **Revert** and click **Next**.



The window provides an overview of the various changes that will be reverted. When you click **Revert Changes**, the changes are reverted. The upgrade wizard closes automatically and the Organize Workspaces window opens again.

To delete the workspace:

The only difference between deleting a standalone workspace and a team development workspace using the workspace upgrade wizard is that in the latter case all locks of that workspace are released. See above for details about the main flow.

To do nothing:

There is no difference between the flow for deleting a standalone workspace and a team development workspace using the workspace upgrade wizard.

Viewing locks on the SCM repository

When developers connect to an SCM system and frequently modify its contents, they might want to know which documents are locked by themselves or which documents are locked by others. Based on this information, the developers can decide to work on those documents that are locked by them so that they can release the locks as soon as possible.

The lock overview feature of CWS provides the facility to view all locks that reside in the SCM repository to which the workspace is connected.

The lock overview enables the developers to see any of the following:

- all locks
- the developer's personal locks
- the locks of other developers who do not belong to their workspace

Before you begin:

- The workspace must be configured to a Source Control Management (SCM) system, for example, the SVN.

To view the locks on the connected SCM repository:

1. On the toolbar of Workspace Documents, click  (Show Locks On Repository Change History).
If there are no locks in the SCM repository, the system will inform you about this. If there are locks in the SCM repository, the Lock Overview window opens, by default displaying all the locks.
2. To show all locks that belong to you in this workspace, click **Personal Locks**.
3. To show all locks that do not belong to this workspace, click **Others' Locks**.

Caution: Be aware of the fact that locks belong to a working copy. As each team development workspace has its own working copy, and the same user is allowed to have multiple workspaces connected to the same SCM repository, the lock overview may include locks of the same AppWorks Platform user, but from different workspaces.

You have successfully viewed the locks that reside in the SCM repository to which the workspace is connected.

Collaborative Workspace Service connection interface

You will need to fill the following fields while adding a new Collaborative Workspace Service Container to the existing Collaborative Workspace Service Group.

The Collaborative Workspace Service Container Parameters are as follows.

Document Cache Size	The number of documents kept in cache per Workspace.
Network Configuration	To select a free port number to enable cache coherence manually, you can select the Manual check box and provide an unused free port number in the Port Number field below. By default, the Manual field is Not selected and the framework automatically assigns an unused free port number. You must restart the service container to apply the change.
Organization	Select the organization in which you want to create the Service Container.
Select Database	Select one of the existing database configurations that have been

Configuration	created. The details of the selected database configuration appear in various fields within the same window.
Advanced Properties	Click  to view the Advanced Properties and fill them.

Chapter 25

Application development (Reference)

The topics in this section provide supplementary, reference information about the various models involved in the creation of a Web application.

The topics are as follows:

- [XForms References](#)
- [WS-AppServer Reference](#)
- [Single Sign-On Reference](#)
- [Web Application Development](#)
- [Working With Alert System](#)
- [Working with Logging Framework](#)
- [Working With Problem Registry](#)

Operator precedence and associativity

Operator precedence describes the order in which the rule engine reads expressions. Operators are arranged in descending order of precedence, that is, from the highest to the lowest. More than one operator appearing in a single row has the same precedence.

The following table lists the operators in the order in which the rule engine reads them. The Associativity column specifies the direction in which each operator is read.

Operator	Description	Associativity
in	Logical in operator	Left to right
+ -	Unary plus/minus operators	Right to left
^	Power operator	Left to right
div Mod	Multiplication, division, modulus operators	Left to right
+ -	Addition/subtraction operators	Left to right

Operator	Description	Associativity
< <=	Relational less than/less than or equal to operators	No Associativity
> >=	Relational greater than/greater than or equal to operators	
= !=	Relational is equal to/is not equal to operators	
and	Logical and operator	Left to right
or	Logical or operator	Left to right

XForms references

This section contains topics that provide supplementary information about the functions and interfaces of AppWorks Platform XForms. It also includes topics that describe the basic concepts and architecture of XForms.

The section contains the following topics:

- [AppWorks Platform XForms overview](#)
- [AppWorks Platform XForms architecture](#)
- [AppWorks Platform AJAX toolkit architecture](#)
- [Model-View-Controller concept in XForms](#)
- [XForms Designer](#)
- [Script Editor](#)
- [XML Editor](#)
- [Interface Design Overview](#)
- [Model Properties dialog box](#)
- [Property Sheet of Controls, Regions, and App Palettes](#)
- [Predefined XForms](#)
- [Predefined and Specific Formats for Data Types](#)
- [Script Editor Toolbar](#)
- [Selectors in Cascading Style Sheets](#)
- [Keyboard Shortcuts \(XForms\)](#)
- [DOM Explorer](#)

AppWorks Platform XForms overview

AppWorks Platform XForms is a rapid application development (RAD) environment that facilitates building of applications based on Web services. Using the features and tools provided in AppWorks Platform, you can design and create interactive web applications that are also easy to maintain.

AppWorks Platform XForms helps you overcome the challenges faced while creating user interfaces using HTML. In applications created traditionally using HTML, it is difficult to maintain consistency across the entire application or product. This task becomes even more difficult if you need to add validations and custom scripts. Such applications are also difficult to maintain.

AppWorks Platform XForms helps to achieve the following:

- Minimal DHTML programming
- Re-use of existing schema definitions such as WSDLs and XSDs
- Integration with business workflow (business process modeling)
- Separation of data, logic, and presentation
- Inline translation during form generation
- Greater interactivity and consistency in interface
- Better performance due to caching of XForms
- Improved response and faster rendering due to decrease in the size of data to be retrieved from the database

AppWorks Platform XForms architecture

AppWorks Platform XForms comprises the following components that work together to generate Web pages:

- XForms Client: The client (browser) sends requests and displays responses. It also updates modified data, synchronizes Web pages, and provides navigation between them.
- XForms Gateway: Receives and responds to requests for Web pages (.caf files) from the client.
- XForms Engine: Executes XForm definitions, sets label translation and validation, generates HTML pages, and sends them to the Gateway.

When you open or refresh a Web page, which is an XForm with a .caf extension, the client routes the request to a Web server that identifies the requested file type.

The Web server forwards all requests for .caf files to the XForms Gateway for processing. The Gateway prepares a SOAP request and forwards it to an XForms Engine, which generates HTML content and sends it back to the Gateway. The Gateway, in turn, sends the HTML content to the Web server, and then to the client, which displays the HTML.

Depending on your requirements, the XForms engine also performs the following tasks while generating HTML:

- Translates the labels that are displayed on the Web page into the specified language.
- Sets validation rules for controls on a Web page using the XSD.
- Caches the generated HTML to ensure faster response.

- Generates SOAP requests based on WSDLs of the Web service operations used.
- Embeds ClickChoice definitions.

Overview of AppWorks Platform AJAX toolkit

AppWorks Platform Ajax Toolkit is a collection of easy-to-use tools and components that facilitate Web Application Development. AJAX Toolkit provides a single-source application generation solution that delivers HTML content.

AppWorks Platform Web Gateway receives request, rendered as an HTML file, from the client and sends it to the Web server. The server accesses the data available in the database and sends it back to the front-end in HTML format. Processing the data and sending it back to the front-end is performed on the server side, thus increasing the server load.

The disadvantages of the typical server side processing are:

- The load on the server increases.
- The client is used as a dumb machine; the full potential of the client machine is not utilized.

AppWorks Platform Ajax Toolkit can create a client-side data processing environment. The AJAX Toolkit passes the client request to the gateway in the Web server. The server obtains the relevant data from the database and passes the result back to the client in an XML format. The client renders the XML to HTML.

The advantages of this processing are:

- The load on the server decreases, resulting in better scalability.
- Data manipulation is done on the client side.

AJAX Toolkit contains a set of rich web components built using JavaScript. These Web components are simple, lightweight components that encapsulate specific functionality (behavior) on a page. When applied to a standard HTML element on a page, a web component enhances that element's default behavior.

Web components enable:

- extending the functionality of existing tags
- extending HTML by creating custom tags
- encapsulation of code
- componentization of code

Within AppWorks Platform Ajax Toolkit architecture, the AJAX toolkit components (library) communicate with the web server using Web Gateway.

Communication between the browser and the gateway is performed by a behavior called the BusDataIsland, which exposes a set of properties, methods, and events to enable the developer to obtain data from the transaction object and manipulate the transaction object. It can also change the organizational context or receiver Service Group at run time.

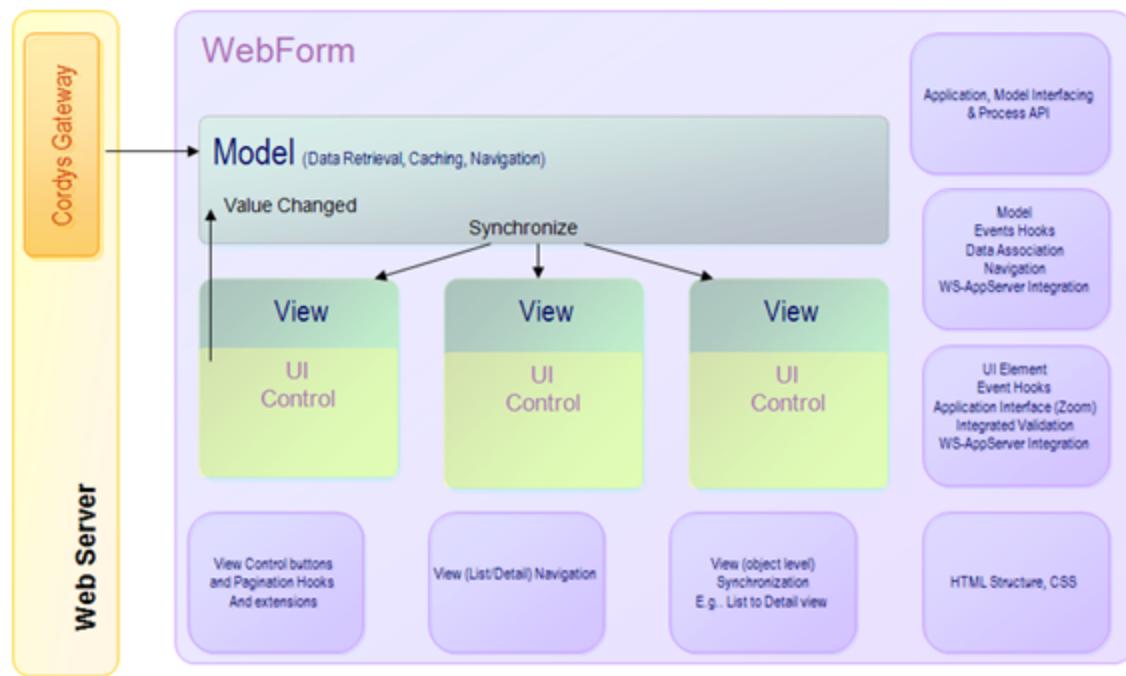
AppWorks Platform XForms uses the AppWorks Platform Ajax Toolkit framework to enable you to generate Web applications. For a description of how the AppWorks Platform Ajax Toolkit framework works while loading the Welcome page, see [Flow of Control while Loading the Welcome page](#).

Model-View-Controller concept in XForms

AppWorks Platform XForms is based on the Model-View-Controller (MVC) architectural design pattern that divides an application into components that control its behavior and presentation.

In most implementations, the Model, View, and Controller components of an application are divided among the web server or application server and the browser. In XForms, however, all associated models, views, and controllers are available on the browser side.

The following diagram represents the MVC pattern in the XForms architecture.



Each XForm can be associated with various models, views, and controls that define its behavior, appearance, and related Web service operations.

A Model:

- Retrieves and caches data in the browser.
- Can be associated with many different Views in an application. This makes it possible to use a common set of data in multiple pages of the application.
- Enables navigation between various Views of the application.
- Stores the transaction states of data, such as insert, delete, and update.
- Passes data and information about the transaction states to the backend.

- Exposes APIs. For example, to know the current transaction state, or for notifying the user of a SOAP fault.
- Notifies controls in a View that are associated with Model, about changes in data.
- Synchronizes controls, in cases where a control displays information about a record selected in the other control.

A View:

- Is created when a UI control is bound to a Model.
- Exposes and allows modification of a part of data from the Model.
- Enables navigation through a data set.
- Enables pagination. Notifies the Model for more information when navigating through a set of records.
- Has APIs that are bound to controls. The APIs are available through controls for ease of use.

A Controller:

- Tracks the records available in the View, and fetches a new set of records, when required.
- Enables a control, associated with a Model, to inherit APIs of the View. This enables you to write code for the controls rather than the View.
- Enables the use of data from a Model in different Views. Also, notifies the Model for data changes in a View, and enables the synchronization of all Views.
- Provides UI-level APIs for data binding and pagination.
- Controls can be registered from other form; for example, table in one form and a groupbox in another.

All requests to retrieve data from the backend channel through the AppWorks Platform Gateway, which sends the request to the server.

Consider an XForm that displays data from a Web service. In this case,

- The View is the user interface of an XForm.
- Data received from a Web service is stored in the Model and displayed in the user interface (View).

All activities performed on an XForm are tracked using events and Web service operations. Events and Web service operations are also used to synchronize Models and Views for changes and data updates made during any activity. When you work with a control in an XForm and provide an input, relevant events are fired.

The Model, View, and Controller of the XForm interact as follows:

1. The Controller captures the events and imitates the Model regarding the changes.
2. The Model synchronizes data with the backend, and receives the updated data from the Web service.

3. The Model then notifies the View of the data update, which displays it in the required format.
4. In case you added a new record through the View, the Model also synchronizes the updated record with the Web service.

This exhibits the separation of the data (Model) from the presentation (View) of the XForm.

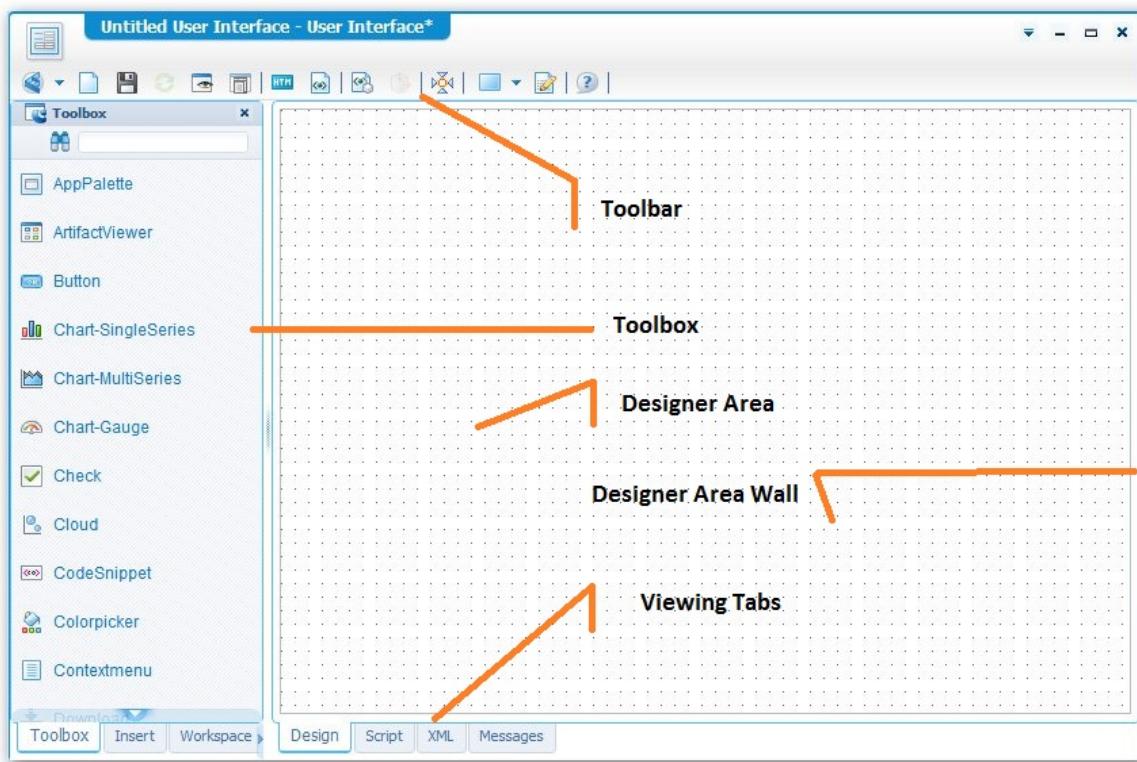
XForms designer

The XForms Designer offers various tools and features to design, create, and edit user interfaces for web applications. It offers an intuitive interface that enables the creation of interactive XForms with minimal programming.

The XForms designer comprises a toolbar and the following viewing tabs.

Toolbox	Displays the controls that are available for creating an XForm. You can use the Toolbox tab to add controls to the XForm.
Insert	Displays the documents that can be used with or added to an XForm.
Workspace	Displays the workspace that you are currently working in. For information about the Insert and Workspace tabs, see Inserting Related Documents and Exploring Workspace Contents .
Design	Displays the Designer Area, which is a grid where you design an XForm. It offers various functions to align controls (UI elements) and format the XForm.
Script	Displays the Script Editor where you can write custom code to modify XForm behavior.
XML	Displays the XML Editor where you can work with XML definitions and XML data islands of the XForm.
Messages	Insert button to add a message. To specify a different text identifier, select Create new and specify the text identifier in the dialog box that displays. The messages can be used in a script. See the AppWorks Platform API Guide .

1. In the Toolbar, Insert, and Workspace tabs, click  to close them.
2. To open the tabs, click  in the title bar of the XForms designer and select the appropriate tab.



The Toolbar contains tools, such as Save, Preview, Import Data, Manage Models, and Show Anchor Bars, to work with the XForm.

Using XForms designer, you can design XForms, and define the properties and run-time behavior of each control in the XForm and of the XForm as a whole. For information about the features that can be used to design XForms in the XForms Designer, see [Working with XForms](#).

Script editor

AppWorks Platform XForms provides a rapid application developer (RAD) tool (the XForms Designer) to help quickly develop web applications. The controls provided in the XForms Designer are associated with specific functionality using which you can create interactive XForms. However, if you need to use additional functionality, you can write your own code using the Script Editor and achieve the desired behavior for your XForm. Using the Script Editor, you can create and modify XForms objects and data models in an XForm.

The Script Editor simplifies the process of writing the code for an XForm.

The script editor provides:

- Intellisense, which is a feature that prompts you with properties and Web service operations applicable to an object when you edit code. Using the Intellisense feature, you can insert the appropriate property or Web service operation in the script. The Intellisense feature also enables easy referencing of object models. You can use it to insert language elements directly into the code.

- Undo and Redo features that enable you to reverse the changes made to code. You can press CTRL+Z to perform the undo action and press CTRL+Y to perform the redo action.
- Features such as Color Coding, Syntax Validation, Script reloading, and Find. For detailed information about these features available with the Script Editor, see [Script Editor Toolbar](#).

Caution: AppWorks Platform does not recommend using the Script Editor to work with DHTML objects as it may lead to backward compatibility issues.

XML editor

The XML Editor enables you to embed XML definitions or XML data islands in an XForm. These definitions or data islands refer to information stored as XML in the XForm. You can use this information in an application by binding the XML definitions with the HTML elements of the XForm. You can also view and customize the SOAP requests generated in case of external Web services in the XML Editor. Click the XML tab in the XForms Designer to access the XML Editor.

AppWorks Platform XForms automatically assigns a unique ID to each XML definition that you add. If required, you can manually change this ID.

To provide an XML data island in the XML Editor, use the following XML document syntax:

```
<xml> <xml id="sampleXML"> <sample> <id>sample</id> <description>some example XML</description> </sample> </xml> </xml>
```

Note that each XML data island contained in the above XML has a unique ID and all the XML data islands are wrapped in a single, parent<xml></xml>tag.

By default, a dummy XML template is added to the editor as shown below. While saving the XForm, the template will be deleted if there are no changes in it. Otherwise, the XForm is saved with the updated template.

```
<xml>
  <xml id="xmlID" xmlns="" />
</xml>
```

To access any XML data island such as in the XML editor, you must use the<XMLIslandid>.XMLDocumentformat in the script. For example, use sampleXML.XMLDocument.documentElement, instead of sampleXML.documentElement.

The Application Definitions used to load any XForm is also defined in the XML editor here.

XML Editor toolbar

The following table lists the buttons available in the XML Editor toolbar.

Button	Button Name	Description
	Validate and Format XML	Validates and formats the syntax of the XML. A notification message displays in case of errors.
	Reload XML Without saving changes	Displays original XML without saving any changes made in the XML Editor.

Interface design overview

AppWorks Platform provides you with various features to create interactive XForms. These features are grouped as:

Containers	Invisible placeholders that hold layouts and controls. Containers help define properties such as size and context-menu options during design time.
Layouts	Visual organization of controls in an XForm. You can also specify layouts for composite controls such as tabs, boxes, groups, and splitters. Layouts can be vertical, horizontal, or free.
Controls	Interface entities used to perform a task. A control may contain other controls or may be associated with elements.
Elements	Interface entities such as zoom button and element bar that are used with controls.
Positions	Visual organization of controls in an XForm, independent of the parent control's layout. The position of a control overwrites the layout applicable to it. Positions can be vertical, horizontal, or free.

AppWorks Platform XForms uses [style sheets](#) to define the appearance of the XForms created.

Layouts

Layouts help you organize the interface of an XForm. You can apply layouts to an XForm as a whole, or to grouping controls in an XForm. Layouts do not apply to or affect the position of a toolbar in an XForm.

AppWorks Platform XForms provides the following layout types.

Vertical layout

You can use vertical layouts to stack controls one below the other. Controls in vertical layouts occupy the maximum available width.

A vertical layout has the following characteristics.

- Controls with default widths automatically resize on adjusting the window at run time.
- Placeholders indicate the locations where a control is to be added.

- Tab order is from left to right (for tables and nested controls) and top to bottom for controls with default widths.
- It is possible to resize the width of controls during design time, but such controls do not respond on resizing the window at run time.
- Resizing of control height is possible only for Textarea, List, Table, Button, Groupbox, Group, and Frame controls.

Horizontal layout

You can use horizontal layouts to sequentially arrange controls in a row. By default, only two controls can be arranged in a row, each control occupying a part of the available width. If you add more than two controls, they are automatically shifted to the next row. However, you can manually resize the controls to arrange more controls in a row.

A horizontal layout has the following characteristics.

- Controls with default widths automatically resize on adjusting the window at run time.
- Tab order is from left to right (for tables and nested controls) and top to bottom for controls with default widths.
- Placeholders indicate the locations at which to add a control.
- It is possible to resize the width of controls during design time. However, controls with custom width do not respond to resizing of the window at run time.
- Resizing of control height is possible only for Textarea, List, Table, Button, Groupbox, Group, and Frame controls.

Free layout

You can use free layouts when you need to position controls randomly in an XForm.

A free layout has the following characteristics.

- Control resizing is supported for all controls. However, for Input, Password, Output, Select, and Check only the width can be modified.
- Tab index is not supported in its default format.
- The anchoring feature is available for all controls.
- Controls must be anchored to respond to resizing of the window at run time

Controls

Controls are the interface entities that are used to perform specific actions in an XForm. The XForms Toolbox tab displays the controls available in XForms.

To add a control to an XForm:

1. Drag it from the Toolbox tab to the XForm.
A placeholder appears to help you position controls. Alternately, double-click the control to add it to the XForm.
2. Define the run-time behavior of a control by setting its properties in its property sheet.

See the *AppWorks Platform API Guide* for more information about the following controls.

Control	Constituent Interface Elements	Additional Interface Elements
AppPalette	AppPalette, Region	-
Artifact Viewer	Container	-
Button	Control box (container), Button(s)	
Chart	Container	-
Check	Label, Check box	
Cloud	Container	-
CodeSnippet	Input area for code text	Lookup button
Colorpicker	Colorpicker	-
Contextmenu	Context menu	-
Download	Invisible container	-
Frame	HTML Iframe	
GoogleGadget	Container	-
GoogleMap	Google map	-
Group	Invisible container	
Groupbox	Header, Container	Element bar
Image	Control box (container), Image(s)	
Input	Label, Input field	Lookup button
List	Label, List box	Lookup button
Output	Label, Output field	Lookup button
Password	Label, Password field	
Preferences	Preferences; invisible at run time	-
Progressbar	Progress bar	-
Radio	Control box (container), Radio button, Labels	
RichText	Input area for rich text	-
Select	Label, Select box	Lookup button
Statusbar	Status bar, Input area	-
TabPage	Tab group (container), Tab page(s)	
Table	Table with rows, Element bar	

Control	Constituent Interface Elements	Additional Interface Elements
Textarea	Label, Input area	Lookup button
Tree	Tree list	-
Toolbar	Toolbar, Save button	Toolbar item
Upload	Invisible container	-
XGrid	Table with rows, Element bar	-
XMLUtil	XMLUtil; invisible at run time	-

The controls mentioned here are provided in AppWorks Platform XForms by default. To create your own composite controls, see [Working with Composite Controls](#).

Note

- The Input, Password, Output, Textarea, Check, Radio, Select, List, Text, and Frame controls are also referred to as Primitive controls as they are basic data-entry controls that comprise a label and an input area to enter data.
- For information about the additional interface elements, see [Elements](#).

APIs for controls

AppWorks Platform XForms exposes APIs for customizing and using controls in an XForm. Some APIs are common for all controls, while some are applicable to specific controls or types of controls such as the API to control pagination bar for grouping controls.

Note: Ensure that you create the required UI using the XForms Designer only. Do not manipulate the DOM object tree by using DHTML in run time, as HTML elements added during run time do not have XForms API associated with them. This may lead to unpredictable behavior of the XForm. Moreover, the future releases of AppWorks Platform XForms may not support the changes made.

Elements

Elements are interface entities that provide specific functionality to controls. Elements do not have any significance on their own, except when used with controls.

The following table lists the elements available in AppWorks Platform XForms.

Element	Namespace	Extends from	Description
Choice box	cas.xforms.designerlibrary.controls.XFormsChoicebox	XFormsGroupControl	Contains controls with mutually exclusive options such as Radio control.
Control bar	cas.xforms.designerlibrary.controls.XFormsControlbar	XFormsGroupControl	Contains buttons applicable to the grouping control.
Control box	cas.xforms.designerlibrary.controls.XFormsControlbox	XFormsGroupControl	Automatically added with the Button, Image, and Radio

Element	Namespace	Extends from	Description
			controls, and can contain multiple controls. It is invisible at run time.
Element bar	cas.xforms.designerlibrary.controls.XFormsElementbar	XFormsGroupControl	Contains Control bar and Pagination bar, and is applicable to grouping controls.
Element bar Item	cas.xforms.designerlibrary.controls.XFormsElementBarItem	XFormsControl	Button on the element bar.
Lookup button	-	-	Refers to the Zoom and Calendar Control buttons. When clicked, the zoom button opens a dialog box to display detailed information, and the Calendar Control button displays a calendar to specify a date. The Calendar Control is applicable only to controls of the Date data type.
Pagination bar	cas.xforms.designerlibrary.controls.XFormsPaginationbar	XFormsGroupControl	Contains buttons to display and navigate through records, and is applicable to grouping controls.
Region	cas.xforms.designerlibrary.controls.XFormsRegion	XFormsTabGroup	Container for AppPalettes.
Region Set	cas.xforms.designerlibrary.controls.XFormsRegionSet	XFormsGroupControl	Container for regions.
Splitter	cas.xforms.designerlibrary.controls.XFormsSplitter	XFormsGroupControl	Splits grouping controls and AppPalettes into multiple parts.
Splitter group	cas.xforms.designerlibrary.controls.XFormsSplitterGroup	XFormsGroupControl	Container for Splitter pages.
Splitter page	cas.xforms.designerlibrary.controls.XFormsSplitterPage	XFormsGroupControl	Split area that contains controls.
Tab caption	cas.xforms.designerlibrary.controls.XFormsTabCaption	XFormsControl	Tab header of a Tab Page.
Tab Group	cas.xforms.designerlibrary.controls.XFormsTabGroup	XFormsGroupControl	Automatically added with a Tab Page, and can contain multiple Tab Pages.
Toolbar	-	-	Contains buttons applicable to the XForm.
Toolbar Item	cas.xforms.designerlibrary.controls.XFormsToolbarItem	XFormsControl	Button on a toolbar.
Toolbar Item Save	cas.xforms.designerlibrary.controls.XFormsToolbarItemSave	XFormsControl XFormsToolbarItem	Save button on a toolbar.
Toolbar Separator	cas.xforms.designerlibrary.controls.XFormsToolbarSeparator	XFormsControl	Separator bar that can be used to group toolbar items

Positions

When added to an XForm or a parent control, controls are automatically positioned according to the layout of the parent control. You can use the Position feature to specify alternate positions for individual controls and arrange them independent of the layout of the parent control. The position that you specify for a control overwrites its layout. The default position of a control is the same as the layout of the parent control.

AppWorks Platform XForms provides the following control positions.

Vertical position

You can use the vertical position to arrange a control similar to controls in a vertical layout. Controls with vertical positions occupy the maximum available width of the parent control.

The vertical position has the following characteristics:

- Controls with default widths automatically resize on adjusting the window at run time.
- Placeholders indicate the locations at which to add a control.
- It is possible to resize the width of controls during design time. However, controls with custom width do not respond to resizing of the window at run time.
- Resizing of control height is possible only for Textarea, List, Table, Button, Groupbox, Group, and Frame controls.

Horizontal position

You can use the horizontal position to sequentially arrange controls in a row, similar to controls in a horizontal layout. Only two horizontally positioned controls can be arranged in a row, each control occupying a part of the available width.

The horizontal position has the following characteristics:

- Controls with default widths automatically resize on adjusting the window at run time.
- Placeholders indicate the locations at which to add a control.
- It is possible to resize the width of controls during design time. However, controls with custom width do not respond to resizing of the window at run time.
- Resizing of control height is possible only for Textarea, List, Table, Button, Groupbox, Group, and Frame controls.
- The positioning attributes of the cascading style sheets (CSS) are used to position controls. This permits overlap of controls. The control on top or the one that was clicked last will be visible in case of overlapping of controls in a free layout.

Free position

You can use the free position-to-position controls randomly in an XForm.

The free position has the following characteristics:

- Control resizing is supported for all controls. However, for Input, Password, Output, Select, and Check controls only the width can be modified.
- The anchoring feature is available for controls.
- Controls must be anchored to enable them to respond to resizing of the window at run time.

AppPalette

AppPalette is a sub-application that is loaded inside another application. An appPalette can be loaded in an application using the `loadAppPalette()` or `loadAppPaletteContent()` methods. An appPalette is an application by itself and, when loaded using the method, it has the same life cycle as an application. Loaded appPalettes can be accessed from the `appPalettes` collection of the application object.

The properties defined for appPalettes are as follows.

Properties	Description
automatic	<p>Optional. Denotes whether or not the content of an App Palette is loaded, when a Web page containing the App Palette is loaded for the first time. Possible values are:</p> <p>true: Default. App Palette content is loaded when the Web page is loaded for the first time.</p> <p>false: App Palette content is not loaded initially. It is loaded only when the user clicks the App Palette.</p>
dockable	<p>Optional. Denotes whether or not an appPalette is docked to a region when dragged to it. Possible values are:</p> <p>true: Default. App Palette can be docked to regions.</p> <p>false: App Palette cannot be docked to regions.</p>
registerViewAs	<p>Optional. Denotes whether an appPalette can be registered in the View Manager (menu in the toolbar of an App Palette) or not. By default, all appPalettes that can be closed are registered. Possible values are:</p> <p>self: appPalette is registered as direct item in options menu.</p> <p>none: appPalette is not registered in view manager.</p> <p>If any other value is assigned, it is considered as a group name, that is multiple appPalettes with the same name (other than self or none) are considered part of a single group.</p>

The methods defined for appPalettes are as follows:

Method	Description
<code>undock()</code>	Undocks the appPalette object to the floating mode. In the floating mode, the appPalette can appear anywhere on the application and must have a title

Method	Description
	bar for identification.
setTitle (sCaption, sDescription)	Sets the title of the appPalette. A title is a combination of caption and description (caption-description).
setCaption()	Sets the caption of the appPalette.
dock()	Positions the appPalette in the docked mode in an Application. In the docked mode, the appPalette is attached to a docking area along the four borders of the region. If no region is specified, the appPalette will be in the floating mode.
hide()	Hides the appPalette if it is visible.
getTitle()	Gets the current title of the appPalette. The title is a combination of <caption> and <description> if they are different, or just <caption> if they are the same.
getCaption()	Gets the current caption of the appPalette defined in the AppPalette Definition.
show()	Displays the appPalette if it is hidden.
close()	Closes the appPalette.

For an appPalette other than static HTML appPalette, the isAppPaletteClosable() method can be implemented and must return boolean when called. The isAppPaletteClosable() method is called by the host application when it is being closed. If any of the appPalettes in the application return false, then the application and the appPalettes will not be closed.

AppPalette definition

```
<script id="testAppPaletteUIApplication" type="cordys/xml">
<Application display="visible" focus="true" registerViewAs="Group Name">
<id>testAppPaletteUI</id>
<url>/cordys/wcp/test/library/ui/testcollapsible.htm</url>
<description>App Palette UI</description>
<caption>App Palette</caption>
<region docked="false">rightRegion</region>
</Application>
</script>
```

If the inPreferences attribute is set to true in the Application tag of an AppPalette definition, then on closing the application, the state of the appPalette is stored as preferences and is retained the next time you open the application.

appPalettes Property

Gets the collection of AppPalettes loaded in an application. Each AppPalette can be iterated either by using the for-each loop or by directly setting the index of the appPaletteId. You can also apply other methods, mentioned for appPaletteUI, to the object.

Syntax

```
\application.appPalettes
```

Return Value

Returns the AppPalette containers collection.

Example

```
var appPalette = application.appPalettes[appPaletteId] ;
appPalette.show() ;
appPalette.hide() ;
```

Applies To

application

Style sheets

AppWorks Platform XForms uses style sheets to ensure consistency in presenting information across the pages of an application.

Style sheets in AppWorks Platform XForms

The current version of XForms comprises multiple style sheets that define the appearance of pages.

Style Sheets in Current XForms	Description
xdefault.css	Distinguishes and defines the position and appearance of interface elements; contains additional properties for complete control over all aspects of interface design
default.css	Provides support for rendering pages created by applications other than AppWorks Platform XForms

Cascade order of style sheets

The style sheets are applied to an XForm in the following order.

Style sheets in order of application	Description
default.css	First style sheet that is applied to an XForm
xdefault.css	Applied over default.css; overrides changes in case of conflicts
<custom style sheet>.css	Any custom style sheet; overrides changes made by default.css and xdefault.css

Cascading style sheets

AppWorks Platform XForms use cascading style sheets to define interfaces. Using style sheets enables you to separate presentation from the content in an XForm. You can use a single style sheet for multiple XForms, and use multiple style sheets for a single XForm.

Cascading style sheets are easy to maintain and use. On editing a CSS, the changes immediately reflect it on all the pages associated with the CSS.

In addition to the style sheets available in AppWorks Platform XForms, you can define and use custom style sheets. To create a custom style sheet, use the available `default.css` file as a template, and edit it as required. Ensure that you maintain the cascade order while editing it.

Cascading style sheets apply to an XForm in the following order:

- `default.css`
- `xdefault.css`
- `<custom style sheet>.css`

The custom style sheet overrides the styles applied by `default.css` and `xdefault.css`. Style sheets in AppWorks Platform applications must use a specified set of selectors to define the interface.

Types of selectors

The selectors are distributed among `default.css` and `xdefault.css`.

The selectors used in style sheets are classified as follows.

Type of Selector	Description
Type	Comprises selectors for rendering basic HTML elements
Presentational	Comprises selectors for defining decorative elements like borders, background and foreground colors, fonts, and images. Mostly found in <code>default.css</code> .
Structural	Comprises selectors for interface elements, but does not define appearance. Mostly found in template style sheets.
Basic UI	Comprises selectors for creating basic interface elements in XForms. Found in <code>xdefault.css</code> , and some template style sheets

Model properties dialog box

You can specify the following in the Model Properties dialog box.

Model ID field	Type a unique ID for the model. If left empty, it is automatically populated when you define a method in the Web Service for 'Get' Operation field of the Dataset tab. Restriction: You cannot assign reserved keywords of JavaScript as the model ID.
Automatic check box	Select to set the model as automatic so that it sends a request to the backend when the XForm is loaded.
Non-transactional check box	Select to create a non-transactional model. The Prompt to Save check box and the Associations tab are automatically hidden from view.
WS-AppServer Integration check box	Select to enable integration with the WS-AppServer. The WS-AppServer tab appears. The Single Transaction functionality is not supported for WS-AppServer models. On selecting a WS-AppServer model for association, the corresponding Single Transaction check box in the Associations tab is disabled. In case of associated models, integrating a model with the WS-AppServer disables single transaction for its associated models.
Instance Schema check box	Select to display the Instance Schema tab. Here, you can define the instance schema for XForms that are not associated with a WSDL. The absence of WSDL makes it difficult to map such XForms for input and output. You can use the Instance Schema tab to add schema to such methods, making them visible in the message map for mapping. While generating an HTML definition, AppWorks Platform XForms interprets the schema for a business object and combines any validation defined for it. When you enter a value in the XForm, at run time, the value is validated and an alert displays if the validation fails. In the Schema field: <ul style="list-style-type: none">■ Click  to select an existing schema fragment from the Select Schema Fragment dialog box that appears. The Business Object field in the Dataset tab is automatically updated with the business object of the selected schema fragment.■ Click  to open the <Schema name> - Schema Fragment window and edit the schema fragment.■ Click  to delete the schema fragment.

	Note
	<ul style="list-style-type: none"> ■ For information about creating schema fragments, see Creating Schema Fragments. ■ For information about creating business process models, see Creating a Business Process Model. ■ For legacy XForms created using Cordys BOP-4 or earlier versions, the Instance Schema tab displays the schema. You can edit the schema, if required.
Prompt to Save check box	Select to enable the display of alerts when you close an XForm without saving it.
Dataset tab	Use the options available to specify details regarding the business objects and methods.
Events tab	Use the options available to specify the methods to be executed for receiving, requesting, and synchronizing data with the backend.
Data Events tab	Use the options available to specify the methods to be executed for tasks involving business objects.
Namespaces tab	Use the options available to specify the namespace for the model. You can add or delete the prefixes and namespaces, as required. The delete option is not available for the default namespaces of a Web service. The namespaces declared on a Web service display in the disabled mode and cannot be modified or deleted.
Associations tab	Use the options available to specify settings for associating models.
Gateway tab	Use the options available to specify settings for the gateway.
WS-AppServer tab	Use the options available to specify settings for XForms integrated with the WS-AppServer.
Model Messages tab	<ul style="list-style-type: none"> ■ In Delete Confirmation Message, type the message to be displayed when a record is deleted. ■ In Save Before Navigation, type the message to be displayed when you navigate through model data without saving the changes made.

The following options are available in the **Dataset** tab.

Business Object field	<p>Type the business object to be used. The Business Object field is automatically updated when you specify a method in the Web Service for 'Get' Operation field, or when you specify a schema fragment in the Instance Schema tab.</p>
Iterator Size field	Type an integer value to specify the number of records to retrieve

	<p>at a time.</p> <ol style="list-style-type: none"> 1. Click  to display the Select WebService Binding Operation dialog box, and select a method, and click OK. 2. On specifying the Web Service for 'Get' Operation, the following methods are automatically assigned, if available in the Web service interface: <p><code>Get<TableName>Object</code> is assigned as the Get method.</p> <p><code>GetNext<TableName>Objects</code> is assigned as the Next method.</p> <p><code>GetPrevious<TableName>Object</code> is assigned as the Previous method.</p> <p>If invalid or empty, the model ID is automatically updated in the Model ID field.</p> <p>The Business Object field is automatically populated with the business object to be associated.</p> 3. Click  to delete a method name.
Web Service for 'Get' Operation field	Click  to display the Select WebService Binding Operation dialog box, and select a method, and click OK.
Web Service for 'Next' Operation field	Click  to display the Select WebService Binding Operation dialog box, and select an option to specify the method from which the next set of business objects is to be retrieved.
Web Service for 'Previous' Operation field	Click  to display the Select WebService Binding Operation dialog box, and select an option to specify the method from which the previous set of business objects is to be retrieved

When you select any one of the Get, Next, or Previous methods, the other two are automatically assigned by the application.

The following options are available in the **Events** tab. See the *AppWorks Platform API Guide* for information about each input UI.

On Request	Select an option to specify the method to be executed before sending a request to the backend.
On Response	Select an option to specify the method to be executed on receiving a response from the backend.
On Data Completed	Select an option to specify the method to be executed after response data is bound to the controls.
On Before Synchronization	Select an option to specify the method to be executed before synchronizing data with the backend.
On Synchronized	Select an option to specify the method to be executed after synchronizing data with the backend

Alternatively, for the fields, click  to display the Script Editor and specify the methods.

The following options are available in the **Data Events** tab.

On Before Insert	Select an option to specify the method to be executed before inserting a new business object in the model.
On Insert	Select an option to specify the method to be executed on creating a business object, before its values are bound to the control.
On After Insert	Select an option to specify the method to be executed after creating a business object.
On Before Change	Select an option to specify the method to be executed before changing data in the model.
On Change	Select an option to specify the method to be executed after changing data in the model.
On After Change	Select an option to specify the method to be executed after the data is changed on the model.
On Before Delete	Select an option to specify the method to be executed before deleting a business object from the model data.
On Delete	Select an option to specify the method to be executed before deleting the business object from the model data and after deleting the associated UI.
On After Delete	Select an option to specify the method to be executed after deleting the business object from the model data

Alternatively, for the above fields, click  to display the Script Editor and specify the methods.

The following options are available in the **Namespaces** tab.

Default Namespace field	Displays the default namespace of the current model.
Namespace column	Displays all namespaces available in the WSDL for the model.
Prefix column	<p>Displays the prefixes available for corresponding namespaces in the WSDL of the model.</p> <p>Note: You can also specify a custom prefix. If specified, all instances of the namespace, such as in references, tree, and XML Editor, are automatically updated. However, changes in the prefix are not automatically updated in the script.</p>

The following options are available in the **Associations** tab.

Single Transaction option	<p>Select to enable the Single Transaction functionality, using which changes made to associated models can be saved in a single transaction at run time.</p> <ul style="list-style-type: none"> ■ The Single Transaction functionality is not available in case of WS-AppServer models or models associated with WS-AppServer
---------------------------	---

	<p>models.</p> <ul style="list-style-type: none"> ■ For information about setting single transaction while associating models, see Adding and Associating Multiple Models.
Parent Model	<p>Select an option to specify the model with which the current model is to be associated.</p> <p>Restriction*: It is not possible to associate a model with itself. Also, models that do not have parameters available for association cannot be associated with other models.</p> <p>*Note: The parameters available for the selected model automatically appear in the Request Parameter column.</p>
Field column	<p>For each parameter, click  to display the Model References dialog box, and select an option to specify the field that is available for association.</p>
Parent Model field column	<p>For each parameter, click  to display the Model References dialog box, and select an option to specify the field of the parent model with which it has to be associated.</p> <p>The Parent Model Field column is editable only if a parent model is specified</p>

The following options are available in the **Gateway** tab.

Gateway URL field	Type the URL of the gateway to which the model will send requests.
Receiver field	Type the distinguished name (dn) of the service to which the gateway will forward requests.
Timeout field	Type an integer value to specify the duration for which the web server must wait for a response from the Web service.

The following options are available in the **WS-AppServer** tab.

Apply Access Control option	Select to allow to enable, disable, hiding, and display controls in an XForm based on business logic available in the WS-AppServer.
Initialization Required option	Select to provide default values for a control in the XForm when a new record or business object is inserted.
Constraint Validation option	Select to enable server-side validation of the data that is displayed in the XForm.
Before Validation	Select an option to specify the method to be executed before the validate request is sent to the WS-AppServer.
After Validation	Select an option to specify the method to be executed after the validate response is received from the WS-AppServer

Property sheet of controls, regions, and AppPalletes

You can right-click a control, region, or App Palette in the Designer Area and choose Properties to view its property sheet. The <UI element name> window appears in the XForms Designer and displays the following options. However, not all options given below are applicable to all UI elements. Depending on the UI element, appropriate options display in its property sheet.

For information about setting the properties of an XForm and the basic properties of a control, see [Setting Properties for Controls, Regions, and App Palettes](#) and [Setting Properties of an XForm](#) respectively.

ID field	Type a unique identifier for the control. This is a design-time property.
Tab Index field	Type an index for the tab order of the control. This is a design-time property.
Model	Select an option to specify the model with which the control is to be bound. If not specified, the model of the parent control is considered. By default, child controls inherit the model attributes of their parent controls, so if you set the model attributes on a parent control all its child controls are bound to that model. This is a design-time property.
Reference field	<p>Click , select the business attribute, whose data is to be bound to the control, from the Request - <model name> dialog box that appears, and click OK. A string, which denotes the search path expression (XPath) of the business attribute, displays in the Reference field. This is a design-time property.</p> <ul style="list-style-type: none"> ■ When you click See namespaces for prefixes in the Request - <model name> dialog box to view related details, the Namespaces tab of the Model Properties dialog box is displayed. ■ The namespace of the model specified in the Reference field is also displayed as a tooltip.
Enumerated check box	Select to enable the Select and List controls to hold enumerated values. This is a design-time property.
Style Class field	Type the name of the style class to be applied on the control. The specified style class is picked from the style sheets specified for the XForm. This is a design-time property.
Controls radio button	Select to specify that only controls can be added and displayed in an App Palette. This option is only available in the property sheet of an App Palette.
Application radio button	Select to specify that only applications can be added and displayed in an App Palette. This option is only available in the property sheet of

	an App Palette.
Resizable option	Select to enable the resizing of the App Palette at run time. This option is only available in the property sheet of an App Palette.
Title Bar option	Select to display a title bar for the App Palette at run time. This option is only available in the property sheet of an App Palette.
Tooltip field	Type the text to be displayed as a tooltip for the control. You can also specify this value programmatically by setting a value for the title property of the control.
Load content when tab page is selected option	Select to set the control as a load-on-demand control. This means that when a page is loaded for the first time at run time, the control is initialized (loaded with content) only when you click it.
Data set field	Click  and specify the data set source in the Pre-fill data source for <control> dialog box.
Checkboxes option	Select to display the check boxes that appear in a Table control. Clear this option to hide the check boxes. This is a design-time property.
Save Grid State option	Select to retain the modifications made to a Table control (column size and placement) when an XForm is opened again at run time.
Display ContextMenu option	Select to enable the context menu for a Table control. Clear this option to hide the context menu. This is a design-time property.
Toggle Columns option	Select to enable the toggle feature for columns in a Table control. Clear this option to disable the toggle feature. This is a design-time property.
Auto Insert option	Check to add a new, empty row to a Table control when you enter valid data in the last row of the table. This is applicable for a Table control associated with a transactional data model only.
Multiple Selection option	Check to enable the selection of multiple records in a List control. This is a run-time property.
Source URL	Type the URL of the XForm to be loaded in the Frame control. The XForm loaded in a Frame is not editable during design time.
Task Part option	Check to create a task part for the control. Selecting the check box, displays the Task Part Details pane, as described in the table below
Zoom Properties pane	Use the options available to specify Zoom properties, as described in the table below.
ClickChoice pane	Use the options available to specify context-menu options for a control that provide access to related, detailed information. For more information, see Creating ClickChoice Relations .
Data Value pane	Use the options available to specify True and False values for Check controls, as described in the table below.
Collapsibility pane	Use the options available to specify collapsibility properties for the

	Groupbox control, as described in the table below.
Events pane	Use the options available to specify events for controls, as described in the table below.
Element bar options pane	Use the options available to specify Element bar properties, as described in the table below.
Synchronized Dialog Options pane	Use the options available to specify properties of synchronized dialogs for Table controls, as described in the table below.
Tab Settings pane	Use the options available to specify the orientation and text direction for Tabs in a region, as described in the table below. This option is only available in the property sheet of a Region.
Settings pane	Use the options available to specify details about the controls to be loaded in an App palette, as described in the table below. These common options are also available when applications are loaded in an App Palette.
Application Definition pane	Use the options to specify the application definition of the Application to be loaded in an App Palette, as described in the table below. This option is only available in the property sheet of an App Palette

The following options are available in the **Task Part Details** pane.

Associated Web Services	Click  and select the Web services that are to be associated with the task part. The button is disabled if no model with a Web service is present on the XForm. You must add a Web service to the XForm to associate it with a control in the XForm.
Name	Select New or Existing , and type the appropriate name in the editable field to define the name of the task part. It is possible to group multiple controls by assigning them to a single task part name.
Not Available Mode radio buttons	Select Hide or Disable to define the control's display mode, when when users who do not hold write permissions open the XForm.

The following options are available in the **Zoom Properties** pane.

Zoom URL	Type the URL of the XForm to load in the lookup dialog box. Alternatively, click  , and select the XForm from the Select URL dialog box. This is a design-time property.
Zoom Field	Type the XPath of the business attributes to be returned from the lookup page. This is a design-time property.
Before Zoom	Select the event handler to execute before the lookup page is opened.
After Zoom	Select the event handler to execute after the lookup page is closed.

The following options are available in the **Data Value** pane.

True Value	Type a value for which the Check control is selected.
False Value	Type a value for which the Check control is cleared.

The following options are available in the **Collapsibility** pane.

Collapsible option	Select to make the Groupbox control collapsible. When selected, a button appears in the title bar of the control. Clicking the button expands or collapses the control. Default setting is False.
Expand on load option	Select to expand the Groupbox on loading an XForm. Selected by default.
Load content when group is expanded option	Sets the control as a load-on-demand control. This means that when a page is loaded for the first time at run time, the control is initialized (loaded with content) only when you expand it.

The following options are available in the **Events** pane. You can use the Message Map for associating data with the events of a control. For details, see [Using Message Map](#).

Click	Select the method to be executed when you click a Button control.
Data Bind	Select the method to be executed before the data is bound to the control.
Change	Select the method to be executed when you modify data.
Validate	Select the method to be executed when you change the value of a control that is based on custom script validation, and for which you can perform your own validation.
InFocus	Select the method to be executed when the control is highlighted or focused upon.
OutFocus	Select the method to be executed when the control is taken out of focus.
Collapse	Select the method to be executed when you collapse a Groupbox control.
Expand	Select the method to be executed when you expand the Groupbox control.
On Row Click	Select the method to be executed when you click a row in the Table control.
On Row Select	Select the method to be executed when you select a row in the Table control.
On Row Checked	Select the method to be executed when you select a check box in the Table control.
On Column Hidden	Select the method to be executed when you hide a column in the Table control.
On Set Default	Select the method to be executed when you in the Table control

The following options are available in the **Element bar options** pane.

Control bar option	Select to enable the control bar.
Pagination bar option	Select to enable the pagination bar

The following options are available in the **Synchronized Dialog Options** pane.

Dialog URL field	Type the URL of the XForm to be used as the synchronized dialog.
On Dialog Open	Select an option to specify the event handler to be executed before opening the synchronized dialog.
ID field	Type a unique ID for identifying and referring the synchronized dialog.
Caption field	Type the text to be displayed in the XForm titlebar and tabcaption.
Description field	Type the text to be displayed in the XForm titlebar. This will be the display name for the XForm in the navigational tree.
Docked option	Select to dock the synchronized dialog, when viewed in the AppWorks Platform Classic view.
Left field	Type an integer value to specify the position of the left edge of the synchronized dialog.
Top field	Type an integer value to specify the position of the top edge of the synchronized dialog.
Width field	Type an integer value to specify the width of the synchronized dialog.
Height field	Type an integer value to specify the height of the synchronized dialog.

The following options are available in the **Tab Settings** pane.

Top radio button	Select to position the tabs at the top of the region.
Right radio button	Select to position the tabs to the right side of the region.
Bottom radio button	Select to position the tabs at the bottom of the region.
Left radio button	Select to position the tabs to the left side of the region.
Vertical radio button	Select to display the text in the tab vertically.
Horizontal radio button	Select to display the text in the tab horizontally.
Roller radio button	Select to display a roller to scroll through the tabs at run time.
Extender radio button	Select to display an extender (navigation arrows) to scroll through the tabs at run time.

The following options are available in the **Settings** pane.

Title Bar option	Select to add a title bar in the App Palette. In case an application is displayed in the App Palette, the title bar is a combination of the Caption and Description, as specified in the application definition.
Store in Preferences	Select to save the state of an application in the App Palette as a user preference. If set to true, the application's state is stored and is

option	retained when the application is launched again.
Dockable option	Select to enable the App Palette to dock in the region in which it is located.
Display in view manager	Select an option to display the App Palette control in the View Manager ( menu in the toolbar of an App Palette) and define its appearance. Options are: <ul style="list-style-type: none"> ■ self: The App Palette will be registered as an option in the View Manager menu. ■ none: The App Palette will not be registered as an option in the View Manager menu. ■ custom: The App Palette will be registered as an option in the View Manager. A separator displays that groups all App Palettes with the same custom group name

The following options are available in the **Application Definition** pane.

URL field	Type the URL of the application to be opened in an App Palette.
Id field	Type a unique identifier for the application.
Caption field	Displays the caption of the application. This name will appear in the taskbar as the tab caption. This will also appear as the titlebar text, as a combination of <caption> and <description> if both are different, and <caption> if both are same.
Description field	Displays the description of the application. This name will appear in the titlebar of the application as a combination of <caption> and <description>.
Region field	Type the region in which the App Palette is loaded.
Docked option	Select to specify that the App Palette will be docked in the region, by default.

Controls available for creating property sheets

The property sheet of a control contains options that you can use to set its design-time and run-time properties. As with other XForms controls, you can associate composite controls with a property sheet.

To create a property sheet for your composite control:

1. In the Untitled Composite Control - Composite Control wizard, click **Property Sheet**. The Untitled User Interface - User Interface window opens.
2. Click the Toolbox tab. The Untitled User Interface - User Interface window opens, displaying the controls commonly used in property sheets.

3. To add the controls to the property sheet, click the controls or drag them to the Designer area.

The controls available for creating a property sheet for a composite control are as follows.

Date	Adds a calendar control to the property sheet.
Event	Adds a field with a lookup button to the property sheet. At runtime, it displays a list along with a look-up button. Clicking the look-up button creates a dummy function for the event in the Script Editor.
Id	Adds an input field with id as its label.
Model	Adds a Model list to the property sheet. At runtime, the list displays the models associated with the XForm to which the composite control is added.
Reference	Adds a Reference field to the property sheet.
Webservice	Adds a Webservice field with a lookup button to the property sheet. At run time, it opens the Web Services window that displays all available Web services.
XML	Adds an output field with a lookup button. At runtime, you can click the look-up button and provide XML in the XML Editor window that appears.

Each of the controls is associated with a property sheet.

The following options are available in the property sheet and govern the behavior of the controls.

ID	Refers to a unique identifier for the control. It is possible to modify the autogenerated IDs.
Reference	Refers to the reference of the control. In case of Event control, the reference represents the event name. It is not possible to modify the Reference for the Id , Model , and Reference controls.
Display Format	Refers to the display format. This option is available only for the Date control.

Predefined XForms

The following list is the predefined files that exist in AppWorks Platform.

- button
- check
- contents
- contextmenu
- format

- frame
- getlabelsbyguid
- getlabelsforlanguage
- group
- groupbox
- image
- input
- inputdialog
- label
- labelselector
- languagemanager
- model
- models
- select
- selectcontent
- selecturl
- setassociateddataset
- showmethods
- tab
- table
- toolbarbutton
- xform
- xformsdef_zoom
- xformtranslator
- zoombuttongroup
- _xform_format
- _xform_preview
- _xform_properties_button
- _xform_properties_check
- _xform_properties_controlbarbutton
- _xform_properties_frame
- _xform_properties_group
- _xform_properties_groupbox
- _xform_properties_image
- _xform_properties_input
- _xform_properties_label

- `_xform_properties_model`
- `_xform_properties_models`
- `_xform_properties_select`
- `_xform_properties_tab`
- `_xform_properties_table`
- `_xform_properties_toolbatbutton`
- `_xform_properties_xform`
- `_xform_selectcontent`
- `_xform_selecturl`
- `_xform_setapplicationdefproperties`
- `_xform_showmethods`
- `_xform_translator_contextmanager`
- `_xform_translator_getlabelsbyguid`
- `_xform_translator_getlabelsforlanguage`
- `_xform_translator_getlabelsforlanguage`
- `_xform_translator_inputdialog`
- `_xform_translator_labelselector`
- `_xform_translator_languagemanager`
- `_xform_translator_xformtranslator`
- `_xform_xformsdef_clickchoice`
- `_xform_xformsdef_dialog`
- `_xform_xformsdef_zoom`
- `_xform_zoombuttongroup`
- `_xform_zoomdef`

Caution: Ensure that you do not use any of the predefined names to name your XForms.

Predefined and specific formats for data types

This topic describes the predefined and specific data type formats.

Predefined formats for data types

AppWorks Platform XForms provides you with various predefined formats for the Integer, String, and Date data types. You can select the appropriate format from the Predefined Format list in the Format tab of the Formatting Options dialog box.

Available predefined formats

Data Type	Predefined Format	Script
Integer	Credit Card	creditcard
String	Text	text
	E-mail	mail
Date	short date (for example, 8/15/1980)	shortdate
	short date-short time (for example, 8/15/1980 20:55)	shortdate shorttime
	short date-long time (for example, 8/15/1980 8:55:01PM)	shortdate longtime
	long date (for example, Friday, August 15, 1980)	longdate
	long date-short time (for example, Friday, August 15, 1980 20:55)	longdate shorttime
	long date-long time (for example, Friday, August 15, 1980 8:55:01)	longdate longtime
	long time (for example, 8:55:01 PM)	longtime
	short time (for example, 20:55)	shorttime

You can use the values mentioned in the Script column to programmatically set the data type for a control. For more information about Format type, see the *AppWorks Platform API Guide*.

Specific formats for data types

For each data type, you can specify the exact format in which values must be entered in a control. You can type the appropriate format in the Specific Format field in the Format tab of the Formatting Options dialog box.

Characters supported for number formats

Character	Character Name	Description
0	Zero placeholder	If the value that is being formatted does not have a number in the 0 position, then 0 is copied. Otherwise, that number is retained.
#	Digit placeholder	If the value that is being formatted has a number in the # position, then that number is copied. Otherwise, nothing is stored in that position.
.	Decimal placeholder	This determines the location of the decimal separator in the formatted value.

Character	Character Name	Description
,	Thousand separator	This specifies whether a comma separator is to be used in the formatted value or not.
%	Percentage placeholder	The value is multiplied by 100 before formatting it. The percentage sign displays as specified in the formatted value.
Other characters	All other characters	All other characters remain in the same position as they appear

Note

- You can use only the #, ., and 0 characters to define the format for the Amount data type.
- Depending on locale settings, the decimal placeholder and thousand separator are used interchangeably.

Characters supported for date formats

Character	Character Name
d, dd, D, or DD	Date
ddd, dddd	Weekday
M, MM	Month
MMM	Abbreviated month name
MMMM	Month name
yy, yyyy, YY, or YYYY	Year
H, HH	Hour in 24hr time format
h, hh	Hour in 12hr time format
m, mm	Minute
s, ss	Second
f, ff, fff	Millisecond

Note

- While defining the display format for a date, you can specify the number of digits to be displayed for milliseconds (.f, .ff, or .fff). If, at run time, you enter more than the specified number of digits, a truncated value is displayed. Existing database values are also truncated and displayed according to the specified display format. However, millisecond values are saved up to a maximum of three places of decimals, if available.
- You can use of two digit (day), four digit (day+month), or eight digit (day+month+year) numbers without separators, to specify dates. The date is formatted based on the

locale's short date format. For example, if you type 10302007, the application automatically formats it as 10/30/2007 for the English (United States) locale.

You can use various combinations of characters to specify formats.

Sample formats for the numerical data types

Specific format	Backend value	Result displayed at run time
0#,##0.00	12345678.9	12,345,678.90
(##)-##0	12341234567	(1)-23412(34)-567
###	123.45	123
#0.# %	0.7589	75.8%
0000.00	12.2	0012.20

Sample formats for the Data data type

Specific format	Result displayed at run time
dd-MMM-yyyy	03-Jul-1980
dd/MMMM/yy	03-July-80
YYYY-MM-DD HH:mm:ss:fff	1980-07-03 15:40:22:325

The predefined formats available for various data types depend on the locale settings of your computer.

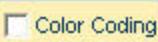
Script editor toolbar

To display the Script Editor:

- Click the **Script** tab or select **View Script** from the context menu. Alternatively, press F7 or click  in the Properties dialog box of a control.
The Script Editor pane appears and the cursor is displayed at the first function associated with the control.

The following table lists the buttons available in the Script Editor toolbar.

Button	Button Name	Description
	Toggle Find Pane	Displays and hides the Find pane.
	Find Next	Displays the next occurrence of text that matches the specified search criteria.
	Check Syntax	Validates the syntax of the script. Alerts display in

Button	Button Name	Description
		case of errors.
	Reload script without saving changes	Displays original script without saving any changes made in the Script Editor.
	Help	Displays the Script Editor topic of the Application Developer Online Help.
	Color Coding	Enables or disables the color coding of syntax keywords. The Color coding functionality is available in the Internet Explorer browser only.

Note: Color coding functionality is not available in non-IE browsers.

The following table lists the various fields in the Find pane.

Field	Description
Find	Enter the text to search and click  to view it.
Replace With	Enter new text and click  to replace the text specified as the search criteria in the Find field.
Goto Line	Enter a line number and click  to view it.

Selectors in cascading style sheets

The following table lists the selectors available in `default.css` and `xdefault.css`. A style sheet for which a selector is available is marked using 'x'.

The selectors are listed in the cascade order in which they occur in the style sheets. While customizing a style sheet, maintain the cascade order of selectors to prevent errors.

Selector Name	Description	default.css	xdefault.css
body	HTML tag that specifies the start and end of the document body. This tag defines font size, font color, and background color for the entire document, and is inherited by controls subject to control properties.	x	x
table	HTML tag that specifies properties for a Table control. It is used in <code>default.css</code> to set the font size to 100% as the table object does not inherit font size correctly.	x	
button	HTML tag that specifies the properties of the Button control.	x	
.input	Class selector used for editable elements that receive user input from the keyboard. It is always marked in white color for easy identification and recall of input-ready areas.	x	

Selector Name	Description	default.css	xdefault.css
textarea	HTML tag that specifies the properties of the textarea control.	x	
.light	Class selector applicable to menus, odd-numbered rows in grids in AppWorks Platform interfaces, secondary windows (such as dialog boxes, error messages, and wizards), and background color.	x	
.lightmedium	Class selector applicable to even-numbered rows in a grid, inactive tab backgrounds, and button color of select boxes.	x	
.medium	Class selector applicable to buttons, grid headers, and background color of toolbar-pressed items.	x	
.mediumdark	Class selector applicable to grid footers, inactive applications, section headers, and context-menu items in a tree.	x	
.dark	Class selector applicable to application headers.	x	
.navbody	Class selector applicable to the navigational elements, such as the default tree menu.	x	
.mainbody	Class selector applicable to the default container in AppWorks Platform application pages. This is almost always similar to the body tag in presentational properties.	x	
.toolbar	Class selector applicable to the application toolbar in AppWorks Platform application pages.	x	
.outfocus	Class selector applicable to a tree node when it is selected, but not focused.	x	
.docked	Class selector applicable to the AppWorks Platform Application Container when it is docked. It affects the appearance of the area around the Application Container in the docked state.	x	
.undocked	Class selector applicable to the AppWorks Platform Application Container when it is undocked. It affects the appearance of the area around the Application Container in the undocked state.	x	
.desktop	Class selector applicable to the AppWorks Platform desktop.	x	
.maincontainer	Class selector that specifies the top-level container that follows the toolbar, if present, in an XForm. You must specify the appropriate padding required. It is an invisible structural element.		x
.simplecontainer	A simple structural element that controls scrollbars and defines areas within groups. It is an invisible structural element.		x
.compositecontainer	Container for grouping controls with implementation similar to that of an HTML table. It is an invisible structural element.		x
.compositecontainerrow	Class selector that renders the rows that display in a grouping control. It is an invisible structural element.		x

Selector Name	Description	default.css	xdefault.css
.compositecontainercell	Class selector that renders the columns in a grouping control. It is an invisible structural element.		x
.h_layout	Class selector that applies a horizontal layout to contents. It is an invisible structural element.		x
.v_layout	Class selector that applies a vertical layout to the contents. It is an invisible structural element.		x
.v_layout_i	Class selector that applies a vertical layout to the contents of a Group control. It is an invisible structural element.		x
.f_layout	Class selector that applies a position-based layout, which can be used with anchors. It is an invisible structural element.		x
.fieldsbox	Class selector that serves as a container for primitive controls. It is an invisible structural element.		x
.f_layout .fieldsbox	Descendant selector that renders the fieldsbox selectors in a free layout. It is a structural construct and is invisible.		x
.f_layout .fieldsbox textarea	Descendant selector that renders the textarea of a fieldsbox in a free layout. It is a structural construct and is invisible.		x
.choicebox	Class selector that serves as a container for Check controls and their labels. It is an invisible structural element.		x
.fieldsbox .choicebox	Descendant selector that renders the choicebox selectors in the fieldsbox container. It is a structural construct and is invisible.		x
.elementbar	Class selector that serves as a container for buttons in control bars and pagination bars associated with controls such as Table and Groupbox. It is an invisible structural element.		x
.controlbar	Class selector that serves as a container for buttons that control actions such as insert, refresh, and delete for a given element (tables, groups, or tabs). It is an invisible structural element.		x
.paginationbar	Class selector that serves as a container for pagination buttons such as First, Previous, Next, and Last for a given element (tables, groups, or tabs). The find action is also used in the pagination bar. It is an invisible structural element.		x
.splittercontainer	Structural element that serves as a container for the splitter control.		x
.split_left	Structural element that serves as a container for the left part of the splitter control.		x
.split_right	Structural element that serves as a container for the right part of the splitter control.		x
.split_top	Structural element that serves as a container for the top part of the splitter control.		x

Selector Name	Description	default.css	xdefault.css
.split_bottom	Structural element that serves as a container for the bottom part of the splitter control.		x
.input	Class selector used for editable elements that receive user input from the keyboard. It is always marked in white color for easy identification and recall of input-ready areas.		x
input	HTML tag that specifies the properties for all input elements.		x
fieldset input	Descendant tag selector to control Check controls and radio buttons in certain migration scenarios. It is an invisible structural element and need not be customized.		x
textarea	HTML tag that specifies the properties of the Textarea control.		x
.output	Class selector that defines the position and appearance of an Output control.	x	x
label, .h_label, .v_label	Group selector that specifies the default position and appearance of a label object, horizontal label, and vertical label in an XForm, respectively.	x	x
button, .h_button, .v_button	Group selector that specifies the default appearance of a button object, horizontal button, and vertical button in an XForm, respectively.	x	
.lookup	Class selector that defines the position and appearance of zoom (lookup) buttons that are positioned next to fields.	x	x
.lookup .disabled, .lookup .output	Group selector that specifies the appearance of disabled buttons and output fields in XForms that appear on using zoom controls.	x	
.lookup img	Class selector that renders images inside zoom (lookup) buttons and displays the image proportionately.		x
.h_button	Class selector that defines the position of consecutive buttons arranged horizontally in a row.		x
.v_button	Class selector that defines the position of buttons that display below one another.		x
.h_image	Class selector that defines the position of consecutive images arranged horizontally in a row.		x
.v_image	Class selector that defines the position of images that display below one another.		x
.elementbar .separator	Descendant selector that defines the position of the separator in an element bar.		x
.elementbar button	Descendant selector that defines the position and appearance of buttons that are displayed in an element bar, which includes the control bar and the pagination bar.	x	x
table.groupby	Class selector that specifies the Table control as a		x

Selector Name	Description	default.css	xdefault.css
	container for group controls. It is a structural construct and is invisible.		
.groupbox .groupheader	Descendant selector that defines the position and appearance of headers for groups in the default state.	x	x
.groupbox .expand	Descendant selector that defines the position and appearance of headers for groups that are collapsed. The group header indicates the 'expand-now' state.	x	x
.groupbox .collapse	Descendant selector that defines the position and appearance of the headers for groups that are expanded. The group header indicates the 'collapse-now' state.	x	x
.groupcontent	Descendant selector that specifies the controls that display in a Groupbox control. It also specifies the overflow and border.	x	x
.invisiblegroupbox	Class selector that specifies the Group control for the XForm interface. It is an invisible control.		x
.simplecontainer iframe	Descendant selector that specifies the position and appearance of the Frame control in an XForm. It does not affect the iframe UI elements used at other places across AppWorks Platform.	x	x
.stackedtabcontainer	Class selector that specifies the position and appearance of container for vertically-stacked tabs. It is a structural element and is invisible.	x	x
.stackedtabattop	Class selector that specifies the position and appearance of buttons that display on top of vertical tab groups. It is a part of .stackedtabcontainer.	x	x
.stackedtabatbottom	Class selector that specifies the position and appearance of buttons that display at the bottom of vertical tab groups. It is a part of .stackedtabcontainer.	x	x
.stackedtabcontent	Class selector that specifies the position and appearance of tab content that display below buttons. It is a part of .stackedtabcontainer.	x	x
.tabcontainer	Class selector that defines the background of a tab container. It is set to "transparent" to display the parent container's background. If a color is specified, it overrides other background properties such as desktop color. The properties of this class selector are common for all horizontal tabs. It is an invisible structural element and contains the .tabstripattop, .tabstripatbottom, .tabcontent, .activetabattop, .activetabatbottom, .inactivetabattop, and .inactivetabatbottom class selectors.	x	x
.tabstripattop	Class selector that defines the properties for the top tab-strip. It is a part of .tabcontainer, and contains the .activetabattop and .inactivetabattopclass selectors. It is an invisible structural element.	x	x
.tabstripatbottom	Class selector that defines the properties for the bottom tab-strip. It is a part of .tabcontainer, and	x	x

Selector Name	Description	default.css	xdefault.css
	contains the .activetabattop and .inactivetabattop class selectors. It is an invisible structural element.		
.tabcontent	Class selector that defines the content and border for a tab. It is common for top and bottom tabs, and is a part of .tabcontainer.	x	x
.activetabattop	Class selector that defines the position and appearance of a caption that appears at the top of the selected tab.	x	x
.activetabatbottom	Class selector that defines the position and appearance of a caption that appears at the bottom of the selected tab.	x	x
.inactivetabattop	Class selector that defines the position and appearance of a caption that appears at the top of an inactive tab.	x	x
.inactivetabatbottom	Class selector that defines the position and appearance of a caption that appears at the bottom of an inactive tab.	x	x
.tabcontentholder, .stackedtabcontentholder, .tabpageholder	Class selectors for the holders that form the structure of the tabs.	x	
th	HTML tag that defines the properties of the header cell.		x
th label	Descendant selector that defines the properties of the label in a header cell.		x
th input	Descendant selector that defines the properties for input in a header cell.		x
td input	Descendant selector that defines the properties for the label in a table cell.		x
tr, td	Group selector that defines the properties for rows and cells. It is an invisible construct and should not be customized.		x
thead	HTML tag that defines the properties of the header row.		x
.oddrow, .evenrow	Group selector that defines the properties of rows of a Table control in AppWorks Platform XForms. Here, it specifies a uniform row height for all types of grids. The height must be specified in pixels.		x
.data	Class selector applicable to a Table control that distinguishes it as a data-entry table, in which at least one column is configured for user input through an Input or Select control.	x	x
.data td	Descendant selector that defines the properties for cells in a data-entry table.	x	x
.data th	Descendant selector that defines the properties for header cells in a data-entry table.	x	x
.data .oddrow	Descendant selector that defines the properties for odd rows in a data-entry table.	x	x

Selector Name	Description	default.css	xdefault.css
.data .evenrow	Descendant selector that defines the properties for even rows in a data-entry table.	x	x
.data .fieldsbox	Descendant selector that defines the properties for a field box in a data-entry table. This is an invisible construct and should not be customized.		x
.data .input	Descendant selector that defines the properties for an input box in a data-entry table.	x	x
.data .output	Descendant selector that defines the properties for an output box in a data-entry table.	x	x
.data .error	Descendant selector that defines the properties for an input box in a data-entry table, when the input box contains incorrect values.		x
.data input	Descendant selector that defines the properties for an input object and is applicable to the Check control in a data-entry table. It is an invisible construct and should not be customized.		x
.data .lookup	Descendant selector that defines the properties for zoom (lookup) buttons positioned next to input boxes in a data-entry table.		x
.data .selectbox	Descendant selector that defines the properties of the Select control in a data-entry table. It is an invisible construct and should not be customized.		x
.data .ifocus	Descendant selector that defines the properties for an input box that is in focus, in a data-entry table.	x	
.navigational	Class selector applicable to the Table control that distinguishes it as a navigational table, in which all columns are configured to navigate to a synchronized dialog (related view). Clicking any row opens the synchronized dialog.	x	x
.navigational td	Descendant selector that defines the properties for cells in a navigational table.	x	x
.navigational th	Descendant selector that defines the properties for header cells in a navigational table.	x	x
.navigational .oddrow	Descendant selector that defines the properties for odd rows in a navigational table.	x	x
.navigational .evenrow	Descendant selector that defines the properties for even rows in a navigational table.	x	x
.navigational .input	Descendant selector that defines the properties for an Input control in a navigational table.		x
.navigational .lookup	Descendant selector that defines the properties for zoom (lookup) buttons positioned next to input boxes in a data-entry table.		x
.display	Class selector applicable to the Table control in AppWorks Platform XForms that distinguishes it as a 'display table', in which all columns are configured to display information.	x	x

Selector Name	Description	default.css	xdefault.css
.display td	Descendant selector that defines the properties for cells in a display table.	x	x
.display th	Descendant selector that defines the properties for header cells in a display table.	x	x
.display .oddrow	Descendant selector that defines the properties for odd rows in a display table.	x	x
.display .evenrow	Descendant selector that defines the properties for even rows in a display table.	x	x
.display .input	Descendant selector that defines the properties for Input controls in a display table.	x	x
.display .lookup	Descendant selector that defines the properties for zoom (lookup) buttons positioned next to input boxes in a display table.		x
.selectbox , eibus\:dropdown	Group selector that defines the eibus tag-level style for a Select control.	x	x
.listbox	Class selector that defines the style for the container of a List control.	x	x
.selectfield	Class selector that defines the style for the input field and the list button in the .selectbox class. It does not affect the list options.	x	
.selectdropdown	Class selector that defines the container for the list that displays when you click the Select control. It exists at the body-tag level and not at the levels lower than that of the .selectbox class.	x	
.option	Class selector that defines all options in the .selectdropdown and .listbox classes.	x	
.optionselected	Class selector that defines the selected options in the .selectdropdown and .listbox classes. The selected options are highlighted using the color defined in the .highlight class. You can specify a new color and override the highlight color.	x	
.selectfield input	Group selector that defines the input box that contains all selected options. Do not customize the input box as its appearance changes with the mode and state of the Select control.	x	x
.selectfield input.input, .selectfield input.output	Group selector used for maintaining proper cascade of Select control widths. For customized default.css files, add the selector and use the border-right width value specified for the .selectfieldinput selector.	x	
.selectfield button	Group selector that defines the button that invokes the list. Do not customize this button as its appearance changes with the mode and state of the Select control.	x	
.selectfield button.closed	Group selector that defines the closed state of the button that invokes the list. The list does not display in this state.	x	
.selectfield button.open	Group selector that defines the open state of the button that invokes the list. The list is displayed in this	x	

Selector Name	Description	default.css	xdefault.css
	state.		
.selectfield button label	Group selector that specifies the label of the button. It is hidden by default.	x	
.selectclose	Class selector that defines the area spanning the list that captures mouse clicks.	x	
.selectbox.disabled	Class selector that defines the appearance of a disabled Select control.	x	
.selectbox.disabled button.closed	Descendant selector that defines the appearance of a button in a disabled Select control.	x	
.input.selectdropdown	Class selector that defines the position of a Select control. Do not customize it.		x
.toolbar	Class selector that specifies the position and appearance of the toolbar.		x
.toolbar span	Descendant selector that defines the position and appearance of a span in the toolbar.		x
#modallayer	Id selector that defines the properties of the covering DIV that functions as the modal layer for the Uniform Feedback Object - AppWorks Platform Dialog. Customization, other than the modification of the background-color, are not recommended.	x	
.dialogcontainer	Class selector that defines position and appearance of the dialog area of the Uniform Feedback Object - AppWorks Platform Dialog.	x	
fb_message	Id selector that defines common properties for the message that displays in the Uniform Feedback Object. This can be further contextualized separately for notification, error, and AppWorks Platform Dialog messages. This is always a paragraph and can contain inline elements such as label and link.	x	
#fb_message label	Descendant selector that defines the presentational properties for the label inside the #fb_message selector.	x	
.dialogcontainer p#fb_message	Contextualized selector for #fb_message coming in dialogs.	x	
.fb_wrapper	Id selector that defines the position and appearance of the container for all Uniform Feedback Object AppWorks Platform Dialogs and the stacked list of error and notification messages. It is recommended not to customize it.	x	
#feedback_wrapper_detail	Id selector that defines the z-index of the wrapper for all Uniform Feedback Object AppWorks Platform Dialogs and the stacked list of error and notification messages. The wrapper comes around the container for Uniform Feedback Object dialogs.	x	
.feedbackbox	Class selector that defines the position and appearance of primary holders for the Uniform Feedback Object - AppWorks Platform Dialogs. These holders contain the title bar (#fb_titlebar) and the	x	

Selector Name	Description	default.css	xdefault.css
	message containers (#fb_container).		
.feedbackbox p	Descendant selector that defines the appearance of paragraphs inside the feedback box container for the Uniform Feedback Objects - AppWorks Platform Dialogs.	x	
.feedbackbox #fb_titlebar	Descendant selector that defines the position and appearance of the title bar of the Uniform Feedback Objects - AppWorks Platform Dialogs, error messages, and notification messages.	x	
.feedbackbox #fb_titlebar h1	Descendant selector that defines the position and appearance of the heading that appears in titlebar of the Uniform Feedback Objects - AppWorks Platform Dialogs, error messages, and notification messages.	x	
.feedbackbox #fb_buttons, .fb_notifybox #fb_buttons, .fb_errorbox #fb_buttons, .fb_wrapper #fb_buttons	Descendant selectors that define the position and appearance of buttons of the Uniform Feedback Objects - AppWorks Platform Dialogs, error messages, and notification messages.	x	
#fb_buttons img	Descendant selectors that define the position and appearance of images in the Uniform Feedback Objects - AppWorks Platform Dialogs, error messages, and notification messages.	x	
.feedbackbox #fb_container, .dialogcontainer #fb_container	Descendant selectors that define the position and appearance of the container in a Uniform Feedback Object. This can be further contextualized separately for AppWorks Platform Dialogs, error messages, and notification messages.	x	
#fb_fbdetail #fb_message	Id selector that defines the common properties for the message that displays in a Uniform Feedback Object. This can be further contextualized separately for AppWorks Platform Dialogs, error messages, and notification messages. This is always a paragraph and can contain inline elements such as label and link.	x	
.feedbackbox iframe	Descendant selector that defines the presentational properties for the iframe inside the feedbackbox selector.	x	
.feedbackbox #fb_controlbox, #fb_controlbox	Descendant selector that defines the position and appearance of the control box inside the feedbackbox selector. The control box contains the button that appears in the message area of the Uniform Feedback Object - AppWorks Platform Dialog.	x	
.feedbackbox #fb_controlbox button, #fb_controlbox button, .dialogcontainer button	Descendant selector that defines the position and appearance of the button that appears in the message area of Uniform Feedback Objects.	x	
.dialogcontainer .input	Descendent selector that defines the position and appearance of the button that appears in the message area of the Uniform Feedback Object - AppWorks Platform Dialog.	x	

Selector Name	Description	default.css	xdefault.css
.feedbackbox #fb_controlbox #extender	Descendent selector that defines the position and appearance for the extender that appears in the message area of the Uniform Feedback Object - AppWorks Platform Dialog. This extender has links to show More/Less.	x	
.feedbackbox #fb_controlbox #extender a	Descendent selector that defines the position and appearance for the links inside the extender (show More/Less) that appear in the message area of the Uniform Feedback Object - AppWorks Platform Dialogs.	x	
#fb_extension	ID selector that defines the container for the area that can be expanded and collapsed in the SOAP Fault details message.	x	
#fb_extension div#fb_details	Descendent selector that defines the properties of the details container, located inside the area that can be expanded and collapsed in the SOAP Fault details message .	x	
#fb_status	ID selector that defines the position and appearance of the Uniform Feedback Object that indicates the status - status message.	x	
#fb_status #fb_titlebar	Descendant selector that defines the position and appearance of the titlebar of the Uniform Feedback Object - status message.	x	
#fb_status #fb_titlebar h1	Descendant selector that defines the position and appearance of the heading that appears in the titlebar of the Uniform Feedback Object - status message.	x	
#stackListWrapper	Id selector to set the position and appearance of the wrapper of a stacked list of Uniform Feedback Object error and notification messages.	x	
#stackListHeader	Id selector to set the position and appearance of the header of a stacked list of the Uniform Feedback Objects - error messages and notification messages.	x	
#stackListHeader label, #stackListHeader label em, #stackListHeader span, #stackListHeader span img, #stackListHeader #fb_pinImage	Descendant selectors to set the position and appearance of the various tags in the header of a stacked list of the Uniform Feedback Objects - error messages and notification messages.	x	
span.notificationdisplay *, span.errordisplay *	Class selectors for defining the properties of all tags, included in notification and error messages, that display in the Stacked list header.	x	
.stackListContainer	Class selector for the container that holds multiple Uniform Feedback Objects (notification and error messages), when they display as a list.	x	
.fb_notifybox	Class selector for the position and appearance of the Uniform Feedback Object - notification message.	x	
.fb_notifybox #fb_titlebar	Descendant selector that defines the position and appearance of the titlebar of the Uniform Feedback	x	

Selector Name	Description	default.css	xdefault.css
	Object - notification message.		
.fb_notifybox #fb_titlebar h1	Descendant selector that defines the position and appearance of the heading that appears in the titlebar of the Uniform Feedback Object - notification message.	x	
.fb_notifybox #fb_container	Descendant selector that is contextualized for the Uniform Feedback Object - notification message - to define the position and appearance of the container of the notification message.	x	
.fb_notifybox #fb_container p#fb_message	Contextualized id selector that defines common properties of the message area in the Uniform Feedback Object - notification message. This is always a paragraph and can contain inline elements such as label and link.	x	
.fb_notifybox #fb_container p#fb_message label	Contextualized id selector that defines common properties for the label in the message area of the Uniform Feedback Object - notification message. This is always a paragraph and can contain inline elements such as label and link.	x	
.fb_notifybox #fb_controlbox button	Selector for the button that appears in the message area of the Uniform Feedback Object - notification message.	x	
.fb_errorbox	Selector for the position and appearance of the Uniform Feedback Object - error message.	x	
.fb_errorbox #fb_titlebar	Descendant selector that defines the position and appearance of the titlebar of the Uniform Feedback Object - error message.	x	
.fb_errorbox #fb_titlebar h1	Descendant selector that defines the position and appearance of the heading that appears in the titlebar of the Uniform Feedback Object - error message.	x	
.fb_errorbox #fb_container	Descendant selector that is contextualized for Uniform Feedback Object (error messages) to define the position and appearance of the error message container.	x	
.fb_errorbox #fb_container p#fb_message	Contextualized id selector that defines the common properties for the message area in the Uniform Feedback Object - error message. This is always a paragraph and can contain inline elements such as label and link.	x	
.fb_errorbox #fb_container p#fb_message label	Contextualized id selector that defines the common properties for the label in the message area of the Uniform Feedback Object - error message. This is always a paragraph and can contain inline elements such as label and link.	x	
.fb_errorbox #fb_controlbox button	Selector for the button that appears in the message area of the Uniform Feedback Object - error message.	x	
.fb_notifybox a, .fb_notifybox a:active, .fb_	Selectors for the default states of links in the Uniform Feedback Objects - error messages and notification messages.	x	

Selector Name	Description	default.css	xdefault.css
errorbox a, .fb_errorbox a:active			
.fb_notifybox a:hover, .fb_errorbox a:hover	Selectors for the state when the pointer rests (hovers) on links in error messages and notification messages.	x	
.calendar	Class selector for presentational properties of the calendar control.	x	
.calendartitle	Class selector for presentational properties of the calendar title.	x	
.dayname, .defaultday, .offday, .currentday, .selectedday	Class selectors for presentational properties of different types of days in the calendar control.	x	
.highlight	Class selector that defines the properties of highlighted items.	x	
.data .highlight, .navigational .highlight, .display .highlight	Group selector that defines the properties of highlighted rows in grids. You can use these selectors to achieve different highlighting effects for each grid type available in AppWorks Platform XForms.	x	
.highlight input.input	Descendant selector that defines the appearance of an input box in a highlighted row. You can use it to specify the appearance of the input control in a highlighted row of the data entry grid.	x	
.ifocus	Class selector that defines the appearance of an input box when focused through mouse action or keyboard tab.	x	
.data .highlight input.ifocus	Class selector that defines the appearance of the input box when highlighted through mouse action or keyboard tab, specifically for the data entry grid. You can extend the same kind of customization to other grid types as well.	x	
input.error, textarea.error	Group selector that defines the state of an input box and a Textarea control when they contain an incorrect value.		x
.selectbox.error	Class selector that defines the state of a Select control when it contains an incorrect value.		x
.selectbox.error input	Descendant selector that defines the state of a Check control when it contains an incorrect value. Do not customize it.		x
.selectbox.disabled input	Descendant selector that defines the disabled state of an input in a Select control. Do not customize it.		x
.left_align	Class selector that aligns elements to the left-hand side.		x
.right_align	Class selector that aligns elements to the right-hand side.		x
.center_align	Class selector that aligns elements to the center.		x
.error .right_align	Group selector applicable to situations where a right-aligned element contains an incorrect value.		x

Selector Name	Description	default.css	xdefault.css
.removed	Class selector that hides an element by setting its display property to none.		x

Caution: When customizing style sheets, ensure that you maintain the cascade order of selectors to prevent errors. If you need to replace the default.css with a custom version, it must contain all selectors to ensure the proper working of the AppWorks Platform Interface.

Keyboard shortcuts (XForms)

The following table lists the common keyboard shortcuts available in AppWorks Platform XForms.

Keyboard shortcut keys	Description
CTRL+Z	Undo
CTRL+Y	Redo
CTRL+A	Select all
CTRL+C	Copy
CTRL+X or Shift+Delete	Cut
CTRL+V or Shift+Insert	Paste
CTRL+S	Save
F7	Open Script Editor

The following table lists the keyboard shortcuts available specifically for the Script Editor.

Keyboard shortcut keys	Description
CTRL+Spacebar	Displays options available in Intellisense.
.	Displays the various properties and methods associated with the current object.
(Displays attributes applicable to a method.
CTRL+I	Displays attribute information for a method.
Esc	Closes the prompt that displays object models.

The following table lists the keyboard shortcuts available specifically for the TabPage control.

Keyboard shortcut keys	Description
CTRL + End	Displays the last tab.
CTRL + Home	Displays the first tab.
CTRL + Left	Displays the previous tab in a horizontally-oriented tab group.
CTRL + Right	Displays the next tab in a horizontally-oriented tab group.
CTRL + Up	Displays the previous tab in a vertically-oriented tab group.
CTRL + Down	Displays the next tab in a vertically-oriented tab group.

The following table lists the keyboard shortcuts available specifically for the Artifact Viewer control.

Keyboard shortcut keys	Description
Arrow keys	Enables navigation among the artifacts.
Alt+down arrow	Selects  to display the menu of a highlighted artifact.
Alt+up arrow	Displays the context-menu of a highlighted artifact.
Menu	Displays the context-menu for an Artifact Viewer.
Enter	Performs the configured double-click action on a highlighted artifact.
Space	Performs the configured single-click action on a highlighted artifact.

DOM explorer

DOM Explorer displays the HTML elements in the XForm designer area into a DOM tree like structure. The DOM Explorer tab can be made visible by selecting the Show DOM Explorer check box in the Form properties. The DOM Explorer tab will be added next to the Design tab. When the design is complex in designer area with many controls or controls within controls, it becomes hectic to select the property sheet of the required control and also re-positioning of the controls. Since the DOM explorer displays all the controls in the designer area into a tree structure with proper hierarchy; property sheet can be opened by clicking on the control name in the DOM tree and the controls also can be dragged and dropped on another control.

Dragging and dropping of a control inside another control takes place only if the control is a container element (for example, Group Control). The changes that are made in the DOM tree as a result of drag and drop will be reflected in the designer area.

Uses:

- DOM Explorer displays the clear hierarchy of the controls in the designer area.
- Property sheet of the overlapped controls in the designer can be opened by clicking on the control in the DOM Explorer.
- Dragging and dropping of the controls inside another control can be done through the DOM Explorer.
- Elements that have free layout will follow the same DOM order irrespective of re-positioning. Hence, pressing the tab button will take the control to the element, which comes next in the DOM order and not to the element that comes next in the display. Therefore, to make it function properly, change the DOM order of the elements per the display, which can be done through drag and drop feature of the DOM Explorer.

Limitations:

- Elements dragged and dropped into a new parent will follow the old property (for example, position, label alignment) irrespective of its new parent.
- Functions that involve pixel calculation in Designer cannot be done through the DOM Explorer as the designer area will be hidden in the DOM Explorer tab.
The examples are as follows:
 - Splitter property sheet is disabled in the DOM Explorer as the options in the splitter property sheet involve pixel calculation.
 - Label width will not be updated.
- New elements will not be created through the DOM Explorer. When you drag and drop a button or an image, a new control box will be created and then it will be inserted in the Designer. This is not possible through the DOM Explorer.
- In the Form designer, when a UI is generated from a Web service or a schema fragment, the DOM Explorer does not display it the first time. Close and reopen the form to view such a UI in the DOM Explorer.

Related API

- **isDraggable** - Indicates if the control can be dragged.
- **canContain** - Checks if the parent control can contain the dragged item (for example, when the input is dragged over group box, return true, and when the input is dragged over the control box, return false).
- **onBeforeInsertAction** - Checks if the dragged item can be placed in the particular parent control and that can be used to perform any specific actions before the insertion (for example, when the button is dragged over the control box, it should return true, and when the button is dragged over group box, it should return false). In the future, when a button is dragged over the group box, a new control box should be created and then true should be returned.
- **getCommonIcon** - Certain elements by default will not have an icon (for example, tab group, element bar, splitter). For these elements, this function returns a common icon.

- **getxfcChild** - Markup is different for different controls. This function obtains the required intentional child (for example, choice box).
- **isExpandable** - Indicates if the control can be expanded to display its children. By default, composite controls and any other controls that does not inherit from the group control will not be expanded. In order to expand the control, this function should be overridden returning true.

Therefore, if a new control with different property or markup needs to be a part of this DOM tree, the APIs must be implemented in its library.

WS-AppServer reference

There are several other tasks that developers can perform using WS-AppServer. At the beginning of documentation, you read about some basic tasks such as creating a Database Metadata, creating a WS-AppServer package, generating Web services on Database Metadata and so on. In this reference section, you will see some additional topics that contain information on:

- [SOAP Requests](#)
- [Event Listeners in WS-AppServer](#)
- [Embedding WS-AppServer Functionality in Applications](#)
- [Handling Database-generated Fields](#)
- [Localizing WS-AppServer Messages](#)
- [Managing Transactions](#)
- [Setting Access Control](#)

This is just an indicative list. To know more, view the contents.

Customizing WS-AppServer SOAP faults

A SOAP fault carries error and status information within a SOAP message. At times, there may be a need to modify the SOAP fault so that it becomes comprehensible. WS-AppServer provides a feature that helps customize the SOAP fault messages before they are rendered on the UI. After customization, the SOAP faults are displayed on the UI in the form of user-friendly messages.

WS-AppServer provides an event named `onTransactionError`, which can be called through the explicit event listener `ITransactionErrorListener`. This event substitutes the SOAP fault with the customized message. To enable this, a new class `com.cordys.cpc.bsf.busobject.exception.WSAppsSOAPFault` has been introduced. The new class is similar to the `com.eibus.soap.SOAPFault`, except that it supports localization of the fault string.

Whenever a SOAP fault occurs, WS-AppServer searches for the class that has implemented the `ITransactionErrorListener` event listener. Upon detecting the class, WS-AppServer

invokes the `onTransactionError` API, which updates the old SOAP fault with the message that is stored. Subsequently, this customized message is sent as a SOAP response to the UI.

To understand how the customization works, consider the following example.

Example

You intend to insert a row in the database with `CategoryID`. However, you do not provide `CategoryName`, which is a mandatory field. This results in an exception. In such a case, a normal SOAP fault will occur. However, you can customize the SOAP fault to provide exact information to the user by using the following code.

```
public class MyListener implements ITransactionErrorHandler {
    public void onTransactionError(TransactionErrorEvent event) {
        /*
         * Below example sets a fault code, fault string and an error detail
         * picked up from actual error that has occurred.
         */
        Throwable t = event.getException();
        if (t instanceof BsfUpdateException) {
            BsfUpdateException bue = (BsfUpdateException) t;
            int noOfErrors = bue.getNrErrors(); // get number of errors

            // Following lines illustrate how to fetch the BusObject
            // and the related error message for the first error.
            // User can cast this BusObject and retrieve the specific field
            // values
            // to mention in the new soap fault

            BusObject bo = bue.getObject(0);
            if (bo instanceof Categories) {
                Categories c = (Categories) bo;
                String catName = c.getCategoryName();
                String catID = String.valueOf(c.getCategoryID());

                // construct a new soap fault
                WSAppsSOAPFault fault = new WSAppsSOAPFault(
                    "Application Error",
                    "Application threw an error while updating Categories with
ID: "
                    + catID, null, bue.getErrorDetails(0));
                // Customizing soap fault
                event.setSOAPFault(fault);
            }
        }
    }
}
```

To further understand how the SOAP fault is customized, see the following request that is sent to WS-AppServer.

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Body>
        <UpdateCategories xmlns="http://schemas.cordys.com/Ws-AppServer/any">
            <tuple>
                <new>
                    <Categories>
                        <CategoryID>21</CategoryID>
                        <CategoryName>abc</CategoryName>
                        <Description>xyz</Description>
                        <Picture>PARAMETER</Picture>
                    </Categories>
                </new>
            </tuple>
        </UpdateCategories>
    </SOAP:Body>
</SOAP:Envelope>

```

An attempt to persist incorrect information into the database results in the following customized SOAP fault.



Along with the SOAP fault, it returns the following response (click the SOAP fault to view the response).

```

<data>
    <SOAP:Fault xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <faultcode
xmlns:ns0="http://schemas.xmlsoap.org/soap/envelope/">ns0:Server</faultcode>
        <faultstring xml:lang="en-US">Database update failed</faultstring>
        <faultactor>http://schemas.cordys.com/Ws-AppServer/any</faultactor>
        <detail>
            <cordys:FaultDetails
xmlns:cordys="http://schemas.cordys.com/General/1.0/">
                <cordys:MessageCode
xmlns:cordys="http://schemas.cordys.com/General/1.0/">Cordys.DBConnectors.Messages.d
atabaseUpdateError</cordys:MessageCode>
                <cordys:LocalizableMessage/>
                <UpdateCategories
xmlns="http://schemas.cordys.com/Ws-AppServer/any"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
                    <tuple

```

```

        xmlns="http://schemas.cordys.com/Ws-AppServer/any"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <new>
        <Categories>
            <CategoryID>21</CategoryID>
            <CategoryName>abc</CategoryName>
            <Description>xyz</Description>
            <Picture>PARAMETER</Picture>
        </Categories>
    </new>
    <error TYPE="Enumeration">
        <elem>pIRowsetChange--&gt;;InsertRow failed failed:
HRESULT: 0x80040e21 (DB_E_ERRORS_OCCURRED)</elem>
        <elem>HRESULT: 0X80040E21 (DB_E_ERRORS_OCCURRED)</elem>
        <elem>Source : Microsoft OLE DB Provider for SQL Server
</elem>
        <elem>Error message: Multiple-step OLE DB operation
generated errors. Check each OLE DB status value, if available. No work was
done.</elem>
        <elem>rowstatus CategoryID DBSTATUS_E_
PERMISSIONDENIED</elem>
        <elem>rowstatus CategoryName DBSTATUS_E_
UNAVAILABLE</elem>
    </error>
    </tuple>
</UpdateCategories>
</cordys:FaultDetails>
</detail>
</SOAP:Fault>
</data>

```

If multiple event listeners are registered, and if the SOAP fault is customized in each of these listener implementations, then the recently updated SOAP fault last executed setSOAPFault() will be retained in the SOAP response. The customized SOAP fault will also appear while using microflows in Business Process Models.

Event listeners in WS-AppServer

The following table contains a list of all the event listeners along with the implicit and explicit methods required to use them.

Action Performed	Name of the Listener (Object Level)	Name of the Listener (Attribute Level)	Context in which the Listener is Used	Implicit	Explicit
Initialize	onInitialize	IObjectInitializeListener	onInitialize_<attribute name>	IAttribute Initialize Listener	When an object or an attribute is initialized
Apply	onConstraint	IObject	onConstraint_	IAttribute	When

Action Performed	Name of the Listener (Object Level)	Name of the Listener (Attribute Level)	Context in which the Listener is Used	Implicit	Explicit
Constraints		Constraint Listener Note: To apply and validate constraints on multiple objects (in a transaction), use ITransaction Constraint Listener	<attribute name> Note: At the attribute level, this constraint event will not be fired on delete request.	Constraint Listener	validation constraints are applied on objects and attributes, and violations if any are reported
Check Availability	-	-	onAvailability_<attribute name>	IAttribute Availability Listener	When an attribute's availability is set
Define Access Control	onAccess	IObjectAccess Listener	onAccess_<attribute name>	IAttributeAccess Listener	When the access level of an object or an attribute is set at the server-level
Set Display (UI)	onDisplay	IObjectDisplay Listener	onDisplay_<attribute name>	IAttributeDisplay Listener	When the display of an object or an attribute on the UI is determined based on the access level
Set Values	-	-	onValues_<attribute name>	IAttributeValues Listener	When values (options) for an attribute are set
After Object Load	onAfterLoad	IAfterLoad Listener		-	When a certain activity is performed after loading an object
Before Attribute Change	-	-	onBeforeChange_<attribute name>	IBeforeAttribute ChangeListener	When a certain activity is performed before an attribute's value changes
After Attribute Change	-	-	onAfterChange_<attribute name>	IAfterAttribute ChangeListener	When a certain activity is performed after an

Action Performed	Name of the Listener (Object Level)	Name of the Listener (Attribute Level)	Context in which the Listener is Used	Implicit	Explicit
					attribute's value changes
Before Object Insert	onBeforeInsert	IBeforeInsert Listener	-	-	When a certain activity is performed before inserting an object
Upon Object Insert	onInsert	-	-	-	When a certain activity is performed immediately after an object is inserted
After Object Insert	onAfterInsert	IAfterInsert Listener	-	-	When a certain activity is performed after inserting an object
Before Object Update	onBeforeUpdate	IBeforeUpdate Listener	-	-	When a certain activity is performed before updating an object
Upon Object Update	onUpdate	-	-	-	When a certain activity is performed immediately after an object is updated
After Object Update	onAfterUpdate	IAfterUpdate Listener	-	-	When a certain activity is performed after updating an object
Before Object Delete	onBeforeDelete	IBeforeDelete Listener	-	-	When a certain activity is performed before deleting an object
Upon Object Delete	onDelete	-	-	-	When a certain activity is performed immediately after an object is deleted

Action Performed	Name of the Listener (Object Level)	Name of the Listener (Attribute Level)	Context in which the Listener is Used	Implicit	Explicit
After Object Delete	onAfterDelete	IAfterDelete Listener	-	-	When a certain activity is performed after deleting an object
Before Transaction Commit	onBeforeCommit	IBeforeCommit Listener	-	-	When a certain activity is performed before committing a transaction
After Transaction Commit	onAfterCommit	IAfterCommit Listener	-	-	When a certain activity is performed after committing a transaction Note: For Container Managed Transactions, this event is triggered after WS-AppServer commit is invoked and before the actual Java transaction is committed.
Before Transaction Abort	onBeforeAbort	IBeforeAbort Listener	-	-	When a certain activity is performed before aborting a transaction
After Transaction Abort	onAfterAbort	IAfterAbort Listener	-	-	When a certain activity is performed after aborting a transaction Note: For Container Managed Transactions, this event is triggered after Ws-AppServer abort is invoked and

Action Performed	Name of the Listener (Object Level)	Name of the Listener (Attribute Level)	Context in which the Listener is Used	Implicit	Explicit
					before the actual Java transaction is aborted.
When WS-AppServer Processor starts or stops	onOpenConnector onCloseConnector	IOnBsf Connector	-	-	When a certain activity is performed when the WS-AppServer processor starts or stops
Evaluate constraints on transaction objects	onTransaction Constraint	ITransaction Constraint Listener	-	-	When objects have to be evaluated for any constraint violation
Retrieve all objects participating in a transaction	onReceive	IUpdateBody Block Listener	-	-	When there is a need to change the behaviour of objects participating in a transaction to fulfill business requirements
After Transaction Completion	onAfterTransaction Completion	ITransaction Completion Object Listener	-	-	When a certain activity is performed after committing or aborting the transaction

Types of event listeners

There are two types of event listeners:

- Implicit Listeners
- Explicit Listeners

Implicit listeners

When you implement a Java method (logic) in an extension class of an object, it becomes an implicit listener. This implicit listener contains all the required information to act on a particular event.

For example, to perform some action such as checking constraints on an object before it is inserted into the database, use `onBeforeInsert` event listener. To invoke the listener implicitly, add a listener method such as `onBeforeInsert` to the extension class and add the code to implement the functionality.

An implicit attribute-level event listener needs to follow the naming convention as shown.

```
on<listener name>_<attribute name>
```

For example, `onInitialize_City`, `onAccess_Name`. The name of the attribute is case-sensitive. Therefore, ensure that it is identical to the actual attribute defined in the object.

Explicit listeners

When you create a class (outside the extension class) containing only the Java interface (no implementation), it becomes an explicit listener. Explicit listeners must be registered before they are invoked. For this purpose, WS-AppServer provides a class called `Listeners` , that contains several methods to register such explicit listeners.

Even here, to check constraint on an object before it is inserted into the database, use the `onBeforeInsert` event listener. However, to invoke the listener explicitly, you need to register a class that implements the `IBeforeInsertListener` interface.

Use explicit listeners in situations where you want to use event listeners outside the scope of the extension class.

Difference between Implicit and Explicit listeners

Implicit Listeners	Explicit Listeners
Allows implementation of a single event listener	Allows implementation of multiple event listeners in a single class
Is internal to an extension class	Requires registration with the <code>Listeners</code> class before being applied
Restricted to the logic within the class	Accommodates implementation of additional logic

Event listener precedence

If your application code contains implicit listeners and explicit listeners, then the implicit listeners are called first, followed by the explicit listeners.

Handling database-generated fields

Certain databases support auto-generation of field values, where the auto-generated value is a result of some other record insertion into the database. For example, when an Order row is inserted into the Order table, the OrderID, which is a unique sequence number, is generated by the database. This field containing the OrderID is a database-generated field.

Sequencing object insertion in WS-AppServer

When writing to a database, it is useful to identify the object with a unique number. To enable this, you need to have a process in place that increments the count of objects each time you insert an object into the database. Each request will possess a unique identifier. However, the situation will vary depending upon whether the request is for inserting an object or for inserting multiple objects.

To insert a single object:

1. Sequence each single object (per request) in the database by placing the appropriate command in the `OnBeforeInsert` event listener in the extension class.
2. Use the following sample code for this purpose.

```
public void onBeforeInsert()
{
    String queryText = "select max(adjustment_order) as maxNr" + "from stock_
adjustment " + "where center = :center";
    QueryObject query = new QueryObject(queryText);
    query.addParameter("center", "stock_adjustment.center", QueryObject.PARAM_INT, new
Short(this.getCenter()));
    AnonymousBusObject aBusObj = (AnonymousBusObject)query.getObject();
    // And add 1 for the new record
    this.setAdjustment_Order(aBusObj.getIntProperty("maxNr")+1);
}
```

To insert multiple objects:

If the request contains multiple objects to be inserted, as a result of optimistic transaction mechanism, each object may get the same identifying sequence number. To prevent this problem, perform the following steps.

1. Retrieve the maximum value of the sequence number from the database (maxDB)
2. Retrieve the maximum number of objects in the current transaction (maxTrans)
3. Retrieve the maximum value of maxDB and maxTrans (maxSeqNo)
4. Increment maxSeqNo by 1 to get the new sequence number.

For example:

```
/***
 * Gets the next available Stock Adjustment Order number for the given center.
 *
 * @param center
 *          The center
 * @return the next stock adjustment order number
 */
```

```

private int getNextAdjustmentOrderNumber(short center)
{
    // Check if there is already a StockAdjustment in
    // the transaction (i.e. the BusObjectManager)
    int transactionMaxOrderNo = 0;
    short sCenter = this.getCenter();
    BusObjectIterator objects = BSF.getObjectManager().getUpdatedObjects();

    while ( objects.hasMoreElements() )
    {
        BusObject busObject = objects.nextElement();
        if ( busObject instanceof StockAdjustment )
        {
            StockAdjustment stckAdj = (StockAdjustment)busObject;
            if ( stckAdj != this && sCenter == stckAdj.getCenter() )
            {
                int orderNo = stckAdj.getAdjustment_order();
                if ( orderNo > transactionMaxOrderNo )
                {
                    transactionMaxOrderNo = orderNo;
                }
            }
        }
    }

    // Get max seqno from DB
    int dbMaxOrderNo = 0;
    String queryText = "select max(adjustment_order) as maxNr" + "from stock_
adjustment " + "where center = :center";
    QueryObject query = new QueryObject(queryText);
    query.addParameter("center", "stock_adjustment.center", QueryObject.PARAM_INT, new
Short(this.getCenter()));
    AnonymousBusObject aBusObj = (AnonymousBusObject)query.getObject();

    // And add 1 for the new record
    dbMaxOrderNo = aBusObj.getIntProperty("maxNr");
    if ( transactionMaxOrderNo > dbMaxOrderNo )
    {
        return transactionMaxOrderNo+1;
    }
    else
    {
        return dbMaxOrderNo+1;
    }
}

```

To insert single or multiple objects:

If there are multiple requests attempting to insert one or more objects at the same time, there is a possibility that some objects would get the same identity sequence number, resulting in a primary key error. To avoid such a problem, take the following preventive measures.

- Use the automatic retry feature that will inevitably re-calculate the numbers.
- Process a single insert operation at a time.
- Use a different algorithm to calculate the sequence number (for example, you can have a different database table to store the last-used sequence number).

Attribute value integrity

Transactions within WS-AppServer are optimistic transactions. If a database-generated field needs to be created, it should always be after an object is committed to the database. Therefore, any logic that creates a database-generated field should be placed in the `OnAfterCommit` event listener.

Since the database-generated value is available only after commit, special provision should be made to use this value for other objects in the same transaction. A typical example is the use of a Foreign Key (FK) to associate objects to each other. For example, in an Order/OrderLine situation, OrderLines are associated to an Order object by sharing the same OrderID value (in an RDBMS scenario, an OrderLine has an FK to the Order's Primary Key (PK)). To handle such situations, WS-AppServer establishes dependency among certain attributes. This is called attribute value integrity.

Currently, you have to write the code to define the integrity dependency. Instead, WS-AppServer provides you with an API (of `BusObject`) that can be invoked on the object dependent upon the source object.

```
public void setAttributeValueIntegrity
(
    BusObject sourceObject,
    String sourceAttributeName,
    String attributeName
)
```

Referring to the Order/OrderLines example, the method is called on the OrderLine object because it is dependent upon the source object Order. As per the example, the `sourceAttributeName` is the attribute in the `sourceObject` (PK), the `attributeName` is the attribute name in the depending object (FK), and the `sourceObject` is an Order object. If attribute value integrity is required for multiple attributes, you can use the following API.

```
public void setAttributeValueIntegrity
(
    BusObject sourceObject,
    String[] sourceAttributeNames,
    String[] attributeNames
)
```

This functionality is based on the Associated Insert feature offered by the XQY Library.

JMX functionality in WS-AppServer

The JMX functionality in WS-AppServer helps System Administrators to remotely monitor the WS-AppServer Service configuration and various managed components of different WS-AppServer-based applications. One of the key performance counters provided by JMX in WS-AppServer is the number of active transactions that can be monitored.

This functionality can be used in two types of applications:

- WS-AppServer applications
- Applications that use WS-AppServer in embedded mode

In WS-AppServer applications

WS-AppServer, by default, creates a sub-component called 'Applications'. A new managed component can be created under 'Applications' and its configuration can be defined by the developer. You can create as many managed components as required for that application.

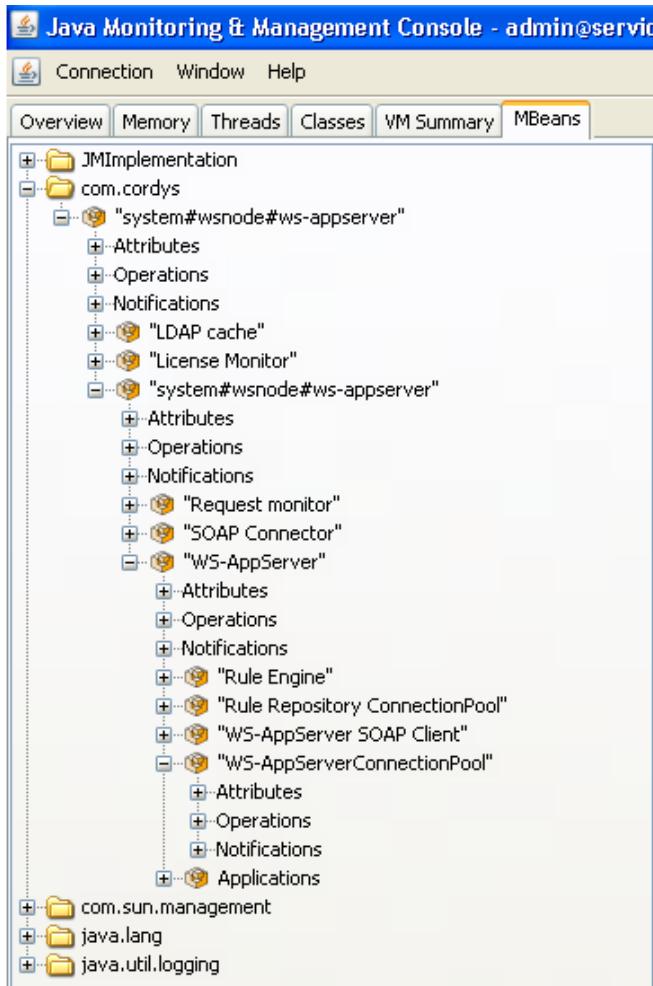
To achieve this, use the following method in the BsfConnector class:

```
public IManagedComponent createManagedApplicationComponent(String type, String description, Object componentImpl)
```

When the components are created, the JMX console tree appears similar to the sample as shown.

```
WS-AppServer managed component
    ->WS-AppServerConnectionPool component
        ->Applications component
            ->FirstApplication (Created by the application developer)
            ->SecondApplication (Created by the application
developer)
```

The actual console would appear as shown:



In Applications Using WS-AppServer in embedded mode

If WS-AppServer is being used in embedded-mode by any other application connector code, then a managed sub-component for WS-AppServer can be created in embedded mode and specified in the WS-AppServer configuration. This arrangement exposes the JMX functionality of WS-AppServer to the management console through the application connector that embeds WS-AppServer. Additionally, a name for the DBConnectionpool can be specified so that WS-AppServer can create a connection pool component with that name under the existing managed sub-component.

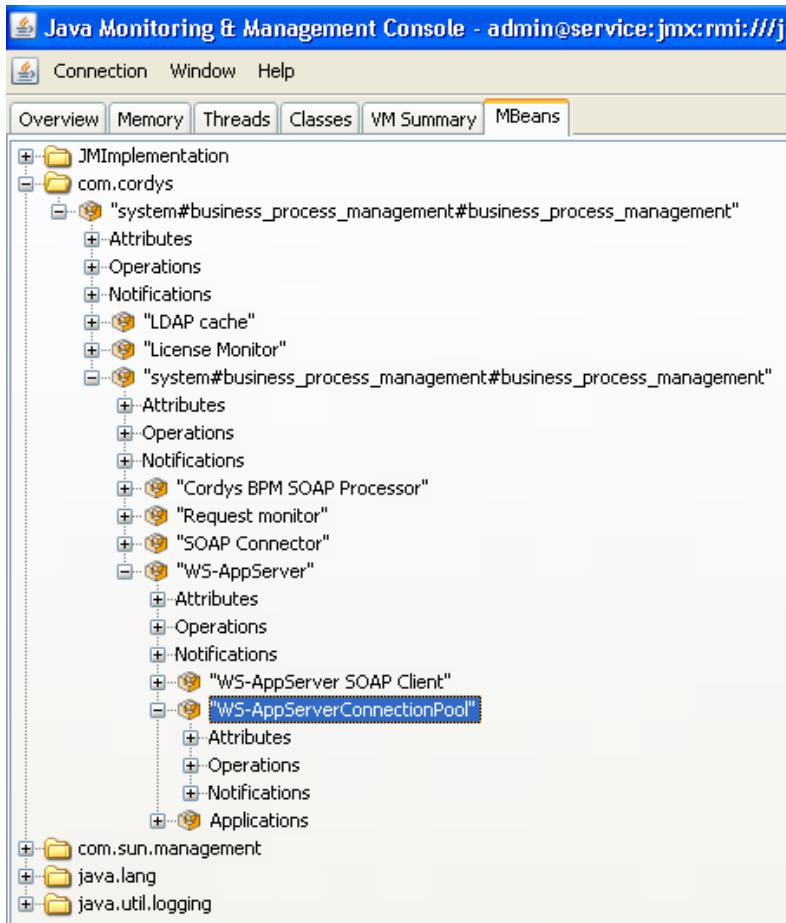
To achieve this, use the following constructor in the Config class:

```
Config(IManagedComponent wsAppServerManagedComponent, String connectionPoolName, int iConfig, boolean setProperties)
```

When the connection pool component is created, the JMX console tree appears similar to the sample shown.

Any application connector managed component
 ->WS-AppServer managed sub-component
 ->Connection pool component

The actual console would appear as shown in the figure.



Localizing WS-AppServer messages

Applications that are developed and deployed using WS-AppServer are equipped with the functionality of reporting the messages to the end-users in their local languages. Currently, WS-AppServer supports localizing messages while using the following classes and methods:

- `BsfConstraintViolationException`
- `BsfApplicationRuntimeException`
- `ConstraintEvent.addError`
- `AccessEvent.setAccess`
- `ValuesEvent.addValue`

To localize WS-AppServer messages:

1. Create a message definition file (XML document) that would contain the required messages. A sample is shown.

```
<MessageBundle id="mycomp.myapp.LocalizedMessages">
    <Message id="CustomerDoesNotExist">
        <MessageText>Customer {0} does not exist</MessageText>
        <Description>The specified customer does not exist, so the reference is invalid.</Description>
        <Annotations>
            <DocumentationURL/>
        </Annotations>
    </Message>
    <Message id="ValueInRange">
        <MessageText>The specified value must be between {0} and {1}</MessageText>
        <Description>To specify that a value is out of range</Description>
        <Annotations>
            <DocumentationURL/>
        </Annotations>
    </Message>
</MessageBundle>
```

Ensure the following while creating the message definition file:

- The name of the file must be of the following format: <id of the localization bundle>#MessageBundle#.xml (which is mycomp.myapp.LocalizedMessages#MessageBundle#.xml in the sample above). After the file is synchronized and published, the content is published to <AppWorks Platform_installdir>\localization folder as mycomp.myapp.LocalizedMessages.xml file
 - To retrieve the messages at runtime, the message definition file must be located in <AppWorks Platform_installdir>\localization. The default messages are located in <AppWorks Platform_installdir>\localization\mycomp.myapp.LocalizedMessages.xml.
 - For a specific locale, the file name has a locale suffix, such as _fr_CA for French in Canada. Thus, the file location for this locale would be <AppWorks Platform_installdir>\localization\mycomp.myapp.LocalizedMessages_fr_CA.xml. The actual locale for a user ideally depends on the browser's locale settings. However, it is possible to define a default locale setting (com.eibus.defaultLocale=nl), while installing AppWorks Platform. This is stored as a property in wcp.properties file. If the locale settings are set to the default value, then the required message file would be mycomp.myapp.LocalizedMessages_nl.xml.
 - Do not include the Description and Annotation fields in the locale-specific file.
2. Generate the Java class from the message definition file, using a command similar to the following.

```
C:\>java com.eibus.tools.internal.MessageGenerator
Usage: MessageGenerator <Messages Definitions Filename> <Fully Qualified Class Name>
[<Target Directory>]

Usage example: java com.eibus.tools.internal.MessageGenerator
D:\Cordys\localization\mycomp.myapp.LocalizedMessages.xml mycomp.myapp.MessageBundle
D:\Cordys

The MessageBundle.java will be created in the directory 'D:\Cordys'
```

The following output is derived.

```
package com.mycomp.myapp;

import com.eibus.localization.message.Message;
import com.eibus.localization.message.MessageSet;

/**
 * This code is generated by running
 * com.eibus.localization.message.MessageGenerator.
 *
 * Generated by :user Input file :mycomp.myapp.LocalizedMessages.xml Generated
 * at :Fri Mar 10 15:01:33 CET 2006
 */

public class LocalizedMessages {
    public static final MessageSet MESSAGE_SET = MessageSet
        .getMessageSet("mycomp.myapp.LocalizedMessages");

    /** Customer {0} does not exist */
    public static final Message CUSTOMER_DOES_NOT_EXIST = MESSAGE_SET
        .getMessage("CustomerDoesNotExist");

    /** The specified value must be between {0} and {1} */
    public static final Message VALUE_IN_RANGE = MESSAGE_SET
        .getMessage("ValueInRange");
}
```

Java class needs to be generated only from the base message definition file and not from the language-specific definitions.

3. In the application code, use this generated code as shown in the following sample.

```
/** 
 * Constraint handler for CustomerID.
 */
public void onConstraint_CustomerID(AttributeConstraintEvent constrEvent)
{
```

```

// check the available Customers whether the Customer exists
// the new CustomerID is passed via the context
String customerID = (String)constrEvent.getValue();
if ( Customers.getCustomersObject(customerID) == null )
{
    constrEvent.addError(this, ATTR_CustomerID,
        LocalizedMessages.CUSTOMER_DOES_NOT_EXIST,
        new Object[]{customerID});
// the traditional way is
// constrEvent.addError(this, ATTR_CustomerID,
// "Customer " + customerID + " does not exist");
}
}

```

The message (`com.eibus.localization.message.Message`) implements the interface `ILocalizableString` (`com.eibus.localization.ILocalizableString`). You can define parameters in the message (`messageParams`). However, these parameters are substituted at the moment the actual message is constructed.

WS-AppServer messages are localized to a particular language.

Mapping of BusObjects in WS-AppServer

WS-AppServer exchanges data in the form of BusObjects through any of the following representations:

- XML - Representation of the BusObject is through a SOAP request and response
- DB - Representation of the BusObject stored in the database (RDBMS or other persistent stores)
- Java - Representation of the BusObject in a Java wrapper around an XML document

During data exchange, a BusObject may pass through a combination of these representations (XML to Java, Java to DB, and so on) before it is finally persisted. The following table provides a quick reference to the various combinations that are used by WS-AppServer.

Combination	Description
DB to Java	Information to be passed from the database to Java either is generated in the base class or is implemented by the developer in the extension class. To transform DB information into a Java class, XQY takes a query as input, and produces the output as XML documents. The query object creates the BusObject based on the XQY output.
Java to DB	The Java class is mapped to XML so that it is understood by XQY. XQY then transfers the content of the XML to the relevant database tables and updates them. The Java to DB mapping usually maps custom classes to standard classes, and invokes insert, update, or delete operations on

Combination	Description
	those standard classes that persist the BusObject to the database.
Java to XML	When a Java object is used in a SOAP response, WS-AppServer refers the BusObject in store, retrieves its XML representation, and sends it in the response.
XML to Java	When an object is received through a SOAP request, WS-AppServer creates a BusObject and places the XML of the SOAP request in it. While doing this, WS-AppServer reads the class registry and maps the incoming XML to the corresponding Java class.

Mapping names to database

If an object of `typeStateBusObject` is inserted or updated into the database, then the name of the class is mapped to the name of the table. Similarly, the name of an attribute is mapped to the name of a column in the table. By default, the names are exactly the same.

WS-AppServer uses a mapping strategy to map a class name to a table name and map an attribute name to a column name. A mapping strategy is implemented in an object that implements `IMapClassStrategy` and `IMapAttributeStrategy`.

WS-AppServer has a number of standard mapping strategies, both for class mapping and attribute mapping.

Mapping Level	Types	Description
Class to Table	TrivialStrategy	Attribute names are not mapped. However, class to table name mapping can be specified. This is the default mapping strategy for <code>StateBusObject</code> .
	DefaultStrategy	Maps class name to table name. This strategy can be configured using <code>IMapAttributeStrategy</code> or <code>DefaultAttributeStrategy</code>
Attribute to Column	NameMappingStrategy	Maps attribute name to column name. This strategy is configured with attribute-column name pairs and is used in situations where column names differ from attribute names.
	DefaultAttributeStrategy	Maps attribute name to column name. This strategy uses <code>AttributeInfo</code> to determine the mapping.

Besides these standard mapping strategies, you can also implement your custom mapping strategy.

Custom mapping

A custom mapping can be defined by implementing a custom mapping strategy, probably in combination with one of the standard strategies. To make use of custom mapping, the custom mapping strategy must be registered with the class. This registration is done in the method `customizeClassInfo` as shown in the code.

```

/**
 * Define the mapping strategy
 */
public static void customizeClassInfo(CustClassInfo ci)
{
    ci.setMappingStrategy(new MyMappingStrategy(ci));
}

```

The following Java codes provide examples of how custom mapping is implemented.

MapOrderDetailsBase

```

package com.cordys.cpc.wsappsserver.sources.application.dbmap;

import com.cordys.cpc.bsf.busobject.BusObjectConfig;
import com.cordys.cpc.bsf.busobject.BusObjectIterator;
import com.cordys.cpc.bsf.busobject.QueryObject;
import com.cordys.cpc.bsf.classinfo.ClassInfo;
import com.cordys.cpc.bsf.classinfo.RelationInfo_FK;
import com.cordys.cpc.bsf.query.Cursor;
import com.cordys.cpc.wsappsserver.sources.application.Products;

public abstract class MapOrderDetailsBase extends
com.cordys.cpc.bsf.busobject.StateBusObject
{
    // tags used in the XML document
    public final static String ATTR_MapOrderID = "MapOrderID";
    public final static String ATTR_MapProductID = "MapProductID";
    public final static String ATTR_MapUnitPrice = "MapUnitPrice";
    public final static String ATTR_MapQuantity = "MapQuantity";
    public final static String ATTR_MapDiscount = "MapDiscount";
    private final static String REL_OrderIDObject = "FK:OrderDetails[OrderID]:Orders
[OrderID]";
    private final static String REL_ProductIDObject = "FK:OrderDetails
[ProductID]:Products[ProductID]";
}

```

```

private static ClassInfo s_classInfo = null;
public static ClassInfo _getClassInfo()
{
    if ( s_classInfo == null )
    {
        s_classInfo = newClassInfo(MapOrderDetails.class);
        s_classInfo.setUIDElements(new String[]{ATTR_MapOrderID, ATTR_
MapProductID});
        {
            // relation OrderIDObject (FK:OrderDetails[OrderID]:Orders[OrderID])
            RelationInfo_FK ri = new RelationInfo_FK(REL_OrderIDObject);
            ri.setName("OrderIDObject");
            ri.setLocalAttributes(new String[]{"MapOrderID"});
            ri.setLocalIsPK(false);
            ri.setRelatedClassName("application.Orders");
            ri.setRelatedAttributes(new String[]{"MapOrderID"});
            ri.setRelatedIdentifier("FK:Orders[OrderID]:OrderDetails[OrderID]");
            ri.setLoadMethod("loadMapOrderIDObject");
            s_classInfo.addRelationInfo(ri);
        }
        {
            // relation ProductIDObject (FK:OrderDetails[ProductID]:Products
[ProductID])
            RelationInfo_FK ri = new RelationInfo_FK(REL_ProductIDObject);
            ri.setName("ProductIDObject");
            ri.setLocalAttributes(new String[]{"ProductID"});
            ri.setLocalIsPK(false);
            ri.setRelatedClassName("application.Products");
            ri.setRelatedAttributes(new String[]{"ProductID"});
            ri.setRelatedIdentifier("FK:Products[ProductID]:OrderDetails
[ProductID]");
            ri.setLoadMethod("loadMapProductIDObject");
            s_classInfo.addRelationInfo(ri);
        }
    }
    return s_classInfo;
}

protected static final String[][] DB_MAPPING = new String[][]{
    {"OrderID", "MapOrderID"},
```

```
MapUnitPrice(long value)
{
    setProperty(ATTR_MapUnitPrice, value, 0);
}

public short getMapQuantity()
{
    return getShortProperty(ATTR_MapQuantity);
}

public void setMapQuantity(short value)
{
    setProperty(ATTR_MapQuantity, value, 0);
}

public float getMapDiscount()
{
    return getFloatProperty(ATTR_MapDiscount);
}

public void setMapDiscount(float value)
{
    setProperty(ATTR_MapDiscount, value, 0);
}

public MapOrders loadMapOrderIDObject()
{
    String queryText = "select * from Orders where OrderID = :OrderID";
    QueryObject query = new QueryObject(queryText);
    query.addParameter("OrderID", "Orders.OrderID", QueryObject.PARAM_INT, new
Integer(getMapOrderID()));
    query.setResultClass(MapOrders.class);
    return (MapOrders)query.getObject();
}

public MapOrders getMapOrderIDObject()
{
    return (MapOrders)getSingleRelationObject(REL_OrderIDObject);
}

public void setMapOrderIDObject(MapOrders a.Orders)
{
    if (a.Orders == null)
    {
        this.setNull("OrderID");
```

```

        or)
    {
        String queryText = "select OrderID as MapOrderID, ProductID as MapProductID,
        UnitPrice as MapUnitPrice, Quantity as MapQuantity, Discount as MapDiscount from
        OrderDetails where (OrderID < :OrderID) or (OrderID = :OrderID and ProductID <
        :ProductID) order by OrderID desc, ProductID desc";
        QueryObject query = new QueryObject(queryText);
        query.addParameter("OrderID", "OrderDetails.OrderID", QueryObject.PARAM_INT,
        new Integer(OrderID));
        query.addParameter("ProductID", "OrderDetails.ProductID", QueryObject.PARAM_
        INT, new Integer(ProductID));
        query.setResultClass(MapOrderDetails.class);
        query.setCursor(cursor);
        return query.getObjects();
    }
}

```

MapOrderDetails

```

package com.cordys.cpc.wsappsserver.sources.application.dbmap;

import com.cordys.cpc.bsf.busobject.BusObjectConfig;
import com.cordys.cpc.bsf.classinfo.ClassInfo;
import com.cordys.cpc.bsf.query.dbmap.NameMappingStrategy;

public class MapOrderDetails extends MapOrderDetailsBase
{
    public MapOrderDetails()
    {
        this((BusObjectConfig)null);
    }

    public MapOrderDetails(BusObjectConfig config)
    {
        super(config);
    }

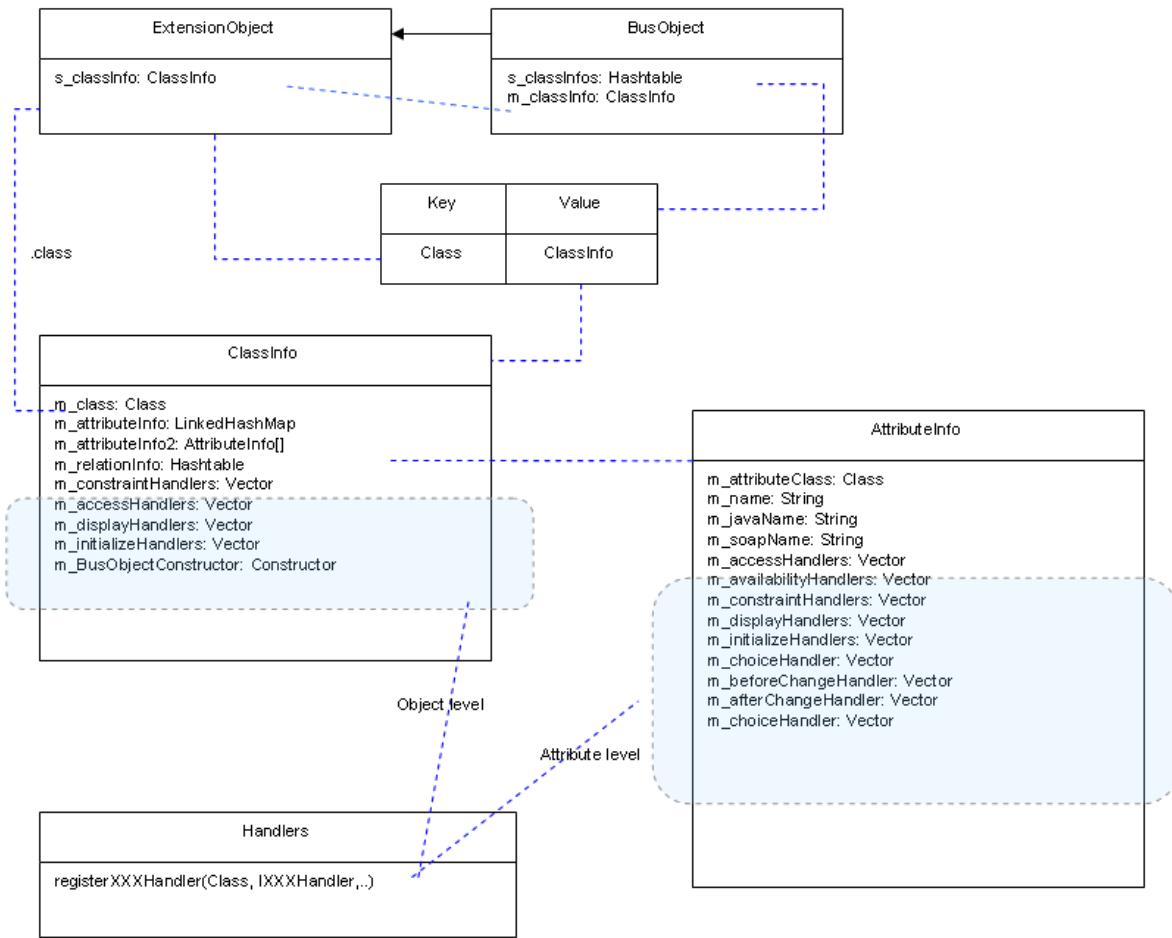
    /**
     * Add DBMapper info to the ClassInfo
     */
    public static void customizeClassInfo(ClassInfo ci)
    {
        ci.setMappingStrategy(
            new NameMappingStrategy(
                "OrderDetails", // the table name
                "OrderDetails", // the class name
                DB_MAPPING)); // column/attribute mapping
    }
}

```

```
}
```

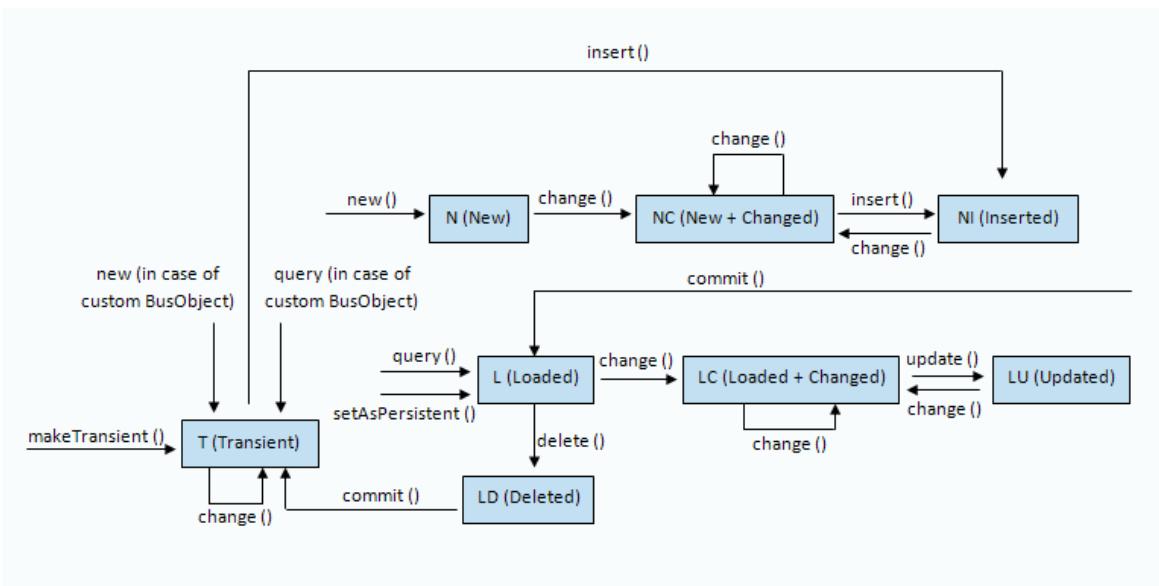
Structure of BusObject

The following image represents the structure of a BusObject.



States of a BusObject

The following flow diagram portrays the different states of a BusObject.



The various states are explained in the following table.

State	Description
N (new)	Indicates that an object has just been created.
NI (inserted)	Indicates that an insert () operation has been performed on the object. When the object is changed, it attains the NC (new + changed) state. When the object is committed, it attains the L (loaded) state.
NC (new + changed)	Indicates that an object, which has just been created, has been changed.
T (transient)	Indicates that the object is independent of any transaction. In such a state, committing a transaction will not affect the object. If an insert () operation is performed on an object in this state, the object attains the NI (inserted) state.
L (loaded)	Indicates that the object has just been loaded from the database into a transaction (for example, using a query). It also indicates that currently the object content has not undergone any change. An object attains this state as soon as it is inserted or updated in the database.
LC (loaded + changed)	Indicates that the contents of the object are changed.
LU (updated)	Indicates that an update () operation has been performed on the object. As soon as one of the attributes of the object is updated, it attains the LC state.
LD (deleted)	Indicates that a delete () operation has been performed on the object.

Managing transactions using WS-AppServer

A transaction could involve insert, update, or delete operation pertaining to a single or multiple database. At times, it is difficult to accommodate several operations in a single transaction, simultaneously. In the absence of a robust transaction management system, the transactions could give in to the load and abort mid-way.

Using WS-AppServer, you can develop business logic pertaining to your application that handles transactions intelligently and effectively, irrespective of the load that is undertaken. WS-AppServer provides a transaction management framework that ensures smooth committing of all transactions.

The transaction management framework of WS-AppServer contains methods that effectively start, commit, and abort transactions depending upon the logic that is provided. This flexibility in handling transactions makes processing multiple database operations in one transaction possible. If any of the operations fail, the entire transaction is rolled back. The transaction is committed only if all the operations succeed.

Handling distributed transactions

In the context of transaction management, WS-AppServer leverages the Reliable Messaging capabilities provided by AppWorks Platform to support Distributed Transactions, where insert, update, or delete operation is performed on two or more data sources.

Distributed Transactions, managed by a Transaction Manager, are helpful when applications must update data in a distributed setup. The Transaction Manager initiates a transaction at a higher level to track the sequential transactions that must be completed before the final commit command is issued. If one of the sequential transactions fails, the entire transaction is rolled back.

This logic is available in the form of annotations that must be added to the Java classes in WS-AppServer. This setup ensures that even in a distributed setup, transaction integrity is maintained and data losses are avoided. To use this feature, WS-AppServer provides annotations that can be used in the WS-AppServer classes.

There are two types of annotations:

- **DistributedTransaction**- Used to annotate BusObject class and the related setter methods
- **StartDistributedTransaction** - Used for BSFJavaCall methods

For example, assume you want to initiate a Distributed Transaction for any insert or delete operation that takes place on the Region class and its attributes. Simultaneously, you want to initiate a Distributed Transaction for the update operation if it is carried out only on the `RegionDescription` attribute of the Region class. In this case, use the `DistributedTransaction` annotation in the Extension class (Region) of the generated Java code as displayed in the following code.

```

@DistributedTransaction({OperationType.INSERT, OperationType.DELETE})
public class Region extends RegionBase
{
    public Region()
    {
        this((BusObjectConfig)null);
    }
    public Region(BusObjectConfig config)
    {
        super(config);
    }
    @Override @DistributedTransaction({OperationType.UPDATE})
    public void setRegionDescription(String value)
    {
        super.setRegionDescription(value);
    }
}

```

The annotations are defined on the class and the `setRegionDescription` method. This results in the following output.

- When Region object is part of the transaction and is being inserted or deleted, the Distributed Transaction is initiated.
- When Region object is part of the transaction and is being updated and RegionDescription value has been changed, only then the Distributed Transaction comes into picture. When the update results in change of attributes other than RegionDescription, the transaction remains local.

To initiate Distributed Transaction for Java Call methods, use the following logic:

```

@StartDistributedTransaction
public void process()
{
    ..
}

```

When WS-AppServer Service invokes the `process` method, it transforms the local transaction into a distributed transaction. Thus, you can handle Distributed Transactions in AppWorks Platform.

You must configure the WS-AppServer Service Container to handle the Distributed Transactions. Without the required configuration, WS-AppServer cannot handle it. In the Connection Point dialog box of the WS-AppServer Service Container, select Enable Distributed Transaction Support option to enable Distributed Transaction management.

Handling Local and Container managed transactions

In WS-AppServer, Container Managed Transactions is enabled by default. If WS-AppServer Service Container is configured with Container Managed Transactions, every WS-AppServer transaction is joined with a java transaction, and the WS-AppServer transaction is committed when the java transaction is committed. To create local transactions in WS-AppServer, use the `BSF.CallInNewTransaction()` method as follows.

```
public static <T> T callInNewTransaction(TransactionalWork<T> work); // class :  
com.cordys.cpc.bsf.busobject.BSF  
public interface TransactionalWork<T>  
{  
    T call();  
}
```

Example

The following example demonstrates how to create local container managed transactions in Ws-AppServer, using the `BSF.CallInNewTransaction()` method.

```
public void insertCategories()  
{  
    BSF.callInNewTransaction(() ->{  
        BusObjectManager bom = new BusObjectManager(null, new Document());  
        try  
        {  
            bom.startTransaction();  
            Categories category = new Categories(new BusObjectConfig(bom, 0,  
BusObjectConfig.NEW_OBJECT));  
            category.setCategoryID(4);  
            category.setCategoryName("Produce");  
            category.setDescription("Dried fruit and bean curd");  
            category.insert();  
            bom.commitTransaction(true);  
        }  
        catch (Throwable t)  
        {  
            if (bom.getTransactionStarted())  
            {  
                bom.abortTransaction();  
            }  
            throw t;  
        }  
        return null;  
    });  
}
```

The call to `BSF.callInNewTransaction()` results in the following:

- The active Java transaction is suspended.
- A new Java transaction is initiated.
- The code provided in the TransactionalWork (in the previous example, the code provided within two curly braces for callInNewTransaction ()) is executed.
- The Java transaction is committed or aborted as required.
- The original Java transaction is resumed.

Starting a transaction

WS-AppServer provides flexibility to handle transactions implicitly (automatically) or explicitly. In an implicit transaction, WS-AppServer starts the transaction automatically along with the AppWorks Platform service. In an explicit transaction, WS-AppServer starts the transaction based on the custom application logic provided. This scenario is more relevant when WS-AppServer functions in an embedded mode within an application.

WS-AppServer provides APIs that contain methods to start a transaction in either of the modes (implicit/explicit). For details on the APIs and the respective methods, refer WS-AppServer SDK.

When Container Managed Transactions is enabled for WS-AppServer, then WS-AppServer suspends the existing java transaction before executing the custom java method, and resumes the java transaction after executing the custom java method.

To start a transaction:

1. Compile the Java code for the method in LDAP.

```
public static ITEM Test(String code) throws Exception
{
    try
    {
        BSF.beginTransaction(String transactionID);
        //Provide logic here;
        BSF.commitTransaction(String transactionID)
    }
    catch (Exception e)
    {
        //Provide abortTransaction logic, when commitTransaction fails.
        Throw e;
    }
    return itemObj ;
}
```

2. In the implementation of the method, set the start parameter to true as shown in the following sample.

```
<implementation type="BsfJavaCall">
```

```

<transaction>
  <start>false</start>
  <!-- If you want to start the transaction through the application, set this
parameter to 'false' -->
</transaction>
<class>ITEM</class>
<method dt="java:ITEM" scope="out">Test</method>
<parameters>
  <key dt="string" scope="in"/>
</parameters>
</implementation>

```

When you omit the transaction element or do not specify anything, by default, WS-AppServer automatically starts the transaction.

3. Run the program to start the transaction.

Retrying a transaction

WS-AppServer supports handling multiple transactions at the same time. In this context, it is possible that two or more processes will try to update the same object simultaneously. To address this, WS-AppServer adopts the concept of optimistic transactions, in which it checks the objects in the database for any interim change that may have happened, and subsequently commits the transaction.

Always, the changes from the first transaction are committed to the database without any difficulty. When the following transaction tries to commit the updates, it receives a `BSFObjectChangedException`, indicating the changed state of the object. Normally, whenever such an exception occurs, the transaction aborts. This condition is more evident in case of update requests, where the request contains both the old and new tuples. When a `BSFObjectChangedException` is thrown, the original data of the object (`<tuple><old>`) is retrieved (using the `object.getOriginalObject()` method) and is updated with the actual data from the database. As a result, the old object in the request attains the same status as that of the object in database. Again, an attempt to commit the transaction is made. This time, the commit would be successful.

Based on the content of the original object, you can take necessary steps. You have the following options:

- In case of an update that is not conflicting, commit the transaction again by using the `com.cordys.cpc.bsf.busobject.BSF class` `commitTransaction()` method. The commit will be successful because the original object data (`<tuple><old>`) will now be according to the content of the database.
- Modify the object based on the content of the original object, commit the transaction by using the `com.cordys.cpc.bsf.busobject.BSF class` `commitTransaction()` method. The commit will be successful because the original object data (`<tuple><old>`) is now according to the content of the database.

- If it is a conflict does not solve on its own, abort the transaction using `com.cordys.cpc.bsf.busobject.BSF` class `abortTransaction()` method , and display an error to the user.

Using WS-AppServer, a transaction can be programmed to make repetitive attempts to commit in spite of such failures and exceptions. The following code is a sample of how you can achieve this.

```

String transactionID = Native.createGuid();
BSF.getObjectManager().setMaxRetry(5);
Region region = null;
do
{
    try
    {
        BSF.beginTransaction(transactionID);
        region = Region.getRegionObject(1);
        region.setRegionDescription(name);
        region.update();
        BSF.commitTransaction();
    }
    catch (Exception e)
    {
        if(e instanceof BsfRetriableException)
        {
            BSF.setRetry(transactionID, (BsfObjectChangedException)e);
        }
        e.printStackTrace();
    }
}while(BSF.retryTransaction(transactionID));

```

The methods `setRetry(String transactionID)` and `retryTransaction(String transactionID)` are instance methods on `BusObjectManager`. The static WS-AppServer methods map to the `BusObjectManager` bound to the current that is, thread bound, `BsfContext`. For details on the methods used in this process, refer WS-AppServer SDK.

Retrieving objects participating in a WS-AppServer transaction

A typical transaction cycle in WS-AppServer encompasses several objects. If the transaction cycle is too long or if it contains a nested transaction, several objects may participate at different stages in the entire transaction cycle.

For various reasons, it may be necessary to identify the objects participating in a transaction. WS-AppServer's transaction management framework helps you to check the objects participating in a transaction at two levels - at the start of a transaction and before a transaction is committed.

Two event listeners in WS-AppServer - `onReceive` and `onTransactionConstraint` help you gain control over the objects in a transaction.

onReceive

This event listener is called at the beginning of a transaction for Insert, Update, and Delete actions. This event will be fired only When request is processed by WS-AppServer. It will not be fired in BPM Transaction context. The event listener contains the following method signature.

```
public void onReceive(UpdateRequestEvent event) throws WSAppsSOAPFault
```

This event listener helps you to get all the objects participating in the transaction. Once you know the objects in a transaction, you can implement any logic that determines the behavior of the objects and controls the transaction. For example, you can set a filter on the objects to get objects of a definite criteria.

To use this event listener, you need to register it explicitly at the start of the relevant service container using the following code.

```
package com.northwind;

import com.cordys.cpc.bsf.connector.BsfConnector;
import com.cordys.cpc.bsf.connector.IOnBsfConnector;
import com.cordys.cpc.bsf.event.Listeners;

public class Initializer implements IOnBsfConnector {
    public void onOpenConnector(BsfConnector connector) {
        System.out.println("Entered onOpenConnector");
        Listeners
            .registerUpdateBodyBlockListener(new MyUpdateBodyBlockListener());
    }

    public void onCloseConnector(BsfConnector connector) {
        System.out.println("Entered onCloseConnector");
    }
}
```

Registration is required just once. Thereafter, the event will be executed for all the transactions. If you do not want the event to be automatically executed, you have to cancel the registration explicitly using the unregister command.

```
public static void
unregisterUpdateBodyBlockListener(IUpdateBodyBlockListener
listener)
```

The unregistration is effective only for the next transaction.

onTransactionConstraint

This event listener is used for checking constraints on multiple objects that participate in a single transaction. It is usually called before the transaction is committed. This event listener helps you to validate if all the set constraints are adhered to. If any constraint violation is reported, the listener aborts the transaction. Thus assured, a transaction can be committed only if the objects in it fulfill the requirements, and are fit to be persisted into the database.

The event listener contains the following method signature:

```
public void onTransactionConstraint(TransactionConstraintEvent
event)
```

To use this event listener:

To use this event listener, you need to explicitly register it. It can be registered in two ways:

- Register on the object manager using the following code.

```
ITransactionConstraintListener tcl = new MyTransactionConstraintListener()
Listeners.registerConstraintListener(BSF.getObjectManager(), tcl)
```

If you are registering the listener on object manager, you need to register it in the Extension class of each class. At any point, if you want to know the event listeners registered with the object manager, use the following code.

```
public static java.util.Enumeration getConstraintListeners(BusObjectManager bom)
```

- Register on the config object using the following code.

```
package com.northwind;

import com.cordys.cpc.bsf.busobject.Config;
import com.cordys.cpc.bsf.connector.BsfConnector;
import com.cordys.cpc.bsf.connector.IOnBsfConnector;
import com.cordys.cpc.bsf.event.ITransactionConstraintListener;
import com.cordys.cpc.bsf.event.Listeners;

public class Initializer implements IOnBsfConnector {

    public void onOpenConnector(BsfConnector connector) {
        System.out.println("Entered onOpenConnector");
        Config config = connector.getConfigObject();
        ITransactionConstraintListener tcl = new TransactionConstaraintListener();
        Listeners.registerConstraintListener(config, tcl);
    }
}
```

```

public void onCloseConnector(BsfConnector connector) {
    System.out.println("Entered onCloseConnector");
}
}

```

If you are registering the listener on config object, you need to register it only once at the start of the relevant WS-APPServer service container, that is, by implementing `IOnBsfConnector` interface. The same event will be effective on all transactions. The following code displays the one-time registration of the listener on config object:

```
public static java.util.Enumeration getConstraintListeners(Config config)
```

Once the event listener is registered using this method, it will be called for every transaction. To prevent it from being automatically called, you have to cancel its registration, using the following command:

```
public static void unregisterConstraintListener(Config config,
ITransactionConstraintListener listener)
```

The unregistration will be effective only for the next transaction. The current transaction will still execute the `onTransactionConstraint()` event.

Example

In the application logic, you have specified a constraint that the transaction should contain at least one object of Jumbo class. Before the transaction is committed, you would want to check if this criterion is met. If not, you would like to abort the transaction.

Using the `onTransactionConstraint` event listener will get all the objects participating in the transaction. Then, you can execute the constraint of finding a Jumbo object. The following code sample describes the constraint.

```

public void onTransactionConstraint(TransactionConstraintEvent event)
{
// there must be at least 1 Jumbo Object in the transaction
BusObjectIterator jumbos = event.getUpdatedObjects().getObjectsByType(Jumbo.class);
int nrOfJumbos = 0;
while ( jumbos.hasMoreElements() )
{
// Add required logic
nrOfJumbos++;
}
if ( nrOfJumbos == 0 )
{
event.addError(null, null, "Number of Jumbos being inserted in the request is 0");
}
}

```

As evident from the code, if a Jumbo object is not found, it will result in an error.

Adding a new property

Before you start

- Access the Management Console and connect to LDAP as an authenticated user.

To add a new AppWorks Platform property to wcp.properties:

1. Click **Platform Properties** in the Management Console window.
The Platform Properties window opens.
2. Click .
The Add Property window opens.
3. Enter the **Name** and **Value** for the property.
4. Click **OK**.
The new property appears in the list of properties in Platform Properties window.
5. Click .

A new property is added to the `wcp.properties` file.

Adding dynamic filters

This topic describes the procedure to add dynamic filters to the implementation and interface of the Web service operation (method).

To add dynamic filters:

1. In the implementation of the GetEmployeeDF method, change the WHERE field and add the following dynamic filters:
`WHERE :DF1 and :DF2 and :DF3`
In this case you must provide 3 conditions at runtime. It is also possible to add one dynamic filter in the implementation and at runtime.
2. Change the parameter value by replacing the EmployeeID parameter with the following:
 - `<DF1 type='dfilter' />`
 - `<DF2 type='dfilter' />`
 - `<DF3 type='dfilter' />`

Implementation is as follows:

```
<implementation type="DBSQL">
<constructor language="DBSQL">
<query>SELECT "EmployeeID", "LastName", "FirstName", "City" FROM "Employees"
```

```

WHERE :DF1 and :DF2 and :DF3</query>
<parameters>
<DF1 type="dfilter"/>
<DF2 type="dfilter"/>
<DF3 type="dfilter"/>
</parameters>
</constructor>
</implementation>

```

3. In the interface of the method, change the GetEmployee DF element and the interface is as follows:

```

<xsd:element name="GetEmployeeDF">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="DF1">
<xsd:complexType mixed="true">
<xsd:attribute default="dfilter" name="type"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="DF2">
<xsd:complexType mixed="true">
<xsd:attribute default="dfilter" name="type"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="DF3">
<xsd:complexType mixed="true">
<xsd:attribute default="dfilter" name="type"/>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

```

Transient attributes

Generally, all attributes from a standard class map to a column in the database table. When the object of such a class is saved to the database, each attribute is mapped to a column (1-1 mapping). This is a persistent attribute.

Another type of attribute known as a transient attribute does not map to the column in a database table, and is therefore excluded from persisting to the database. Typical examples of transient attributes are calculated and fields in related tables. WS-AppServer facilitates setting an attribute's persistent mode through a property called "persistence". If you leave the property value blank (which is also the default value), the attribute is set to "Persistent" mode, in which the attribute is mapped to a column and the data is persisted. If you select transient, the attribute is set to Transient mode, in which the attribute is not persisted. The persistence mode of an attribute is stored in the Class Information (ClassInfo) and is available during runtime.

When an object with transient attributes is saved in the database (through insert, update, or delete), the transient attributes are excluded from mapping to the database. For an object that has both persistent and transient attributes, the persistent attributes are updated, while the transient attributes are ignored.

For persistent attributes, the new state in the database is stored in the object (for example, database-generated number, data/time formatting, and so on). However, when a transient attribute is updated inside a transaction, it does not change the state of the object. If transient attributes need to be updated because of the updates in a transaction, use `onAfterCommit` event listener.

Example

Generate the WSAppSOrders class on WSAppSOrders table, where one of the attributes is EmployeeID, which is a persistent field mapped to the EmployeeID column of the WSAppSOrders table. The transactional requirement is to display the EmployeeName along with EmployeeID. EmployeeName is a persistent field in the WSAppSEmployee table. This means, when a WSAppSOrders object is retrieved using GetWSAppSOrdersObject method, the EmployeeName for the given EmployeeID should be available in the response. The attribute EmployeeName will be used only for display and will not be persisted because it does not belong to WSAppSOrders table.

To achieve this:

1. Add a transient attribute EmployeeName to the WSAppSOrders class.
You can add this by following the procedure in topic [Adding an Attribute to a WS-AppServer Model](#).
2. While filling the Persistence properties, select **Transient**.

You have created a Transient Attribute.

To implement the logic:

- In the extension class of WSAppSOrders, implement the `onAfterLoad()` method as shown.

```
public void onAfterLoad() {

    int empid = this.getEmployeeID();

    String queryText = "select * from \"WSAppSEmployees\" where \"EmployeeID\" = :employeeid";
    QueryObject query = new QueryObject(queryText);
    query.addParameter("employeeid", "WSAppSEmployees.EmployeeID",
    QueryObject.PARAM_INT, new Integer(empid));
    query.setResultClass(WSAppSEmployees.class);

    WSAppSEmployees emp = (WSAppSEmployees)query.getObject();
    System.out.println(emp.getLastName());
    this.setEname(emp.getLastName());
}
```

So, when the WSAppSOOrders object is persisted, only the EmployeeID is persisted and not the EmployeeName.

When objects are loaded, by default, only the persistent objects are loaded. This is done through a query. However, there are other ways to load the transient attributes:

- Using the onAfterLoad listener - After an object is read through a query, the onAfterLoad listener is called. You can fill the transient attributes (based on calculations, reading other data, etc.) in this listener.
- Using Query - In certain situations, especially with related fields (such as the EmployeeID and EmployeeName), the transient attributes can be loaded along with the persistent attributes in the same query (for example, using join). However, such a workaround requires exclusive coding by the application developer.

Attribute properties interface

These properties determine the behavior of the attribute in the model to which it is being added.

General

Name	The name of the attribute
Mapped Name	<p>The name that corresponds to a column of the database table. This is also the name that is used to generate the accessor and mutator methods of an attribute. It is the name by which the attribute is implemented in the Java class. If you do not want to retain the default names, you may replace these with names of your preference. For example, in an attribute which has CustomerID as the Name and CustID as the Mapped Name, the following accessor and mutator methods are generated:</p> <pre>public void setCustID(String value) public String getCustID()</pre> <p>The Mapped Name values take precedence over Name values, while generating these methods.</p>
Type	<p>The data type of the attribute. The following data types are supported:</p> <ul style="list-style-type: none"> ■ boolean - A value of either 0 to represent 'false' or 1 to represent 'true' ■ Unsigned Byte (ui1) - A one-byte unsigned integer ■ Unsigned Short (ui2) - A two-byte unsigned integer ■ Unsigned Integer (ui4) - A four-byte unsigned integer ■ Unsigned Long (ui8) - An eight-byte unsigned integer ■ Short Integer (i2) - A two-byte short integer ■ Integer (i4) - A four-byte integer

	<ul style="list-style-type: none"> ■ Long Integer (i8) - An eight-byte long integer ■ Float (r4) - A floating point number with four-byte encoding ■ Double (r8) - A floating point number with double precision ■ Currency (cy) - A floating point number that represents 'currency' data types ■ string - String ■ guid - A string that represents a globally unique identifier ■ dateTime - A date with an optional time data ■ WS-AppServer always works with GMT/UTC date and time format. It is therefore recommended to follow the same format during application development. For example, in the soap method of implementation type "BsfJavaCall", the arguments of type "Date" must be handled appropriately. ■ Base64 encoded BLOB (bin.base64) - MIME-style Base64 encoded binary large object (BLOB)
Persistence	Determines the mode of persistence. The acceptable values are: <ul style="list-style-type: none"> ■ Blank (Persistent) - Default mode of mapping an attribute to a database column ■ Transient - Attribute does not map to a database column and is not persisted. The value will remain in memory as long as the transaction requires it and will be removed from the memory the moment transaction is over.
Changeability	Indicates the changeability property of the attribute. The permissible values are: <ul style="list-style-type: none"> ■ Changeable - Value can be modified at any point of time ■ addOnly - Value can be assigned anytime (during insert or update operation) but only once, and cannot be modified subsequently ■ Frozen - Value assigned during creation of the object but cannot be modified subsequently
Unique	Determines whether the attribute is part of the unique identifier for the class
Required	Indicates the necessity of providing a value for the attribute

Other

Derived	Displays the class from which this attribute is derived
Scale	Indicates the number of digits to the right of the decimal point. This property is not applicable if you select String as the data type.
Precision	Indicates the total number of digits used

Initial	Indicates the initial value of the attribute, applicable only for validate requests
---------	---

Lower/Upper bounds

Max Length	Indicates the maximum length of the attribute. This is applicable for string data type only. For standard classes, the max length is determined by the actual length of the data.
Min Length	Indicates the minimum length of the attribute. This is applicable for string data type only.
Min Inclusive	Indicates the lower limit of the value, including the value. This is applicable for all data types excluding string.
Min Exclusive	Indicates the lower limit of the value, excluding the value. This is applicable for all data types excluding string.
Max Inclusive	Indicates the upper limit of the value, including the value. This is applicable for all data types excluding string.
Max Exclusive	Indicates the upper limit of the value, excluding the value. This is applicable for all data types excluding string.

Using cursors in WS-AppServer

WS-AppServer supports the functionality of the standard cursor. The cursor must be mentioned explicitly as a parameter (with scope "inout") in the implementation. The following are the sample implementation, the Java code, and the SOAP request formats, displaying the usage of the cursor attribute and its properties.

Implementation

```
<implementation type="BsfJavaCall">
  <class>application.Orders</class>
  <method ct="tuples" dt="java:application.Orders[]"
scope="out">getOrdersObjects</method>
  <parameters>
    <fromCustomerID dt="String"/>
    <toCustomerID dt="String"/>
    <cursor ct="cursor" dt="String" scope="inout"/>
  </parameters>
</implementation>
```

Java code

```
public static BusObjectIterator getITEMObjects(int fromCODE, int toCODE,
com.cordys.cpc.bsf.query.Cursor cursor)
```

```

{
    String queryText = "select * from ITEM where CODE between :fromCODE and
:toCODE";
    QueryObject query = new QueryObject(queryText);
    query.addParameter("fromCODE", "ITEM.CODE", QueryObject.PARAM_INT, new Integer
(fromCODE));
    query.addParameter("toCODE", "ITEM.CODE", QueryObject.PARAM_INT, new Integer
(toCODE));
    query.setResultClass(ITEM.class);
cursor.setSameconnection(true);
query.setCursor(cursor);
return query.getObjects();
}

```

SOAP request

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <getOrdersObjects xmlns="http://schemas.cordys.com/appserver">
      <cursor SameConnection="true" numRows="10"/>
      <fromCustomerID>10000</fromCustomerID>
      <toCustomerID>20000</toCustomerID>
    </getOrdersObjects>
  </SOAP:Body>
</SOAP:Envelope>

```

Setting `SameConnection = "true"` preserves the integrity of the cursor information under all conditions. However, it may result in a slight performance overload because of indefinite waiting time for the connection to be freed up.

Using Eclipse to implement event listeners

In WS-AppServer, both explicit and implicit event listeners are used to perform several functions. While implicit event listeners are implemented as methods in the extension class, explicit event listeners are available in a separate class and are accessible through the Java interface of that listener.

As a developer, you will often be working with these event listeners. A convenient way to work with them is to use the Eclipse tool.

Importing templates into Eclipse

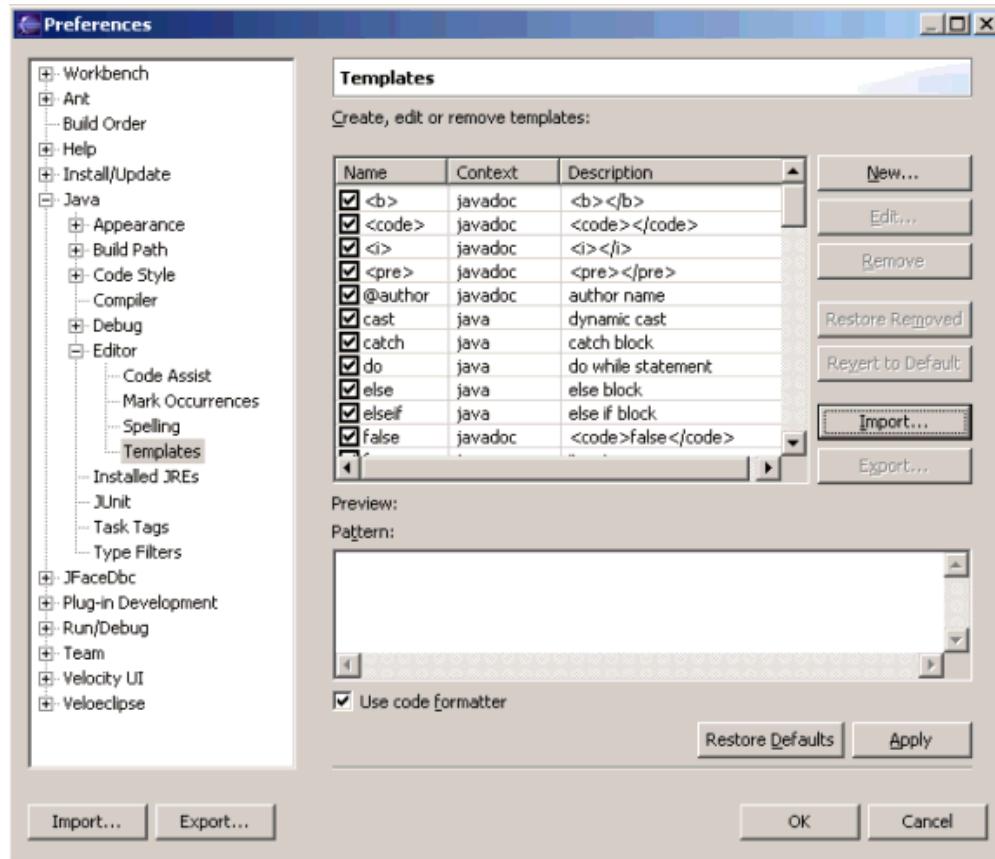
WS-AppServer packages all the event listener templates in a file that can be downloaded and used in Eclipse.

To implement the WS-AppServer event listeners using Eclipse:

1. Create an XML file using the code in WS-AppServer Event Listener Templates and save it on your local system.

2. In **Eclipse**, open **Window > Preferences**.

The Preferences window appears.



3. In the left pane, click **Java > Editor** and select **Templates**.

The Templates work area appears in the right pane.

4. In the Templates work area, click **Import**.

The Importing Templates dialog box appears.

5. Browse the folders to locate the XML file containing the event listener templates, and click **Open**.

The events from the list of available event listener templates are imported and displayed in the Templates work area (right pane).

6. Click **OK**.

The event listener templates are ready to be used.

The procedure is described according to Eclipse 3.0. In later Eclipse versions, this procedure may either remain the same or slightly vary.

Implementing event listeners

Event listeners follow different naming convention at object and attribute levels:

- onAccessEvent at object level -public void onAccess(ObjectAccessEvent)
- onAccessEvent at attribute level, 'abc' being the attribute -public void onAccess_abc(AttributeAccessEvent)

While using Eclipse, you need to remember these conventions to identify the correct event listener. When you want to insert a specific event listener into the code that you develop, just type the initial letters of the event listener. Eclipse will display all the event listeners starting with those initial letters, along with the code. You can then select the relevant event listener. For example, you want to use the onAccess event listener. In Eclipse, type 'onAcc' and press CTRL+Space to get the event listeners starting with 'onAcc' along with the respective code. Upon selection, the code is inserted into the workspace, as shown in the following samples:

- At the object level, the code is

```
public void onAccess(com.cordys.cpc.bsf.event.ObjectAccessEvent event) {  
    //add the logic here  
}
```

- At the attribute level, the code is

```
public void onAccess_attrName(  
    com.cordys.cpc.bsf.event.AttributeAccessEvent event) {  
    //add the logic here  
}
```

As you can see, the code is developed with minimal effort.

In the attribute level code, replace attrName with the actual name of the attribute.

Embedding WS-AppServer functionality in applications

WS-AppServer's functionality in applications is primarily made available in the form of a Service. However, as an alternative, it is possible to leverage its functionality as an embedded server within an application's processing layer, without creating a WS-AppServer Service. In the absence of an external processor, these applications use WS-AppServer's functionality through Java APIs that are available in the SDK. These APIs help establish communication between the application and the configuration file containing WS-AppServer functionality.

WS-AppServer uses the XML Query (XQY) functionality provided by AppWorks Platform to establish database connectivity. Therefore, to use WS-AppServer in embedded mode, you must install AppWorks Platform.

Before you begin:

- Ensure that AppWorks Platform is installed on your computer and all the required classes, jars, and external jars required for your application to work are available at the

appropriate location on your computer.

- Ensure that `XmlForJava.dll`, which is available as part of the AppWorks Platform installation, is set in the path of system environment variable.

To embed WS-AppServer functionality:

1. Create an XML file using the structure of the `Bsfconfig` file that is provided in `bsfconfig.xml` with JDBC attributes.
In the config file, you need to provide a Base-64 encoded value for the `Password` property.
2. Place this file at a location of your choice.
3. From `<AppWorks Platform_installdir>\<instance name>\components\wsappserver\config`, open the `wsapps.properties` file using a suitable text editor.
The `wsapps.properties` file opens, displaying two attributes.
4. Provide the path of the file containing WS-AppServer processor configuration details (that you created in Step 1) against the attribute `wsappserver.default.config.file=`, in the format as shown.
For example, in Windows, `wsappserver.default.config.file= D:\<AppWorks Platform_installdir>\<instance name>\bsfbsfconfig.xml` and in Linux `wsappserver.default.config.file= /opt/Cordys/ESB/bsf/bsfconfig.xml`.
In case of Windows, remember to place the "`\`" before ":" exactly as shown here for the reference to work.
WS-AppServer framework uses the value of this attribute to locate and read the WS-AppServer processor configuration details from the configuration file.
5. Save the file and close it.
6. Invoke the WS-AppServer client using the Java code.

WS-AppServer is configured to be used in embedded mode.

Alternative approach

As an alternative, you can avoid creating the XML file and still configure WS-AppServer to function in an embedded mode.

Insert the following sample code (containing configuration details) into the Java code that is used to invoke WS-AppServer:

```
package com.mycomp.myapp; import java.io.UnsupportedEncodingException; import
com.cordys.cpc.bsf.busobject.BSF; import com.cordys.cpc.bsf.busobject.BsfContext;
import com.cordys.cpc.bsf.busobject.Config; import com.eibus.xml.nom.Document;
import com.eibus.xml.nom.XMLException; import
com.cordys.cpc.bsf.busobject.exception.BsfRuntimeException; public class MyApp {
private static Document m_oDocument = new Document(); private static BsfContext
context; public static void initialize() { int iConfig = 0; try { iConfig = m_
oDocument.parseString(" <configurations/>"); } catch (UnsupportedEncodingException
e) { e.printStackTrace(); } catch (XMLException e) { e.printStackTrace(); } //}
```

```

configurations refers to the WS-AppServer configuration. Config wsAppConfig = new
Config(); wsAppConfig.setConfig(iConfig); context = BSF.initBsfContext(); if
(context == null) { throw new BsfRuntimeException("Context not initialized
properly."); } } public static void main(String args[]) { // initialize context
initialize(); // Write your application specific code here. BSF.unregisterContext
(context); //To unregister the context that was initialized. }

```

Subsequently, running the Java code would configure WS-AppServer to function in embedded mode.

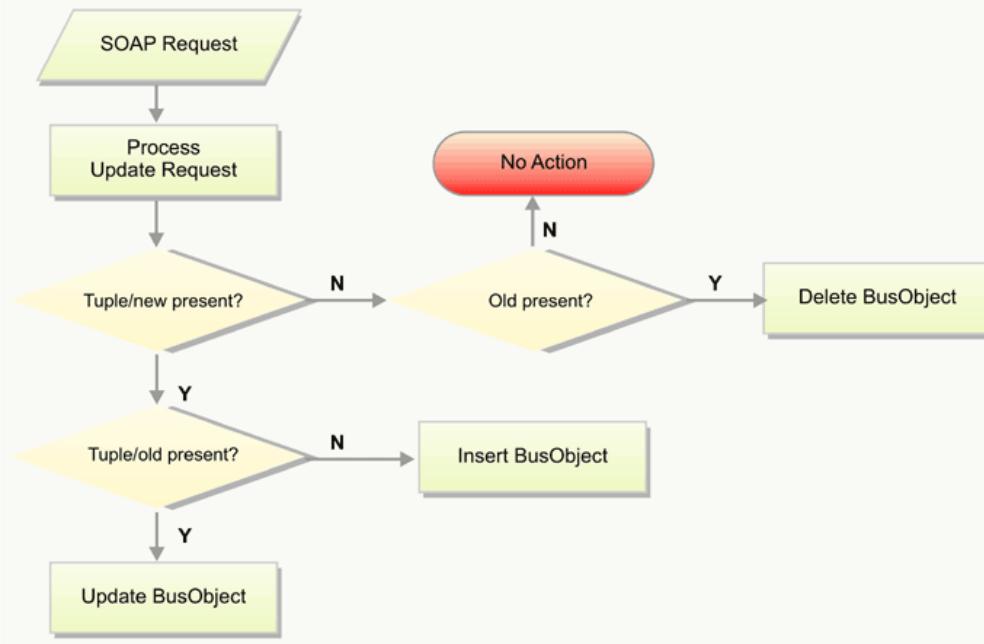
Working with SOAP requests and responses

WS-AppServer is the business logic layer in AppWorks Platform. It supports a standard set of SOAP requests that a SOAP client uses for XML messaging. It also defines a standard set of response formats that will return SOAP results as desired.

The following table lists the SOAP requests used in WS-AppServer.

Request	Purpose
Update	Invoke the application logic to insert , update , and delete objects in the database. The execution of these actions depends upon the tuple status. This is illustrated in the figure following this table. In any update SOAP request, the XML attribute reply= 'yes no' determines whether the response will contain the tuple or not. Specifying 'yes' will return the tuple in the SOAP response, whereas specifying 'no' will not return the tuple. However, if nothing is specified, the response will contain the tuple (as a result of default behavior being reply= 'yes').
GetObject	Invoke the application logic to retrieve objects from the database and display it on the user interface. For example, GetEmployeesObject , GetEmployeesObjects.
Validate	Invoke the application logic to validate objects without saving them in the database.
Query	Retrieve data from the database using a query.

Workflow of a Generic Update SOAP Request



The following table lists the different response types that are returned.

Response Type	Description	Method Used
XML	Response is in the form of an XML document.	execute
Single java object	Response is in the form of a single object. The class of the object that is returned can be specified using the setResultClass method. If you do not set the result class, an object of class AnonymousBusObject is returned.	getObject
Collection of java objects	Response is in the form of a collection of objects. The class of the objects that are returned can be specified using the setResultClass method. If you don't set the result class, objects of class AnonymousBusObject are returned.	getObjects

Creating and running a SOAP request

In WS-AppServer, you can write logic to create a SOAP request and run it. You can set the response type to any of those described in Table 2.

The following sample code describes the creation of a SOAP request.

```
// make SOAP request String namespace =
```

```

"http://schemas.cordys.com/NewAppServererver"; String methodName =
"GetCustomersObject"; String customerID = "GREAL"; String[] paramNames = new
String[]{CustomerID}; Object[] paramValues = new Object[]{customerID}; // execute
SOAPRequestObject sro = new SOAPRequestObject(namespace, methodName,
paramNames, paramValues); sro.setResultClass(Customers.class); Customers cust =
(Customers)sro.getObject();

```

Parameters of the SOAP request

The following table describes the two parameters used in the example.

Parameter	Description	
paramNames	Contains the name of the parameter.	
paramValues	Contains the values for the specified parameter names. The parameter values are defined based on the following specification:	
	Parameter Type	Placement in the SOAP Request
	BusObject	As a tuple
	BusObjectIterator	As a collection of tuples
	Any other type	As a string representation (using <code>toString()</code>)

Parameters for the SOAP request are passed as arguments to the constructor. The following code sample illustrates this statement:

```

SOAPRequestObject oReq = new SOAPRequestObject(namespace, methodName, paramNames,
paramValues);

```

The `SOAPRequestObject` class has a method called `addParameterAsXML` that allows you to pass complex XML documents as parameters.

Controlling the XML output of the SOAP response

By default, if you use the `execute` method to run the SOAP request, the SOAP response will be embedded in the `tuple/old` and the `methodname` tags. If you do not want the response to be embedded in these tags, ensure that your Java class implements the `com.eibus.applicationconnector.java.Tupable` interface, or there should be an attribute called `wt` with value `false` in the method implementation.

```

<implementation type="BsfJavaCall">
  <class>bsf.ub.gen.UIOrder</class>
  <method ct="elements" dt="int" scope="out" wt="false">getOrderProduct</method>
  <parameters/>
</implementation>

```

Ensure that the content type attribute (ct) in the implementation has been set to elements.

Tracking the requests and responses

You can build a code in your program to trace the XML Query (XQY) requests and responses and view them on the Admin console.

- To trace the request, set

```
com.cordys.cpc.bsf.query.xqy.XqyUpdateHandler.traceRequest = true;
```

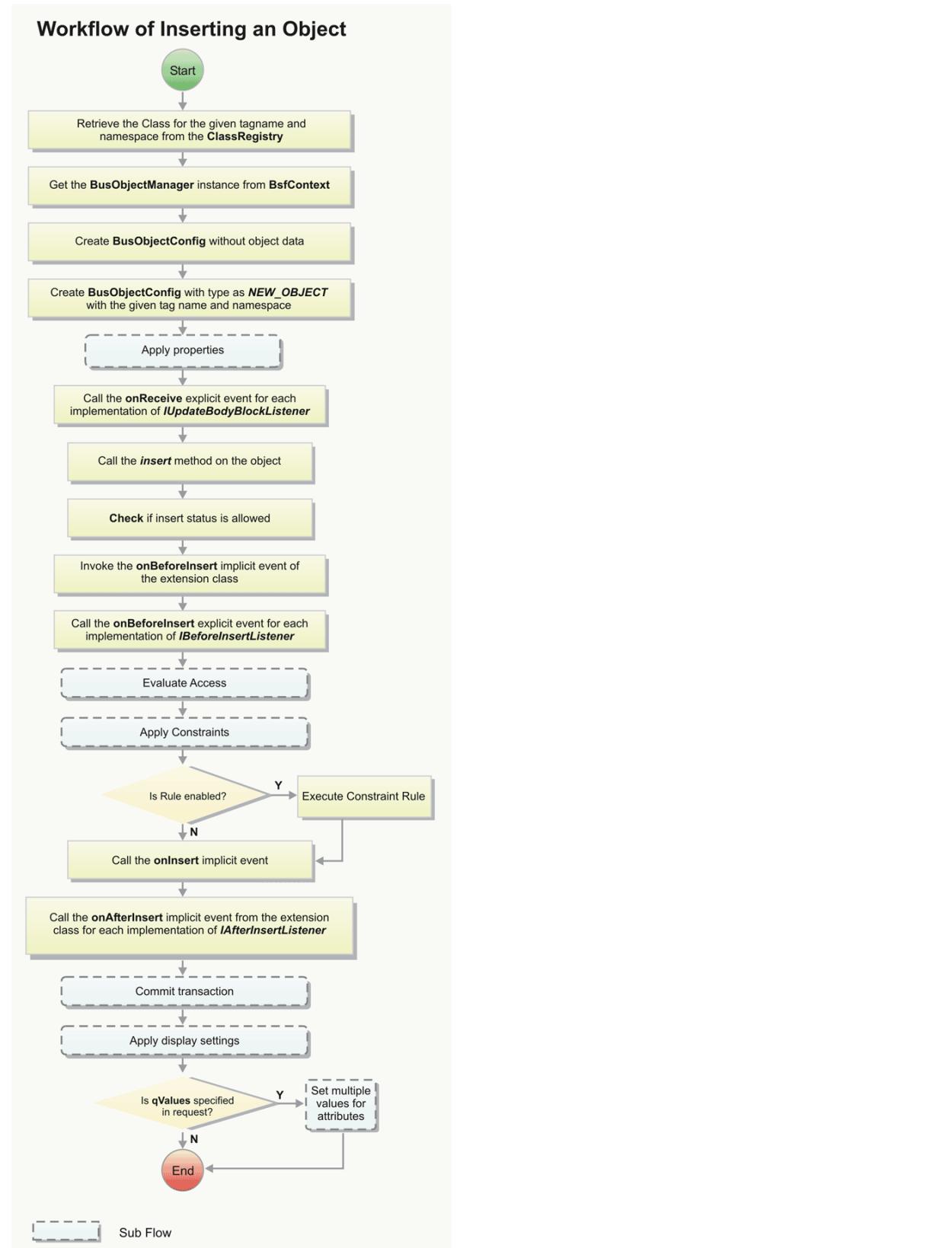
- To trace the response, set

```
com.cordys.cpc.bsf.query.xqy.XqyUpdateHandler.traceResponse = true;
```

Insert BusObject SOAP request

The Insert BusObject SOAP request is used to insert new data into the database. The SOAP request is an XML structure, whereas the actual implementation of the request is through a Java method.

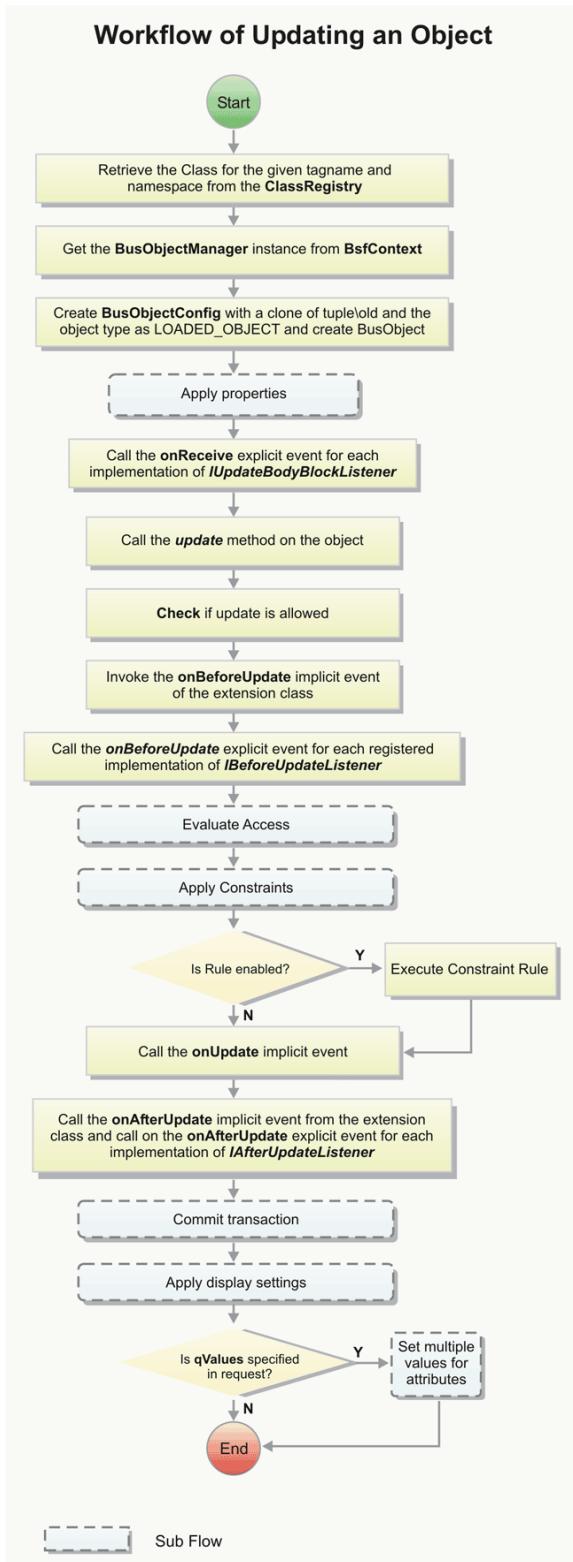
When the request is sent to WS-AppServer to insert a BusObject, it triggers a series of events. This is illustrated in the following figure.



WS-AppServer facilitates using some of the XQY extensions while executing these requests.

Update BusObject SOAP request

The Update BusObject SOAP request is used to update existing data in the database. The SOAP request is an XML structure, whereas the actual implementation of the request is through a Java method. When the request is sent to WS-AppServer, it triggers a series of events. This is illustrated in the following figure.



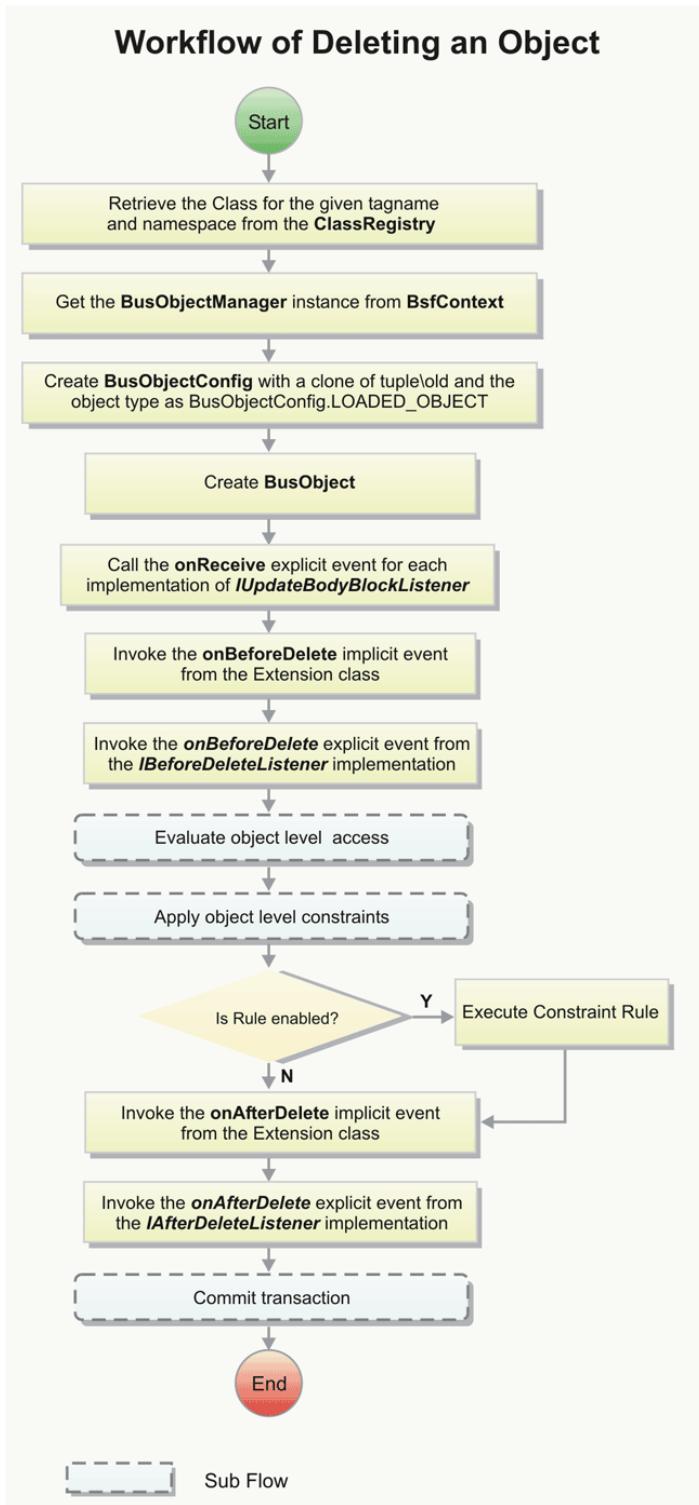
Updating BusObjects without loading them from database

As an alternative approach, you can update BusObjects without sending a request to the database. You can retrieve a BusObject using its primary key, and use the SetAsPersistent method to change the state of the object to LC (Loaded + Changed). To update the BusObject in the database, you can change the values of the required attributes, and update the object in the database by using the update method.

WS-AppServer facilitates using some of the XQY extensions while executing these requests.

Delete BusObject SOAP request

The Delete BusObject SOAP request is used to delete data from the database. The SOAP request is an XML structure, whereas the actual implementation of the request is through a Java method. When the request is sent to WS-AppServer, it triggers a series of events. This is illustrated in the following figure.



Deleting BusObjects without loading them from database

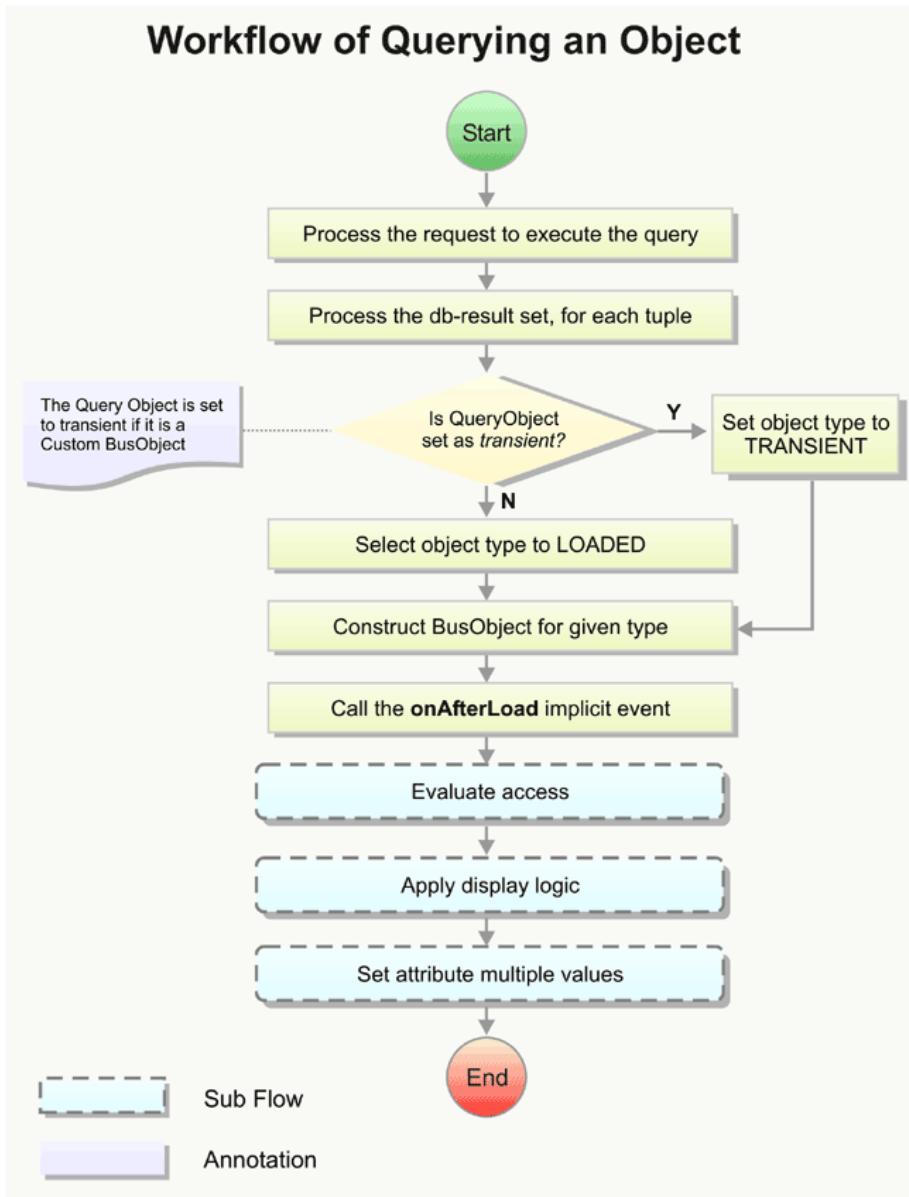
As an alternative approach, you can delete BusObjects without sending a request to the database. The procedure to delete a BusObject is similar to the update procedure. After you change the state of the BusObject to LC (Loaded + Changed), you can delete the object from the database by using the delete method.

Note: WS-AppServer facilitates using some of the XQY extensions while executing these requests. For more information, see Leveraging XQY Extensions in WS-AppServer in the *AppWorks Platform Administrator's Guide*.

Query BusObject SOAP request

The Query BusObject SOAP request is used to retrieve data from the database. The SOAP request is an XML structure, whereas the actual implementation of the request is through a Java method. When the request is sent to WS-AppServer, it triggers a series of events. This is illustrated in the following figure.

When the request is sent to WS-AppServer, it triggers a series of events. This is illustrated in the following figure.



A custom busobject is made transient upon querying.

The **GetObject** request and response

In a client-server interaction through SOAP request and response, **GetObject** is used to query an underlying database through the XQY database access layer. The **GetObject** request is enveloped in XML structure and contains attributes that help in better client-server communication. The response is also framed in an XML structure.

The following section provides information on the different SOAP request and response formats based on the soapresult types, in two scenarios - when there is a single BusObject and when there are multiple BusObjects.

SOAP result: tupleset

Single BusObject

```
<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
  <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
      <id>PARAMETER</id>
      <name>PARAMETER</name>
    </Test>
  </param1>
</InsertObject>
```

Multiple BusObjects

```
<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
  <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
      <id>PARAMETER</id>
      <name>PARAMETER</name>
    </Test>
  </param1>
</InsertObject>
```

Request

```
<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
  <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
      <id>PARAMETER</id>
      <name>PARAMETER</name>
    </Test>
  </param1>
</InsertObject>
```

Response

```
<InsertObjectResponse xmlns="http://schemas.cordys.com/bsf/Testing"> <tuple>
<old> <Test xmlns="http://schemas.cordys.com/bsf/Testing"> <id>PARAMETER</id>
<name>PARAMETER</name> </Test> </old> </tuple> <tuple> <old> <Test
xmlns="http://schemas.cordys.com/bsf/Testing"> <id>PARAMETER</id>
<name>PARAMETER</name> </Test> </old> </tuple> </InsertObjectResponse>
```

SOAP result: notuple

Single BusObject

```
<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
  <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
      <id>PARAMETER</id>
      <name>PARAMETER</name>
    </Test>
  </param1>
</InsertObject>
```

Mulitple BusObjects

```
<InsertObjectResponse xmlns="http://schemas.cordys.com/bsf/Testing">
  <return>
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
      <id>PARAMETER</id>
      <name>PARAMETER</name>
    </Test>
  </return>
</InsertObjectResponse>
```

Request

```
<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
  <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
    <item xmlns="http://schemas.cordys.com/bsf/Testing">
      <Test xmlns="http://schemas.cordys.com/bsf/Testing">
        <id>PARAMETER</id>
        <name>PARAMETER</name>
      </Test>
    </item>
    <item xmlns="http://schemas.cordys.com/bsf/Testing">
      <Test xmlns="http://schemas.cordys.com/bsf/Testing">
        <id>PARAMETER</id>
        <name>PARAMETER</name>
      </Test>
    </item>
  </param1>
</InsertObject>
```

Response

```
<InsertObjectResponse xmlns="http://schemas.cordys.com/bsf/Testing">
  <return>
    <item>
```

```

<Test xmlns="http://schemas.cordys.com/bsf/Testing">
    <id>PARAMETER</id>
    <name>PARAMETER</name>
</Test>
</item>
<item>
    <Test xmlns="http://schemas.cordys.com/bsf/Testing">
        <id>PARAMETER</id>
        <name>PARAMETER</name>
    </Test>
</item>
</return>
</InsertObjectResponse>

```

SOAP result: default

Single BusObject

```

<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
    <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
        <Test xmlns="http://schemas.cordys.com/bsf/Testing">
            <id>PARAMETER</id>
            <name>PARAMETER</name>
        </Test>
    </param1>
</InsertObject>

```

Multiple BusObjects

```

<InsertObjectResponse xmlns="http://schemas.cordys.com/bsf/Testing">
    <tuple>
        <old>
            <Test xmlns="http://schemas.cordys.com/bsf/Testing">
                <id>PARAMETER</id>
                <name>PARAMETER</name>
            </Test>
        </old>
    </tuple>
</InsertObjectResponse>

```

Request

```

<InsertObject xmlns="http://schemas.cordys.com/bsf/Testing">
    <param1 xmlns="http://schemas.cordys.com/bsf/Testing">
        <item xmlns="http://schemas.cordys.com/bsf/Testing">
            <Test xmlns="http://schemas.cordys.com/bsf/Testing">
                <id>PARAMETER</id>
                <name>PARAMETER</name>
            </Test>
        </item>
    </param1>
</InsertObject>

```

```

        </Test>
    </item>
    <item xmlns="http://schemas.cordys.com/bsf/Testing">
        <Test xmlns="http://schemas.cordys.com/bsf/Testing">
            <id>PARAMETER</id>
            <name>PARAMETER</name>
        </Test>
    </item>
</param1>
</InsertObject>

```

Response

```

<InsertObjectResponse xmlns="http://schemas.cordys.com/bsf/Testing">
    <tuple>
        <old>
            <Test xmlns="http://schemas.cordys.com/bsf/Testing">
                <id>PARAMETER</id>
                <name>PARAMETER</name>
            </Test>
        </old>
    </tuple>
    <tuple>
        <old>
            <Test xmlns="http://schemas.cordys.com/bsf/Testing">
                <id>PARAMETER</id>
                <name>PARAMETER</name>
            </Test>
        </old>
    </tuple>
</InsertObjectResponse>

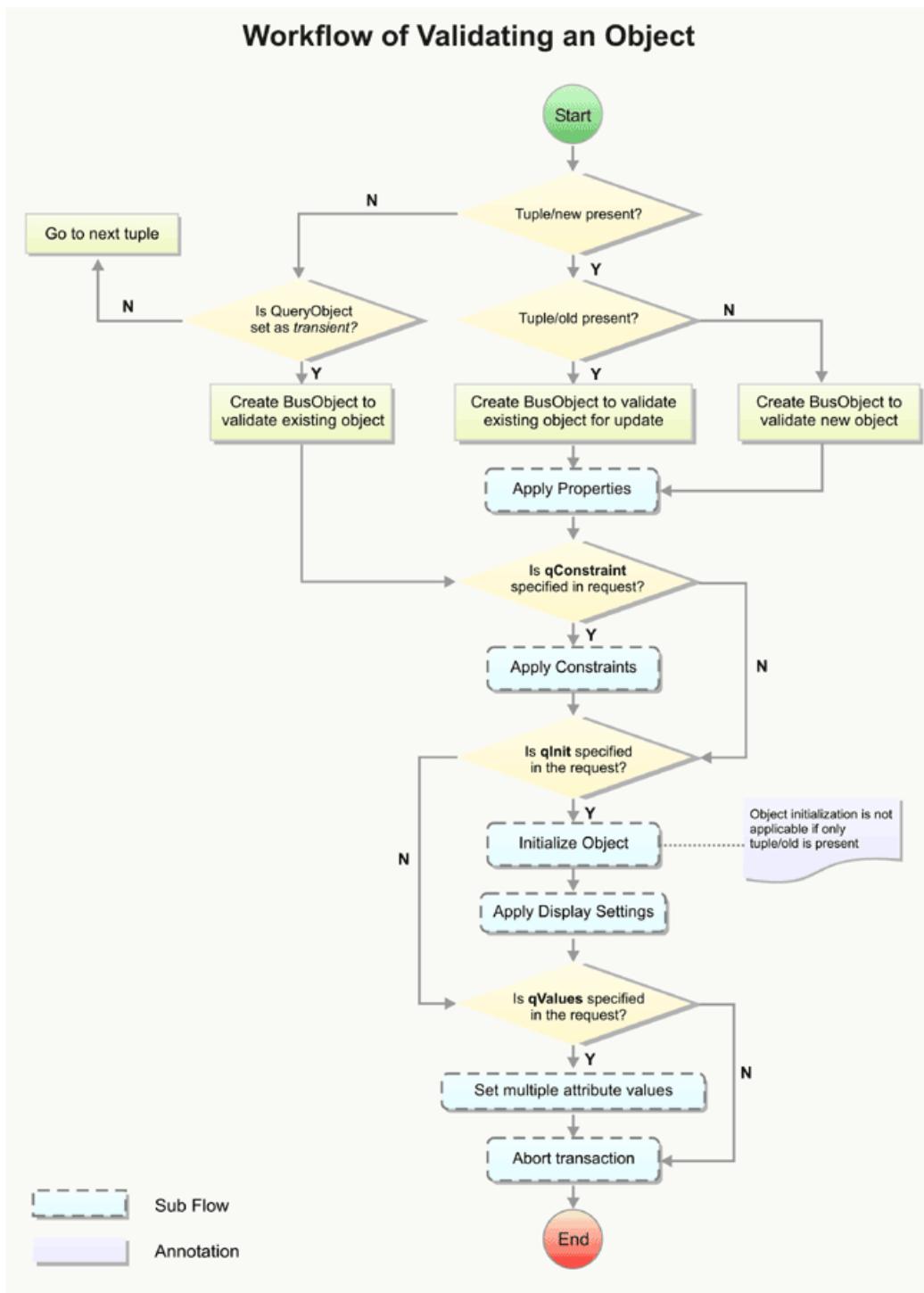
```

WS-AppServer facilitates using some of the XQY extensions while executing these requests. For more information, see Leveraging XQY Extensions in WS-AppServer in the *AppWorks Platform Administrator's Guide*.

Validate BusObject SOAP request

A Validate BusObject SOAP request is meant to execute application logic in the backend, without updating the database immediately. Similar to the Update request, the Validate request is in XML and uses tuple old/new to pass objects. A Validate request requires its objects and attributes to be initialized, and to carry some constraints.

When the request is sent to WS-AppServer, it triggers a series of events. This is illustrated in the following figure.



WS-AppServer validates an object only when the `qConstraint` attribute for that object is set to "1". WS-AppServer analyses the information in the `tuple/old` and `tuple/new` tags for differences. If there are differences, then the constraint handlers for those elements are triggered. The different values for `qConstraint` are listed in the following table.

Value	Description
0	Do not use the object, but check only enabled attributes. Stop if error found.
0*	Do not use the object, but check only enabled attributes. Do not stop if a constraint error is found.
1	Do use the object and stop at the first constraint error.
1*	Use the object and check the constraints for all attributes. Do not stop if a constraint error is found.

The Validate response would include the modifications in the object carried out by the event handlers, which apply to the object. Certain XML attributes are added to the XML node that are used to perform the validation. The following attributes can be returned.

Attribute	Output/Description
access	Access level of the property. The possible values are 'edit' (default), 'read', and 'no'. The UI will adapt the form accordingly.
check-error	Error message
check-info	Information message
check-warning	Warning message
default	Determines whether a certain property is set as a result of applyDefaults. The possible values are "true" and "false" (default). Note: This attribute is important for a next validation that uses applyDefaults. Only fields with default='true' can be updated by applyDefaults.

Initialization of objects and attributes

Initialization is necessary to check the behavior of an object when business logic is applied on it. Objects are initialized when Validate requests are fired.

In a Validate SOAP request, the `qInitattribute` is used (at the object level) to initialize the object. `qInitaccepts` two values:

- 1 - sets `qInit` to true and initializes the object
- 0 - sets `qInit` to false and does not initialize the object

Even before an object is initialized in a Validate SOAP request, the initial values should be set to both the object and its attributes. This is done using the implicit and explicit event listeners.

To set initial values to an object using implicit listener:

- In the extension class of the object, write the application logic in the `onInitialize` method.

To set initial values to an object using explicit listener:

- Register a class that implements the `IObjectInitializeListener` interface.

To set initial values to an attribute using implicit listener:

- In the extension class, write the application logic in the `onInitialize_<Attribute Name>` method.

To set initial values to an attribute using explicit listener:

- Register a class that implements the `IAttributeInitializeListener` interface.

Constraint logic on objects and attributes

Constraint logic is useful when you want to check the validity of objects and attributes before they are persisted into the database. In a transaction flow, it is necessary to check if the participating objects and attributes are appropriate to the context. Constraint logic on objects checks the validity of multiple attributes in the context of the object, whereas constraint logic on attributes checks if correct values are defined.

To apply constraint logic to the objects and attributes, you can use the constraint listeners. These constraint listeners can be used implicitly or explicitly.

Constraints on attributes can also be set in the BTX file.

To apply constraint logic on an object using implicit listener:

- Use the `onConstraint` listener to write application logic to check for constraints.

To apply constraint logic on an object using explicit listener:

- Register a class that implements the `IObjectConstraintListener` interface, and use the `onObjectConstraint` method to write the application logic.

To apply constraint logic on an attribute using implicit listener:

- Implement the `onConstraint_<Attribute Name>` method in the extension class.

To apply constraint logic on an attribute using explicit listener:

- Register a class that implements the `IAttributeConstraintListener`, and use the `onAttributeConstraint` method to write the application logic.

Even if the constraint logic is set on an object, the constraint check will be performed only if the `qConstraint` attribute for that object is set to 1.

Constraint violations and exceptions

When you validate an object using the `Validate` request, if a constraint is violated, an error message is reported to the method that is called. You can use the `addError` method in the `AttributeConstraintEvent` object to report constraint violations. The error message is included in the XML response as an attribute called `msg-error`. WS-AppServer checks for constraint violations in the entire object and reports the errors collectively.

Alternatively, WS-AppServer reports constraint violations through exceptions. When you use an insert or an update request, WS-AppServer throws an exception (`BsfConstraintViolationException`) as soon as an error is reported.

Repetitive validation

During runtime, when values within an object keep changing as a result of constant updates, the validate request is triggered repetitively until the transaction ends. In such situations, to avoid performance issues, WS-AppServer employs a technique to omit those elements that are already validated and remain unchanged during the rest of the transaction. All such elements are tagged with an attribute called `PreVal`. When the cyclic activity of validation happens, any element that has this attribute is compared with the current value. If both the values are same, WS-AppServer ignores the element and passes over to the next element. However, if it detects that the value of the element with `PreVal` attribute does not match the current value of the element, it triggers the constraint listener. The constraint listener searches for the availability of the changed value and updates the element accordingly. If it does not find the current value, it returns an error message, which is included in the response.

WS-AppServer JAR dependencies

WS-AppServer depends on the following external JAR files. The following table lists the JAR files along with their location information within the `<AppWorks Platform_installdir>`. Ensure that each JAR is placed in its prescribed path.

Name of the JAR	Location (Relative to <code><AppWorks Platform_installdir></code>)
wsappserver.jar	<code><AppWorks Platform_installdir>/components/wsappserver/</code>
managementlib.jar	<code><AppWorks Platform_installdir>/components/managementlib/</code>
esbclient.jar	<code><AppWorks Platform_installdir>/components/esbclient/</code>
ldap.jar	<code><AppWorks Platform_installdir>/ext/</code>
acllib.jar	<code><AppWorks Platform_installdir>/components/acllib/</code>
log4j-1.2.15.jar	<code><AppWorks Platform_installdir>/ext/</code>
basicutil.jar	<code><AppWorks Platform_installdir>/components/basicutil/</code>
eibxml.jar	<code><AppWorks Platform_installdir>/components/eibxml/</code>
jdbccapender.jar	<code><AppWorks Platform_installdir>/ext/</code>
dbconnection.jar	<code><AppWorks Platform_installdir>/components/dbconnection/</code>
ccutil.jar	<code><AppWorks Platform_installdir>/components/ccutil/</code>
jta-1_1-1.0.jar	<code><AppWorks Platform_installdir>/ext/</code>
jdbc driver jar	Default location of the JDBC driver jar

Memory management in WS-AppServer

Default memory management in WS-AppServer

In WS-AppServer, BusObject stores the object data as XML nodes. Each BusObject is a Java object that points to an XML node in NOM. Therefore, memory taken up by WS-AppServer spans both Java as well as NOM.

By default, BusObject is cleaned up by the Java garbage collector when the object is not reachable. If a lot of Java heap is available, the garbage collector will not be aggressive. The Java heap can grow to one-fourth of the physical memory on the computer, while the NOM memory by default can grow only to some extent, which can be set through `bus.xml.vm.maxsize`. When the Java heap size is around 128MB, the JVM may not feel the need to collect all the unreachable memory, as there is no scarcity of Java heap. However, by this time NOM may already reach its memory limit, the client encounters out of memory errors, and the application logic starts failing.

Memory management in WS-AppServer with Auto Cleanup enabled

The Auto Cleanup property can be set while configuring the WS-AppServer service container or in `bsfConfig.xml` in embedded mode. The possible values for Auto Cleanup are true and false where false is the default value.

In WS-AppServer, each BusObject is associated with a BusObjectManager. The BusObjectManager registers all the BusObjects created with it. The BusObject registry is cleaned up when a final commit occurs on the transaction associated with the BusObjectManager. When Auto Cleanup is enabled, WS-AppServer also cleans up the NOM memory by deleting the Busobject XML data associated with the BusObject. When Auto Cleanup is enabled, not all the objects created with that BusObjectManager are accessible after the final commit occurs on that BusObjectManager.

Sometimes applications may need a few BusObjects even after the transaction commit occurs on the BusObjectManager with Auto Cleanup enabled. This can be done by setting `keepAlive()` on BusObject. When a BusObject is set with `keepAlive()`, its BusObject XML data will not be deleted by the WS-AppServer framework with the transaction commit, but will leave it for garbage collection. The BusObject with `keepAlive()` set can be used even after the transaction commit, and until that object is cleaned up by garbage collection.

If Auto Cleanup is set to false, then calling the `keepAlive()` method on the BusObject will have no effect because when Auto Cleanup is set to false, all objects are kept alive anyway.

Running WS-AppServer in multitenant mode

Multitenant refers to a single instance of an application serving multiple client organizations (tenants). The following are the salient features of Multitenant deployment of AppWorks Platform:

- Achieve the desired isolation between tenants by creating a separate organization for each tenant.

- Ensure scalability by sharing resources. In AppWorks Platform, this is achieved by using the same Service Container to cater to multiple tenants.

A WS-AppServer Service Container running in system organization will be able to perform the following tasks:

- Work with multiple databases, one for each tenant if needed.
- Work with multiple versions of the application JAR, one for each tenant if required
- Allow the hosting party to offer different levels of service to different tenants (This feature is restricted to Advanced Properties of a DSO currently).

For managing the new tenants or changing the configuration of the existing tenants, the Service Container need not be restarted.

To run WS-AppServer in multitenant mode:

1. Install the application package.
2. Place the application JARs in the WS-AppServer Deploy folder.
 - a. The `wsapps.properties` file in `CORDYS_INSTALL_DIR/components/wsappserver/config` folder has the property `wsappserver.deploy.folder` that specifies the location of the WS-AppServer Deploy folder.
3. Create a Service Group and Service Container for the application in the System organization.
 - a. Attach the application Web services to the Service group.
 - b. Create a DSO for the application in the System organization and point the Service Container to the DSO. (The DSO should be a valid DSO though it is not used).
4. For a tenant to subscribe to the application, the following steps should be performed:
 - a. Create a DSO with the name of the DSO to which the Service Container points.
 - b. Make the DSO point to the tenant database.
 - c. If the application JARs have been customized, place the JARs specific to the tenant in the WS-AppServer deploy folder. For example if the organization name is Acme then the folder name will be Acme in the WS-AppServer deploy folder.
5. Customize the Advanced Properties of a DSO for the tenant. Customizing the properties involves the following steps:
 - a. Define the levels of the Service in the XML Store in plans file, which is in the Shared space at the location `/Cordys/configurations/`.
 - b. The structure of the XML is as follows.

```
<Levels xmlns="http://schemas.cordys.com/levels">
    <Level name="gold">
        <dso>
            <cursorCacheSize>50</cursorCacheSize>
            <cursorCacheRefreshInterval>30</cursorCacheRefreshInterval>
            <queryCacheSize>50</queryCacheSize>
            <queryCacheRefreshInterval>3600</queryCacheRefreshInterval>
            <maximumWriteConnections>10</maximumWriteConnections>
            <maximumReadConnections>10</maximumReadConnections>
            <minimumReadConnections>1</minimumReadConnections>
            <minimumWriteConnections>1</minimumWriteConnections>
            <connectionPoolRefreshInterval>3600</connectionPoolRefreshInterval>
            <multibyte>true</multibyte>
            <multithreaded>true</multithreaded>
            <sharedPool>true</sharedPool>
            <xsiNilForNullData>true</xsiNilForNullData>
        </dso>
    </Level>
    <Level name="silver">
        <dso>
            <cursorCacheSize>50</cursorCacheSize>
            <cursorCacheRefreshInterval>30</cursorCacheRefreshInterval>
            <queryCacheSize>50</queryCacheSize>
            <queryCacheRefreshInterval>3600</queryCacheRefreshInterval>
            <maximumWriteConnections>5</maximumWriteConnections>
            <maximumReadConnections>5</maximumReadConnections>
            <minimumReadConnections>1</minimumReadConnections>
            <minimumWriteConnections>1</minimumWriteConnections>
            <connectionPoolRefreshInterval>3600</connectionPoolRefreshInterval>
            <multibyte>true</multibyte>
            <multithreaded>true</multithreaded>
            <sharedPool>true</sharedPool>
            <xsiNilForNullData>true</xsiNilForNullData>
        </dso>
    </Level>
    <Level name="platinum">
        <dso>
            <cursorCacheSize>50</cursorCacheSize>
            <cursorCacheRefreshInterval>30</cursorCacheRefreshInterval>
            <queryCacheSize>50</queryCacheSize>
            <queryCacheRefreshInterval>3600</queryCacheRefreshInterval>
            <maximumWriteConnections>15</maximumWriteConnections>
            <maximumReadConnections>15</maximumReadConnections>
            <minimumReadConnections>1</minimumReadConnections>
            <minimumWriteConnections>1</minimumWriteConnections>
            <connectionPoolRefreshInterval>3600</connectionPoolRefreshInterval>
            <multibyte>true</multibyte>
            <multithreaded>true</multithreaded>
            <sharedPool>true</sharedPool>
            <xsiNilForNullData>true</xsiNilForNullData>
        </dso>
    </Level>
</Levels>
```

Currently, there is no User Interface for defining levels.

- c. After defining the levels, Manage Database Configurations screen displays a select box through which the administrator can define the level of the DSO. This will ensure that the Advanced Properties of a DSO are picked from the XML Store.
 - d. If no levels are defined, the Advanced Properties of a DSO are picked from the Service Container.
6. Whenever a tenant is changed, the administrator should invoke the ResetTenantCache Web service in the organization of the tenant.
 7. After any global change that affects all the tenants, the administrator should Reset the Service Container cache.

Setting access control

Access control is about defining accessibility of an object by attaching certain restrictions to the object or its attributes. It is meant to help the user in controlling the behavior of an object or its attributes when they are used in business process flows. Object-level access control defines the operations that can be performed on a particular object and its attributes.

Object-level access control is dynamic in nature. It allows you to determine the access level of an object based on its content. This kind of control is implemented by writing Java logic in the event listeners that are supplied with the WS-AppServer installation. Dynamic access control involves extensive validation of each object in the flow, and may result in reduced performance levels of the application.

Attribute-level access control can be set based on an attribute's properties such as changeability, availability, and its value during runtime. Attribute-level access control is both static and dynamic in nature. The static type of access control is set when the application is being modeled (design-time). This information is applied to the attribute of a particular class. Static attribute access control is implemented by setting the changeability of a particular attribute. The dynamic type of access control is used to set access levels to an attribute of a particular object based on its availability and its value at runtime.

After the system knows who the user is through authentication, authorization is how the system determines what the user can do. This is defined in Access Control Lists (ACLs), where an administrator can set permissions for various resources. Role-based permissions and the privileges of an administrator are examples of configuring permissions.

Access control on objects and attributes can be defined either at the server level or at the client level. Access control defined at the server level controls the operations that are carried out on an object and its attributes, whereas access control defined at the client level determines the display of an object on the client interface (UI).

Setting access control at the server level

You can set access control on objects and attributes to restrict their accessibility in transactions. Access control on objects is set using the `onAccess` event listener, whereas access control on attributes is set using the `onAccess_<Attribute Name>` event listener.

The following access modes are set on objects and attributes transacting at the server:

- Read-only
- Read-write
- Hide

Access control on objects

- To set access control on objects, you can invoke the listener either implicitly or explicitly. However, when you invoke the listener in both ways, precedence is given to implicit listeners.
- To invoke the listener implicitly, use the `ObjectAccessEvent` event in the `onAccess` method.
- To invoke the listener explicitly, register a class that implements the `IObjectAccessListener` interface, and use the `onObjectAccess` method.

Alternatively, you can set access on attributes at the object level, using the `setAccess` method of the `ObjectAccessEvent` class.

The following are different ways in which you can set access on attributes, based on the requirement:

- You can set access on a particular attribute by providing the name of the attribute and the access mode.
- You can set the same access on all the attributes of an object by providing the parameters "null" and the "access mode".
- You can also set different access controls on different attributes of the same object by combining the above two approaches.

Setting the access mode of all attributes in a single instance reduces the number of methods to be defined. It not only minimizes the coding required but also ensures better performance by minimizing the number of methods to be invoked at runtime.

Setting access control at the client level

You can set access control on objects and attributes to determine how they are displayed on the client interface or the UI. Access control on objects is set using the `onDisplay` event listener, whereas access control on attributes is set using the `onDisplay_<Attribute Name>` event listener.

The following access modes are set on objects and attributes to control their display:

- Read-only
- Read-write
- Hide

Access control on objects

- To set access control on objects, you can invoke the listener either implicitly or explicitly. However, when you invoke the listener in both ways, precedence is given to implicit listeners.
- To invoke the listener implicitly, use the `ObjectAccessEvent` event in the `onDisplay` method.
- To invoke the listener explicitly, register a class that implements the `IObjectDisplayListener` interface, and use the `onObjectDisplay` method.

Access control on attributes

- To set access control on attributes, you can invoke the listener either implicitly or explicitly. However, when you invoke the listener in both ways, precedence is given to implicit listeners.
- To invoke the listener implicitly, use the `AttributeAccessEvent` event in the `onAccess_<Attribute Name>` method.
- To invoke the listener explicitly, register a class that implements the `IAttributeAccessListener` interface, and use the `onAttributeAccess` method.

Alternatively, you can set access on attributes at the object level, using the `setAccess` method of the `ObjectAccessEvent` class.

The following are different ways in which you can set access on attributes, based on the requirement:

- You can set access on a particular attribute by providing the name of the attribute and the access mode.
- You can set the same access on all the attributes of an object by providing the parameters "null" and the "access mode".
- You can also set different access controls on different attributes of the same object by combining the above two approaches.

Setting the access mode of all attributes in a single instance reduces the number of methods to be defined. It not only minimizes the coding required but also ensures better performance by minimizing the number of methods to be invoked at runtime.

ACL parameters

The parameters required to configure an access control list are as follows:

- **service:** This is an optional property that denotes the distinguished name (DN) of the Service Group for which the access control object is set. When configured for Web service interfaces and Web service operations, the top level of object in that setting must be the DN of the Service Group, using which the object is referred. When service attribute is not mentioned, the ACL set on Web service interfaces and Web service operations is applied to all the Service Groups.
- **acl:** This property denotes the distinguished name of a user or a role for which the access control is set.
- **acobjecttree:** This property represents the access control list that is set for a user or role. A generic ACL Object Tree can be defined as follows:
`<object acl="open/blocked/condition" id="aclObject"/>`
- The aclObject attribute denotes the object for which the permissions have been set. This can be the DN of a Service Group, labeled URI of a Web service interface, CN of a Web service operation, name of a table, name of a field, key of an XML Store object or DN of an LDAP object.

The acl attribute can be open, blocked, or condition.

The open attribute indicates that the object can be accessed and blocked indicates that the object cannot be accessed. A condition indicates that the object is opened only if the condition defined on it is satisfied. In such cases, the condition can be defined at the object level or at its parent level. The hierarchy of object settings will be followed ifaclattribute is not specified. A lower hierarchy acl object must always be specified within the scope of a higher hierarchy object. For example, consider the following case of setting ACL on a Web service operation inside a Web service interface.

```
<object>
<object acl="blocked" id="Method Set NameSpace">
<object id="Method CN"/>
</object>
</object>
```

In this case, the permission set for the higher level object is applied to the lower level object also.

ACL can be set on Web service interfaces and Web service operations inside the Applications. For such cases, the service attribute is not set and the top level object tag will not have the id attribute.

ACL definitions

The ACL settings in AppWorks Platform allow users to define all the ACL objects in a single tree for a particular user or role. For example, if the ACL is set under metadata services, a single object maintains the ACL settings for tables, fields, related tables, and stored procedures. There is one such object for each metadata service for a particular user or role.

Similarly, when ACL is set at the Service Group level, there is one ACL object per Service Group. At the namespace level, there is a single object for the whole organization, which has ACL for Web service Interfaces and Web service operations.

The following sections describe the ACL definitions for various services available in AppWorks Platform.

Database services

The object hierarchy for Database Services is **Database > Table > Field**.

Here, the ACL for a lower level can be specified as open although the higher level may be blocked. In other words, a field can be open to a user even if the table and the database are blocked.

The following is a sample unconditional ACL object tree that blocks the FirstName field of the Employees table and the EmployeeID field of the Orders table for update and insertion respectively, in the Northwind database.

```
<object condition="false">
<object id="Northwind">
<object id="Employees">
<object id="FirstName">
<method acl="blocked" id="update"/>
</object>
</object>
<object id="Orders">
<object id="EmployeeID">
<method acl="blocked" id="insert"/>
</object>
</object>
</object>
</object>
```

The following is a sample conditional ACL object tree that allows users to update only those records of the Employees table of the Northwind database, where the value in the EmployeeID field is greater than 4.

```
<object condition="true">
<object id="Northwind">
<object id="Employees">
<method acl="condition" id="update">EmployeeID in [3,5] and FirstName =
'Nancy'</method>
</object>
</object>
</object>
```

Database stored procedures

The ACL settings for Database stored procedures/functions is **Database > Stored Procedure Method Name**.

The following is a sample ACL object tree. It is a modified version of the object tree shown above. This blocks the stored procedure `CustOrdersOrders` in the Northwind database.

```
<object>
<object id="Northwind">
<object id="Employees">
<object id="FirstName">
<method acl="blocked" id="update"/>
</object>
</object>
<object id="Orders">
<object id="EmployeeID">
<method acl="blocked" id="insert"/>
</object>
</object>
<object acl="blocked" id="CustOrdersOrders"/>
</object>
</object>
```

XML Store Objects

ACL can be set on files or folders that are located in the XML Store. When ACL is set on a folder in the XML store, the ACL settings are applicable to all files under that folder, unless otherwise specified.

The hierarchy of ACL settings for XML store objects is defined by the structure of the XML Store Explorer and the location of the required object in the XML tree.

ACL at the XML Store level can be set at the object or Web service operation level:

- If a file is blocked at the object level, it is blocked for reading.
- A file can be blocked at the Web service operation level for Read, Update, Insert, or Delete.
- If a folder is blocked at the object level, it is blocked for reading. In this case, the user sees it as an empty folder. However, the objects within the folder can still be open.
- If a folder is blocked at the Web service operation level, the objects within the folder can be open.

The following is a sample ACL object tree, which blocks the Application Package folder of the XML Store. However, the user can read the copyfile object under the Application Package folder.

In the example above, the top-level object inside which the whole ACL tree is set includes an attribute ID that denotes the version of the XML object blocked. As shown, ACL can be set for application, organization, or user versions of the same XML Object.

```
<object>
```

```

<object id="isv">
<object id="cordys">
<object id="wcp">
<object id="menu">
<object id="isv package" acl="blocked">
<object id="copyfile" acl="open">
<method id="update" acl="blocked" />
</object>
</object>
</object>
</object>
</object>
</object>
</object>
</object>
<object id="user">
<object id="cordys">
<object id="wcp">
<object id="menu">
<object id="isv package">
<object id="copyfile">
<method id="read" acl="blocked" />
</object>
</object>
</object>
</object>
</object>
</object>
</object>
</object>
</object>

```

LDAP Objects

ACL can also be implemented for LDAP objects. This can be done in the following ways:

- Object level access control - This creates an acobjecttree from the DN of an LDAP entry
- Object-type level access control - This creates an acobjecttree based on the object class of an LDAP entry For each type of control, the Web service operation can be set appropriately. Currently, this has been implemented for the following LDAP Web service operations: GetLDAPObject
- Update

The following is an example of an acobjecttree in LDAP:

```

<object>
<object id="dn">
<object id="o=acme.com">
<object id="cn=cordys">
<object id="o=system">
<object acl="open" id="cn=organizational users">

```

```

<object id="cn=jsmith">
<method acl="blocked" id="read"/>
</object>
</object>
</object>
</object>
</object>
</object>
<object id="objectclass">
<object id="top">
<method id="read">read</method>
</object>
<object id="busorganizationalobject">
<method id="read"/>
</object>
<object id="busorganizationaluser">
<method id="read"/>
</object>
</object>
</object>

```

Note:

- For LDAP ACL settings, the Object ID (CN of the actual LDAP Object) must be specified in RFC format, that is all letters in lowercase (For example: cn=AppWorks Platform northwind demo 1.0).
- When setting the ACL through metadata services, the top-level object tag need not have an ID attribute, implying that these settings are independent of the Service Groups.
- For ACL settings to function properly, ensure that you set the method tag for any ACL setting immediately below the object tag.

A quick reference to access control

This topic provides information on various access controls.

Access control on objects

The following table describes the access control on objects, and their effect at the client side and the server side. It also shows how access control on objects determines access control on attributes.

Object Level	Attribute Level					
	Read-write		Read-only		Hide	
	Server Access	Client Access	Server Access	Client Access	Server Access	Client Access
Hide	Read	Hide	Read	Hide	Read	Hide
Read-write	Read-write	Read-write	Read	Read	Read	Hide
Read-only	Read	Read	Read	Read	Read	Hide

Consider the first row in the table and assume you have set the access mode for the entire object to "Hide". Accordingly, if you set the access mode of an attribute to "Read-write," then the access mode of the object at the server will be set to "Read." This is because the object-level access mode (in this case, "Hide") takes precedence over the attribute-level access mode. The "Hide" access mode does not apply to server-side objects (since Hide is specific to a user-interface), and therefore, the access mode translates to "Read". However, at the client-level, the access mode is set to "Hide".

In composite objects, access modes can be set on the outer object and the inner object. The access mode that you set on the outer object will always take precedence over the access mode that has been set on the inner object.

Example

```
<purchaseOrder orderDate="1999-10-20">
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>.
  .
</purchaseOrder>
```

In the above object, PurchaseOrder is the outer object, and billTo is the inner object. Assume that you want to set the access mode for the PurchaseOrder object to "Read-only," and set the access mode for billTo object to "Read-write." The "Read-only" access that has been set on the outer object has greater precedence, and therefore, the access mode for the billTo object also is set to "Read-only."

Access control on attributes

The following table describes the access control on attributes, and their effect at the client side and the server side.

Availability Setting	Changeability Setting	Access Logic Setting	Current Value	Server Access	Client Access	
False	-	-	-	Read-only	Hide	Hide
True	Changeable	Read-write	-	Read-write	Read-write	Read-write
True	Changeable	Read-only	-	Read-only	Read-only	Read-only
True	Changeable	Hide	-	Read-only	Hide	Hide
True	AddOnly	Read-write	New/existing+empty	Read-write	Read-write	Read-write
True	AddOnly	Read-only	New/existing+empty	Read-only	Read-only	Read-only
True	AddOnly	Hide	New/existing+empty	Read-only	Hide	Hide
True	AddOnly	Read-write	Existing+filled	Read-only	Read-only	Read-only
True	AddOnly	Read-only	Existing+filled	Read-only	Read-only	Read-only
True	AddOnly	Hide	Existing+filled	Read-only	Hide	Hide
True	Frozen	Read-write	-	New object: Read-write Existing object: Read-	Read-write	Read-only

Availability Setting	Changeability Setting	Access Logic Setting	Current Value	Server Access	Client Access	
				only		
True	Frozen	Read-only	-	Read-only	Read-only	Read-only
True	Frozen	Hide	-	Read-only	Hide	Hide
					New Object	Existing Object

Case 1: Consider the first row in the table above, where the availability of an attribute is set to "false". As a result, the attribute becomes "read-only" when accessed at the server level, and "hidden" at the client level.

If availability of an attribute is set to "false," you cannot set its changeability or write access logic.

Case 2: When the availability of an attribute is set to "True" and changeability to "AddOnly," it indicates that you cannot modify the attribute value once it is set. If the object contains a value, it will be "Read-only" at both the server and the client, even if you set the logic to "Read-write." However, if the object is empty, and you set the access logic to "Read-write," the object will have the "Read-write" state at both the server and the client.

The changeability settings primarily determine the accessibility of an object at the server and the client.

Retrieving access control information

When objects undergo constant change, it is necessary to review their accessibility information from time-to-time so that they are appropriate to the activity in which they participate and reflect correctly on the UI. You can retrieve the access control information of objects and attributes using a query, or an update, or a validate request. The response contains the access control information.

Before you send a request, include the Access attribute and set it to 1.

ACL explorer interface

The following table describes the fields that appear while viewing the access permissions granted for a role.

Description	Description of the ACL
User/Role	Distinguished name of a user or a role for which the access control is set
Name	Name of the ACL

Service Groups	The Service Groups for which the ACL is set
Object Tree	<p>The XML representation of ACL. A generic ACL Object Tree can be defined as follows:</p> <pre><object> <object id="Northwind"> <object id="Customers"> <method acl="blocked" id="Delete"/> </object> </object> </object></pre>

Here, the first object Id (Northwind) refers to Database Metadata, the second object ID (Customers) refers to the Database Table, method acl indicates whether the object is accessible or not, and Id indicates the level of restriction. As per this example, the delete operation is blocked for the role or user.

Single sign-on reference

The following are the libraries and APIs in Single Sign-On.

- [Client-side Single Sign-On library](#) - The Single Sign-On client library that is available for developers creating applications using the AppWorks Platform client framework.
- [Server-side Single Sign-On API](#) - The Single Sign-On API that is available for developers not using the AppWorks Platform client framework to create client applications

Client-side single sign-on library

The AppWorks Platform Web Toolkit contains a client-side library for the Single Sign-On functionality. With this library, a developer can use the following functionality:

- Login and logout with Single Sign-On
- Cache SAML assertions
- Check if the user is logged on
- Retrieve the user name of the logged in user

When the user logs in the username and password are sent to the AppWorks Platform Single Sign-on service container. After they are validated, a SAML Assertion is generated and signed with the SSO certificate. This SAML assertion proves the identity of the logged in user. With the SAML assertion a SAML artifact is generated. This SAML artifact is a reference to the SAML assertion. Both are cached in the front-end sso object. A SAMLart browser session cookie is set that holds the SAML artifact. This session cookie is automatically sent to the AppWorks Platform back-end when a soap request is sent. Note that only the SAML artifact cookie is communicated between the browser and back-end.

The retrieved SAML assertions are, by default, valid for eight hours. Before the assertions become invalid, the AppWorks Platform Single Sign-On client library will automatically request new assertions from the Single Sign-On service.

Usage

Access the SSO library through the `CordysRoot.sso` object.

To get the default configured login UI use:

```
CordysRoot.sso.logMeIn();
```

Show the current logged in username.

```
alert(CordysRoot.sso.getUserName() + "is logged on.");
```

In a standalone page use the following function to get the sso object:

```
function getSSO() { if (!CordysRoot.sso) { CordysRoot.application.addType(CordysRoot, "wcp.library.system.SSO"); } return CordysRoot.sso; }
```

To use client-side single sign-on:

1. When a SOAP request is done, `busdataisland` sends a SOAP request to the web gateway and routed to the addressed service container.
2. When the user is not authenticated and the service requires authentication, it returns a SOAP response indicating that the user is not authenticated.
3. The `busdataisland` calls the `sso` object, which will show a login form, so that the user can enter credentials (user name and password).
4. When the login form is closed, `busdataisland` takes the credentials from the input form. The credentials are used in a SOAP request to the Single Sign-On service group, to get SAML assertions and SAML artifact.
5. The original SOAP request is saved for later use. The SOAP request to the Single Sign-On is a WS-Security request with a SAML AuthenticationQuery.
6. The `busdataisland` instance receives the response from the Single Sign-On service, takes the assertion and artifact out of the response and stores them in the SSO object. The SSO object sets a session cookie with the SAML artifact as content.
7. The `busdataisland` instance resends the original SOAP request. The SAML artifact session cookie is automatically transmitted with the request.

Library functions

The following functions are available in the AppWorks Platform Single Sign-On client library.

- [Single Sign-On library: void logMeIn\(\)](#)
- [Single Sign-On library: boolean login\(\)](#)

- Single Sign-On library: void logout()
- Single Sign-On library: boolean loggedon()
- Single Sign-On library: string getUsername()

Single sign-on library: boolean login()

Performs a login given the userName and password, or a credential structure. Intended to be called for applications that show their own login screens.

Syntax

```
boolean login(userName, password, oCredentials, applicationOrganization)
```

Parameters

Parameter	Description	Data Type
applicationOrganization	This parameter is optional. The organization context to be set after login. If not specified, the default organization of the user is used, otherwise the organization provided is set.	String
oCredentials	Object containing two fields 'userName' and 'password'. This parameter is only used if the other parameters are null.	Object
password	The password of the user that must be authenticated.	String
userName	The name of the user that must be authenticated.	String

Return value

Returns true if the login was successful, false otherwise

If no parameters are provided, false is returned.

Events

The login function raises two events.

Event name	Description
onbeforelogin	Raised before the actual login. Can be used to prepare before login.
onbeforelogout	Raised after the actual login. Can be used to refresh data displayed

Single sign-on library: void logout()

The logout function of the client-side Single Sign-On library is used for logout.

Syntax

```
void logout()
```

Return value

No return value.

Events

The logout function raises two events.

Event Name	Description
onbeforelogout	Raised before the actual logout. Can be used to save data in an application.
onafterlogout	Raised after the actual logout. Can be used to update information displayed about the user.

Single sign-on library: boolean loggedOn()

The loggedOn function of the client-side Single Sign-On library checks if someone is logged in. This function will see if a user is logged in by checking the SAML artifact cookie or SAML assertions or via the SAMLart url parameter.

Syntax

```
boolean loggedOn()
```

Return value

Returns 'True' if someone is logged in, otherwise 'False'.

Note: To retrieve the name of the user logged in, use the function `getUserName`.

Single sign-on library: String getUserName()

The `getUserName` function of the client-side Single Sign-On library gets the name of the currently logged in user.

Syntax

```
String getUserName()
```

Return value

Returns the user name of the currently logged-in user.

Single sign-on library: void logMeIn()

Calling this function will result in displaying the configured Login form. This can be either default AppWorks Platform login form or that of an external Identity Provider.

Syntax

```
boolean logMeIn()
```

Example

Use the function as follows:

```
CordysRoot.sso.logMeIn()
```

Events

As the logMeIn function triggers a login UI the following events are raised.

Event name	Description
onbeforelogin	Raised before the actual login. Can be used to prepare before login.
onafterlogin	Raised after the actual login. Can be used for example to catch the user login name.
onbeforelogout	Raised before the actual logout. Can be used to ask the user to Save data before logout.
onafterlogout	Raised after the actual logout. Can be used to refresh data displayed.

Server-side single sign-on API

A SOAP request has to specify the credentials of the sender of the SOAP request. Any identity type supported by AppWorks Platform can be used. For information about the supported identity types, refer to Identity Types in *AppWorks Platform Administration Guide*.

When the AppWorks Platform Single Sign-On solution is used, the Single Sign-On API can be used. It contains a Web service operation to request SAML assertions to be used in subsequent SOAP requests. For more information, refer to Single Sign-On API: Request.

Single sign-on API: request

The Header must contain a WS-Security user name token.

The Body must contain the Request tag with

`namespace:urn:oasis:names:tc:SAML:1.0:protocol`. The Request tag must contain the Authentication Query specifying the user for which the SAML assertions are requested.

For an example on SOAP messages, see Example SOAP Messages for SAML Authentication in the *AppWorks Platform Administration Guide*. In this example, the Header contains a WS-Security user name token for user scrat with clear text password. The body specifies scrat as the user for which SAML assertions are requested. The SOAP response contains either the requested SAML assertions or a SOAP Fault.

SAML assertions returned by the RequestWeb service operation can be used in subsequent SOAP requests.

Certificate

A certificate is a guarantee that a public key belongs to a certain owner. This guarantee is given by a Certificate Authority (CA), which signs the certificate so that the public key cannot be altered or forged. The certificate authority needs to verify the identity of the owner of the public key before issuing a certificate. If there is no certificate authority, a certificate owner can sign its own certificate. These certificates are called self-signed certificates.

A certificate consists of:

- serial number
- public key
- identity of the certificate authority that signed the certificate
- expiration dates
- and other data associated with the identity

A public key has a corresponding private key, which must be held secret by the owner.

Web application development

This section contains the following topic:

- [Overview of AppWorks Platform Ajax Toolkit](#)

Working with alert system

An alert is any 'notable occurrence' which might be of interest to an operator (possibly automated) of AppWorks Platform, and which can be detected and described programmatically. Alerts are meant for the administrators of AppWorks Platform. Some co

Some common examples of alerts used in AppWorks Platform are:

- Service has started or stopped or crashed
- There is an unauthorized access to the system
- Connection with the database could not be established

Alert System in AppWorks Platform has been designed to monitor the health of software. Through the Alert System an operator or an administrator of a system is informed of undesirable situations such as exceptions, errors, overload on a part of the system and so on. With such a mechanism in place, operators can pro-actively respond to deviations in performance and optimize the system usage, thereby preventing the error condition to spread to the other parts of the system. The Alert System provides the functionality to define an alert on a managed component and issue it when the need arises.

The alert system issues, captures, processes and forwards alerts to various consumers. A consumer is a program that processes the alerts appropriately.

The alert system in AppWorks Platform has the following features:

- Defines the standard structure of alerts
- Provides an interface using which the AppWorks Platform and AppWorks Platform-based applications issue 'alerts'
- Possesses criteria to filter alerts
- Routes the alerts to consumers for specific purposes
- Issues alerts as JMX notification
- Manages the descriptive text within the messages
- Provides alerts in default culture-neutral language

For more information on alerts, visit log4j manual. This section includes the following topics:

- [Generating Alerts](#)
- [Generating Log Messages from Repository](#)
- [Defining Alerts](#)
- [Publishing Alerts](#)

Generating alerts

The process of creating and using alerts involves a series of steps right from defining the structure for alerts, to having an interface for using them and routing them to appropriate consumers. The alert system mechanism is capable of managing culture-specific descriptive text within the messages.

Before you begin:

- Developers are the intended audience for this document.

To define and use alerts:

1. Define alerts using a Message Repository, which is an XML file (example, `Cordys.Integrator.Messages.xml`) and place the same in the `<AppWorks Platform Installation Directory>\<localization folder>`.
2. Generate the corresponding Message class using the Message Generator tool. See [Generating Messages from Repository](#) for more information.

3. When the message classes are generated, use them to [define AppWorks Platform alerts](#) on a managed component and [publish](#) them.

Generating log messages from repository

The AppWorks Platform framework provides a tool called `com.eibus.tools.internal.MessageGenerator` that generates message bundles from the `<Message Repository>.xml` file.

To generate log message from repository:

1. Provide the input parameters and run the tool `com.eibus.tools.internal.MessageGenerator`.
 - a. For the `com.eibus.tools.internal.MessageGenerator` tool, specify the alert repository file name and the fully qualified name of the class to be generated (this includes the package to which the class belongs).
 - b. Set the Classpath, Path and `Process_Platform_install_dir` in the following format:

Windows	<ul style="list-style-type: none"> ■ Set <code>Process_Platform_install_dir = <AppWorks Platform_installdir></code> ■ Set <code>CLASSPATH = %CLASSPATH% ; <AppWorks Platform_installdir>/Cordys.jar</code> ■ Set <code>PATH = % PATH% ; <AppWorks Platform_installdir> / Lib</code>
Linux	<ul style="list-style-type: none"> ■ Export <code>Process_Platform_install_dir = <AppWorks Platform_installdir></code> ■ * Export <code>CLASSPATH = \$CLASSPATH : <AppWorks Platform_installdir>/Cordys.jar</code> ■ * Export <code>PATH = \$PATH : <AppWorks Platform_installdir> / Lib</code>

- c. Run the Message Generator tool to generate the `.java` file.

This file is generated in the directory, from where you run the tool. If you mention only the class name as an input parameter, the package will not be defined.

A sample code to generate the Java file is given below:

```
java com.eibus.tools.internal.MessageGenerator <Messagedefinitions.xml> <Fully qualified classname>
Example: java com.eibus.tools.internal.MessageGenerator
Cordys.ESBServer.LogMessages.xml
com.eibus.localization.message.internal.LogMessages
```

Here, `Cordys.ESBServer.LogMessages.xml` is the LogMessage repository file name while `com.eibus.localization.message.internal.LogMessages` is the name of the class to be generated. The `LogMessages.java` file containing the source is generated.

2. Run the `LogMessages.java` from the command prompt.

Here's an example:

```
javac -d . LogMessages.java
```

The logMessages class file is generated.

3. Modify the classpath to include the class file.
4. Copy the <message repository>.xml file to <AppWorks Platform_installdir>\<localization folder>.

Defining AppWorks Platform alerts

An alert on AppWorks Platform framework is defined on a managed component during its creation. This implies that all the alerts on a particular managed component are defined before registering the component. After a managed component is registered, a new alert cannot be created on it.

Sample code for an alert definition

The following sample code shows how to define an alert on a managed component.

```
{ //Creation of managed component ManagedComponent m_managedComponent =
ManagedProcess.createManagedComponent('SOAPProcessor',description); . . . // defining
the alert for LDAP server failure on the managed component IAlertDefinition s_
monitorStartLdapFailureAlert = m_managedComponent.defineAlert(AlertLevel.ERROR,
Messages.MONITOR_START_ERROR_LDAP_FAILURE, Messages.MONITOR_START_ERROR_LDAP_
FAILURE_DESC); // define all alerts . . . //registration of component m_
managedComponent.registerComponentTree(); }
```

Publishing alerts

Alerts are published by defining them on managed components and invoking the issueAlerts() Web service operation on it. The following is sample code for issuing an alert.

```
try { } catch(LDAPException ldapEx) { // raise the alert defined earlier s_
monitorStartLdapFailureAlert.issueAlert(ldapEx); }
```

Additional parameters can also be passed if the alert messages in the XML file have parameter placeholders in the form of {0}, {1}, and so on.

This can be done using another form of issueAlert() API such as
`alertDefinition.issueAlert(exception, object array);`

Once the alert is issued, it gets published to the consumers configured in the Log4jConfiguration.xml. For example, if the consumer configured is a file appender such as aProcessNamedDailyRollingFileAppender, then all the alerts will be published to appropriate files in <AppWorks Platform_installdir>\logs directory.

A sample alert message published would be:

```
<log4j:event logger='com.eibus.management.AlertSystem' timestamp='1138694792285'
level='ERROR' thread='main'> <log4j:message><! [CDATA[<LocalizableMessage
resourceID='Cordys.Integrator.Messages.monitorStartErrorLdapFailure> The OpenText
AppWorks Platform (&lt;instance name&gt;) could not be started due to failure to
access the LDAP Server. Please ensure that the LDAP server is running. If so, the
machine could be slow. Add the 'bus.ldaprequest.timeout=<timeout>' property in the
wcp.properties file and specify the duration for request time
out.</LocalizableMessage>]]></log4j:message> <log4j:NDC><! [CDATA[host=CIN0389
processid=3172 messageid=__Unknown__ user=__Unknown__ component=__Unknown__
__]]></log4j:NDC> <log4j:throwable><! [CDATA[LDAPEXception: Connect Error (91) Connect
Error LDAPEXception: Server Message: Connect Error LDAPEXception: Unable to connect
to server cin0389:6,366 (91) Connect Error java.net.ConnectException: Connection
refused: connect at com.eibus.applicationconnector.ldap.LDAPWrapper.<init>
(LDAPWrapper.java:72) at
com.eibus.applicationconnector.monitor.Launcher.isLDAPServerReady(Launcher.java:370)
at com.eibus.applicationconnector.monitor.MonitorService.isLDAPServerReady
(MonitorService.java:21) Caused by: LDAPEXception: Unable to connect to server
cin0389:6,366 (91) Connect Error]]></log4j:throwable> <log4j:locationInfo
class='com.eibus.management.AlertSystem' method='issueAlert' file='AlertSystem.java'
line='23'> </log4j:event>
```

Working with logging framework

The logging framework is a key feature of AppWorks Platform that records certain critical activities during runtime. It is based on Log4j and is used for debugging by developers and support personnel.

The information gathered from the log files is used to analyze the error state and fix support calls. AppWorks Platform supports different kinds of log consumers such as file, event service, and so on, including those from Log4j. This framework provides a web-based log viewer for viewing the messages. Apache's Chainsaw is an open source GUI-based tool that can also be used for viewing the log messages. Composite Application Logging (CAL) is an extension of Logging Framework that facilitates application level logging.

Logging framework helps the support personnel or developers or administrators to analyze the error states. Refer to Logging Framework Process for information on the overall working of logging framework.

Developer's Best Practices provides some basic guidelines on using logging framework.

Examples of log messages

Examples of log statements are:

- License key has been validated.
- Request has been received.
- Response has been sent.
- An exception occurred.

This section contains the following topics:

- [Using the Logging Framework](#) - This topic explains how to instrument the logging framework.
- [Generating Log Messages](#) - This topic provides information on how to create and use the localized log messages.

Using logging framework

A developer can use the logging framework by inserting log messages of the required severity in the existing source code (shown in the example below) , and executing it when the option to log is enabled. Log statements are published to selected consumers (also known as publishers or appenders), based on the option selected during configuration. See the following example for logging a string message.

```

import com.eibus.soap.ApplicationTransaction;
import com.eibus.soap.BodyBlock;
import com.eibus.util.logger.CordysLogger;
import com.eibus.util.logger.Severity;
import com.eibus.xml.nom.Node;
public class Transaction
    implements ApplicationTransaction
{
    static CordysLogger srLogger =
CordysLogger.getCordysLogger(Transaction.class);
//getCordysLogger Web service operation takes the same class as the argument
    public void commit()
    {
    }
    public void abort()
    {
    }
    public boolean canProcess(String type)
    {
        return 'serviceType'.equals(type);
    }
    public boolean process(BodyBlock requestBodyBlock, BodyBlock
responseBodyBlock)
    {
        boolean invoked =false;
        try
        {
            int response = responseBodyBlock.getXMLNode();
            int request = requestBodyBlock.getXMLNode();
            if (srLogger.isInfoEnabled())
                srLogger.log(Severity.INFO,'Request received ..\n'
+Node.writeToString(request,true));
            //
            // implementation here
            //
            if (srLogger.isInfoEnabled())
                srLogger.log(Severity.INFO,'Response from the Service is ..\n'

```

```
+Node.writeString(response,true));
}
catch (Exception ex)
{
srLogger.log(Severity.ERROR,'exception',ex);
}
return invoked;
}
}
```

For more information on other methods, see the SDK Logging IStringResource.

The class containing log messages (IStringResource) has to be created before using this API. Messages can be logged using the ID of the message.

The MessageID returns com.eibus.localization.message.Message object. This is an implementation of the IStringResource.

See [Generating Log Messages](#) for creating the IStringResource.

Example:

```
import com.eibus.localization.message.internal.LogMessages;
import com.eibus.soap.ApplicationTransaction;
import com.eibus.soap.BodyBlock;
import com.eibus.util.logger.CordysLogger;
import com.eibus.util.logger.Severity;
import com.eibus.xml.nom.Node;
public class Transaction
implements ApplicationTransaction
{
static CordysLogger srLogger =
CordysLogger.getCordysLogger(Transaction.class);
//getCordysLogger Web service operation takes the same class as the argument
public void commit()
{
}
public void abort()
{
}
public boolean canProcess(String type)
{
return 'serviceType'.equals(type);
}
public boolean process(BodyBlock requestBodyBlock, BodyBlock
responseBodyBlock)
{
boolean invoked =false;
try
{
if (srLogger.isInfoEnabled())
srLogger.info(LogMessages.METHOD_INVOKED);
```

```

// Method Implementation here
//
//
}
catch (Exception ex)
{
srLogger.error(ex, LogMessages.METHOD_INVOKE_ERROR);
}
return invoked;
}
}

```

`LogMessages. METHOD_INVOKED & LogMessages. METHOD_INVOKE_ERROR` returns the Message objects (Implementation of `IStringResource`) corresponding to the ID. For more information on other methods, please refer to SDK.

How Messages Are Logged

Messages are logged to the selected consumer with the class name as Category.

Generating log messages

The process of creating and using log messages involves a series of steps right from defining the structure for them, to having an interface for using and routing them to appropriate consumers. Also, the log messages mechanism is capable of managing culture-specific descriptive text within the messages. The process of defining and using the log messages follows.

To generate log messages:

1. Define log messages using a Message Repository, which is an XML file and place the file in the `<AppWorks Platform_installdir>\<localization folder>`.
2. Generate the corresponding message class using the Message Generator tool.
See [Generating Log Messages from Repository](#) for more details.
3. Use the generated to log `LocalizableLogMessage`.

Here's an example:

```

import com.eibus.util.logger.CordysLogger; import
com.eibus.localization.message.internal.LogMessages; public class
LocalizableLogMessageSample { static CordysLogger logger =
CordysLogger.getCordysLogger(LocalizableLogMessageSample.class); /** * This Web
service operation Logs the LocalizableLogMessage to CordysLogger. */ public void
loggerTest() { try { //some operation here logger.info(LogMessages.METHOD_INVOKED);
} catch(Exception ex) { logger.error(ex,LogMessages.METHOD_INVOKE_ERROR); } }

```

Best practices for developers using logging framework

The following guidelines can help a developer to use the logging framework:

- After the logging instance is obtained, it should be made as a static object. The same instance must be used subsequently.

```
public class LoggerDemo1 {  
    private static final CordysLogger logger = CordysLogger.getCordysLogger  
(LoggerDemo1.class);  
  
    public void methodOne()  
    {  
    }  
  
}
```

- Before using the logger, check whether the selected category is enabled with specified severity. This can save a performance penalty for composing the arguments. Do not do this for error or fatal.

The error and fatal severities are configured for logging by default, so there is no value in adding calls to `isErrorEnabled()` and `isFatalEnabled()`; it only clutters the code.

- It is a best practice to make the log messages translatable (the nontranslatable APIs are deprecated).

```
public class LoggerDemo2 {  
    static CordysLogger logger = CordysLogger.getCordysLogger(LoggerDemo2.class);  
  
    public void methodZero()  
    {  
        int node = 0; // get a NOM node  
        if (logger.isDebugEnabled())  
        {  
            logger.debug(node);  
            logger.debug("Hi there"); // don't use translateable messages for debugging,  
            those are meant for (peer) developers.  
        }  
    }  
  
    public void methodOne()  
    {  
        if (logger.isInfoEnabled())  
        {  
            logger.info(Messages.INFORM_ADMINISTRATOR);  
        }  
    }  
  
    public void methodThree()  
    {  
        try  
        {  
            // do something...  
        }  
    }  
}
```

```

}
catch (Exception e)
{
logger.error(e, Messages.ERROR_WHILE_THREE);
}
}

public void methodFour()
{
if (logger.isInfoEnabled())
{
logger.info(Messages.INFORM_STATUS, status); // use insertions
}
}
}

```

- If the logger is used to log any exception based messages, it must pass the whole exception to the logger; not just the message:

```

public class LoggerDemo3 {
static CordysLogger logger = CordysLogger.getCordysLogger(LoggerDemo3.class);

public void methodOne()
{
try
{
// do something...
}
catch (Exception e)
{
logger.error(e, Messages.ERROR_WHILE_THREE);
// Don't do this: logger.error(null, Messages.ERROR_WHILE_THREE, e.getMessage());
}
}
}

```

Working with problem registry

Service containers can face issues like a lost database connection that make it impossible for them to handle requests. In such a situation, the service container can enter the so-called problem state, indicating it cannot handle client requests. This is particularly useful in a system that is configured for high availability: if a service container flags a problem, the routing algorithm responds to that and will not deliver requests to the container facing the problem, but distribute the work over the remaining containers that do not face a problem.

The system will periodically call back to the component that registered the problem, to see whether the problem can be resolved now, for instance by trying to connect to the database again. If the problem is resolved, the service container will automatically leave the problem state and thus receive requests again. This is called self-healing, also known as autonomous computing.

This feature is implemented through the Problem Registry, a framework that helps application developers register and unregister problems. This framework is based on State SyncUp, which distributes the registered problems across all service containers. These problems can also be retrieved programmatically.

All problems are classified as CRITICAL, that is, there are no hierarchy or severity levels for problems. When a service container enters or leaves the problem state, it issues an alert (see Understanding Alert System), to which administrators/operators can respond.

An example of working with the Problem Registry is as follows:

- A service on Machine A detects loss of connection with the database. It registers a problem.
- The web gateway gets a new request for the service. It detects that the service instance on Machine A has a problem, so it passes the request to another instance of the service.
- When the database connection is restored, the service instance on Machine A unregisters the reported problem.
- The web gateway resumes forwarding requests to the service instance on Machine A.

This section includes the following topics:

- [Problem Registry for Database Connectors](#)
- [Using Problem Registry](#)

Problem registry for database connectors

The following is ensured by the Problem Registry framework when it is implemented for the database connectors.

- When a database server goes down, all JDBC Service Containers using that database server detect the loss of database connection. The detection occurs when the request is being processed. Once the loss of connection is detected, a database connection failure problem is registered.
- All related Service Containers register the database connection problem.
- The client ensures that the request is not sent to the Service Containers that have registered the problem.
 - To use this feature:
 - The SOAP request must contain the Service Group DN as the target
 - Use mirror databases

- After the problem is registered by the JDBC Service Container, the connector attempts to connect to the database in two-minute intervals.

Note for the application developers

The following points are for all the application developers who use the DBConnectionPool API in the com.eibus.applicationconnector.sql package:

- A utility called DBProblemMonitor is provided in com.eibus.applicationconnector.sql package, which uses the DBConnectionPool in the same package to connect to the database server. This utility can be used to monitor database connection failure problems.
- Application developers can also extend this class to detect and resolve domain-specific problems in addition to database connection failure issues.

Using problem registry

To use the Problem Registry framework, the application developer of a component (application connector) must ensure that the component:

- defines the potential problem through the related managed component and
- raises the problem in case it occurs.

Sample code for defining a potential problem

A potential problem is defined on the managed component. The defineProblem operation takes four parameters, related to the two alerts that are issued when a problem is raised or cleared. The first pair of messages defines the message and belonging description of the error alert issued when a problem is raised. The second pair relates to the informational alert issued when the problem is cleared.

```
IProblemDefinition problemDefinition = managedComponent.defineProblem(
    MyMessages.PROBLEM_RAISE_ALERT_MESSAGE, MyMessages.PROBLEM_RAISE_ALERT_DESCRIPTION,
    MyMessages.PROBLEM_CLEAR_ALERT_MESSAGE, MyMessages.PROBLEM_CLEAR_ALERT_DESCRIPTION)
```

The return value is a problem definition that should be used to raise the problem when it occurs.

Sample code for raising an actual problem

When a problem occurs (e.g. upon a ConnectionException) the code should call raiseProblem on the problem definition:

```
problemDefinition.raiseProblem(new MyProblemResolver(connection), 30000, e, hostName);
```

The problem resolver (see below) is a piece of code that is responsible for resolving the problem, for instance by reconnecting. This resolver is called after the specified interval has passed. The optional exception is included in the alert error log message and the given parameter (hostName in our example) is given as insertion for the message of that alert.

Sample code for implementing the problem resolution

The previous `raiseProblem` call consumes a problem resolver. This is a custom implementation of the interface `IProblemResolver`. This object is responsible for resolving the problem, for instance by reestablishing the connection. This is a simple example:

```
import com.eibus.management.IProblemResolver;
import com.eibus.management.IProblemStatusEventListener;

/**
 * Example handler to resolve a connection problem.
 */
class MyProblemResolver implements IProblemResolver
{
    private final MyConnection connection;

    /**
     * Instantiate a new problem resolver.
     *
     * @param connection the connection to be reestablished
     */
    MyProblemResolver(MyConnection connection) {
        this.connection = connection;
    }

    @Override
    public void resolveProblem(IProblemStatusEventListener
problemStatusEventListener) {
        try {
            connection.reconnect();

            /*
             * A problem is only considered resolved after notifyResolved is invoked
on the IProblemStatusEventListener
            */
            problemStatusEventListener.notifyResolved(connection.getHostName());
            return;
        }
        catch (ConnectionException e) {
            // Optionally log reconnect failure
        }

        /*
         * Indicate interval after which the problem registry needs to check for
resolution again
        */
        problemStatusEventListener.notifyNotResolved(30000);
    }
}
```

The effect of the code on this page is that when the application code detects a problem (flagged by `problemDefinition.raiseProblem`), the reconnect method of the connection will be repeatedly called, on an interval of 30 seconds, until the reconnect is successful. At the time the problem is registered, an error alert is issued and the service container enters the problem state. This will cause the load balancing algorithm to redirect all traffic to other service containers in the same service group (provided they exist and do not also face problems). After the problem is resolved (flagged by `problemStatusEventListener.notifyResolved`), an informational alert is issued and the service container leaves the problem state. From then onward, the routing algorithm will again deliver requests to it.

Reading WSDL from other locations

In addition to LDAP Store, AppWorks Platform facilitates reading WSDL from other locations such as CoBOC and file system.

To read WSDL from other locations:

1. Place WSDL Reader class under the package `com.eibus.tools.wsdl.<storename>` that should implement the interface `com.eibus.tools.wsdl.IWSDLAction`.
`<storename>` should be in lower case. For e.g., to read WSDL stored in CoBOC, the implementation class name should be `com.eibus.tools.wsdl.coboc.WSDLReader`, `<storename>` is assumed as coboc.
2. Ensure that the WSDL URL to be read will have one more query string parameter, `cwsdlstore` that will have the store name from which the WSDL is read.
For example, to read WSDL stored in CoBOC, the gateway URL will be:
`http://<machineName>/cordys/com.eibus.web.tools.wsdl.WSDLGateway.wcp?cwsdlstore=coboc&<required query string>`
When the `cwsdlstore` attribute is not part of the URL, WSDL is read from the default store LDAP.

Invoking AppWorks Platform web services externally

AppWorks Platform Web services can be invoked from an external environment through WSDL Gateway. The business logic in AppWorks Platform is contained in the Web service operations that are encapsulated within Web service interface. The implementation and interface of AppWorks Platform Web service operations is contained in WSDL that is exposed as a Web service. These Web services can be accessed from outside AppWorks Platform using the URLs specified below.

To read data from Explorer, specify the WSDL gateway URL, where service would be the `<namespace>/<Web service operation to be read>`.

Access Web service operation or Web service interface	URL Format
Access the WSDL for an Web service operation	<code>http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.WSDLGateway.wcp?service=<namespace>/<Web service operation to be read></code>
Access the WSDL for a Web service interface	<p>for application Web service interface - <code>http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.WSDLGateway.wcp?service=http:<namespace>/&version=isv*</code></p> <p>for Custom Web service interface - <code>http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.WSDLGateway.wcp?service=http:<namespace>/*</code></p>

Note: If the port number specified above is the default HTTP port then specifying the port number is optional.

Following query string parameters can be added to the URL:

Parameters	Description
Organization	It takes the DN of the organization in which the Web service operation or Web service interface is present. If it is not specified, then it takes user's default organization.
Version	Specify 'isv', if the Web service operation or Web service interface is present in the application. If it is not specified, it reads from the organization.
resolveexternals	It takes a boolean value. To read imported or included schemas set it to True. The default value is false.
multipleschema	It takes a boolean value. Set False, to keep all the schemas with same target Namespace under single schema element. The default value is True. Set this parameter, only if, you need to access WSDL of all Web service operations.

Example

To read the WSDL of 'GetEmployees' Web service operation present in the Web service interface with namespace '`http://schemas.cordys.com/Employees`', in an organization the URL will be:

```
http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.WSDLGateway.wcp?service=http://schemas.cordys.com/Employees/GetEmployees&organization=dn
```

To read the included or imported schemas along with the WSDL, the URL will be:

```
http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.
WSDLGateway.wcp?service=http://schemas.cordys.com/Employees/GetEmployees&organization=<organization dn>&resolveexternals=true
```

To read the WSDL of 'GetEmployees' Web service operation present in 'Web service interface Northwind' with namespace 'http://schemas.cordys.com/1.0/demo/northwind' of AppWorks Platform Northwind Demo 1.0 , the URL will be:

```
http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.
WSDLGateway.wcp?service=http://schemas.cordys.com/Employees/GetEmployees&organization=<organization dn>&version=isv
```

To read all WSDL Web service operations present in the Web service interface with namespace 'http://schemas.cordys.com/Employees', in an organization, the URL will be:

```
http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.
WSDLGateway.wcp?service=http://schemas.cordys.com/Employees/*&organization=<organization dn>
```

The WSDL is displayed when the multipleschema parameter has the default value ('True').

If the URL is:

```
http://<machinename>:<port number>/cordys/com.eibus.web.tools.wsdl.
WSDLGateway.wcp?service=http://schemas.cordys.com/Employees/*&organization=<organization dn>&multipleschema=false
```

The WSDL is displayed when the multipleschema parameter is set to 'False'.

Note: com.eibus.web.tools.wsdl.WSDLGateway.wcp can also be replaced with WSDLGateway.wcp.

Exploring web service and its child entities

Web service has a hierarchy of child elements, which together provide the required functionality. There are couple of ways to access each child element and you can also perform certain activities on each one of these.

The following table provides all the necessary details.

Child Element	Accessing	Purpose	Allowed Actions
Web service interface (Child of Web service)	<ul style="list-style-type: none"> ■ In Workspace Documents (Explorer), expand the Web service. You can see the Web service interface. Double-click to open it. ■ In Workspace 	<ul style="list-style-type: none"> ■ A container for holding the Web service operations ■ Contains information regarding the implementation class of the Web service 	<ul style="list-style-type: none"> ■ Validate and Publish to the organization ■ Attach to the Service Group so that the Web service operations use that Service

Child Element	Accessing	Purpose	Allowed Actions
	<p>Documents (My Recent Documents), double-click the Web service and click  on the toolbar of the Web service Properties window. You can see the Web service interface in a pane on the right side. Double-click to open it.</p>		
Web service operation (Child of Web service interface)	<ul style="list-style-type: none"> ■ In Workspace Documents (Explorer), expand the Web service interface. You can see the Web service operation. Double-click to open it. ■ In Workspace Documents (Explorer), double-click the Web service interface and click  on the toolbar of the Web service Binding window. You can see the Web service operation in a pane on the right side. Double-click to open it. 	Contains the method implementation	<ul style="list-style-type: none"> ■ Validate and Publish to the organization ■ View WSDL ■ Test Web service operation

Event service

AppWorks Platform contains the provision for monitoring the activities of any entity within the AppWorks Platform framework. An entity can be any resource in the framework. When any party within AppWorks Platform wants to monitor the activities occurring in an entity, it can do so by subscribing to these activities using event service. This is called Event subscription. The requesting party is called the Event subscriber. The entity whose activities are being monitored is called subject. Every activity occurring in the subject is published to the subscriber. The activities occurring in the subject are called events.

Event Service publishes all the events that occur over a subject within AppWorks Platform to the subscriber. It maintains a list of subscribers per subject. The user can choose to subscribe or unsubscribe to a subject. If the event service fails to deliver an event to any of the subscribers, that subscriber is immediately unsubscribed. An example of event subscription is loading applications using Application Manager. As the installation of the application progresses, many messages are displayed at the bottom of the screen. These messages usually indicate the status of the installation and these messages are called events.

Another usage of event service is the mail exchange client. The user need not perform any special operation to check if a new e-mail is received after opening the Inbox. New messages alerts are automatically displayed to the user.

Event Service is also very useful while debugging in AppWorks Platform.

Event service also persists subscriber information, that is, it maintains the subscriber information in the `wcp.properties` file as a collection per subject.

Event service connector

AppWorks Platform provides you with the 'Event Handling Service Connector' while creating service containers to work with event service.

The 'Event Service' application connector publishes all the events that occur in AppWorks Platform.

Eventservice in Ajax Toolkit

AppWorks Platform also provides you with Eventservice APIs that can be used in the User Interface (UI). These APIs can be used while developing applications to take advantage of the event service feature.

Impact of event service failure

The following applications require the services of event service:

- Inbox
- CWS

The following table describes the impact on each application in case of a failure.

Application	Impact	Workaround
Inbox	Push Mode view of Inbox uses Event Service. In case of event service failure, no messages will be received in Inbox.	Click Refresh icon on the Inbox toolbar to view the latest tasks
Collaborative Workspace	Updates for documents are not pushed to the browser anymore.	Manually refresh the workspace and editors.

Overview of DOM

According to W3C, Document Object Model (DOM) is a language and domain independent interface that allows programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page.

AppWorks Platform uses Native Object Model (NOM) as the XML infrastructure. Native Object Model (NOM) is a customized DOM implementation in AppWorks Platform. It does not follow all the DOM specifications.

The characteristics of NOM are as follows:

- Uses tree model to represent XML in memory
- Native objects are used in the tree model, making it faster than Java object tree
- Exposes the native handlers for Java developers, resulting in better performance than that of Java objects
- Deleted XML nodes are put in free-list for reuse, resulting in better performance
- For creating nodes, Virtual memory is used instead of Heap memory again resulting in faster performance.

NOM provides better performance. But it has no provision for automatic garbage collection. Deleting a node two times caused problems. To avoid these problems, AppWorks Platform provides you with wrapper around NOM. This wrapper plugs the gaps of NOM. It acts as a cover around NOM so that the developer believes he or she is working with DOM.

The following are the features of DOM wrapper implemented in AppWorks Platform:

- DOM implementation: AppWorks Platform implements level 3 core interfaces defined by W3C. For more information, see [DOM level 3 Core specifications](#).
- Automatic Memory Management: Users do not have to take care of memory management. The framework manages garbage collection. However, if you do want to take care of memory management then AppWorks Platform provides with this option too. You may want to explicitly clean up memory in cases of heavy load. For more information, see NOMNodeclass in AppWorks Platform SDK documentation.
- Double Delete: Deleting a node more than once results in an exception. This helps developers to handle such situations during development itself rather than troubleshooting it later.

- DOM to NOM and NOM to DOM conversions: There are provisions to convert a DOM node into NOM node and vice versa. For more information, see NOMNode and DOMFactory classes in com.cordys.xml.dom package in AppWorks Platform SDK documentation.

Hooking the implementation into Java API for XML Parsing (JAXP)

To hook DOM wrapper into JAXP:

- Set javax.xml.parsers.DocumentBuilderFactory property to com.cordys.xml.dom.internal.DocumentBuilderFactoryImpl as shown during the startup of the application (probably in a static block).

```
System.setProperty("javax.xml.parsers.DocumentBuilderFactory",
"com.cordys.xml.dom.internal.DocumentBuilderFactoryImpl");
```

Setting this property and loading of com.cordys.xml.dom.internal.DocumentBuilderFactory will also cause the corresponding JAXP-XPath implementation to be loaded. See JAXP-XPath implementation.

JAXP-XPath implementation (javax.xml.xpath)

As a supplementary for the DOM implementation, there is an implementation of JAXP-XPath package, which is the primary mechanism for executing XPath expressions on DOM trees. When DOM wrapper is used, by default this implementation is also loaded.

To hook the XPath implementation without loading it:

- Set javax.xml.xpath.XPathFactory: http://java.sun.com/jaxp/xpath/dom property to the value com.cordys.xml.dom.internal.XPathFactoryImpl as shown.

```
System.setProperty
("JAXP.XML.XPATH.XPATHFACTORY:HTTP://JAVA.SUN.COM/JAXP/XPATH/DOM",
"COM.CORDYS.XML.DOM.INTERNAL.XPATHFACTORYIMPL");
```

Example

The following example demonstrates the usage of DOM wrapper over NOM.

```
import java.io.ByteArrayInputStream; import java.io.IOException; import
java.io.UnsupportedEncodingException; import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory; import
javax.xml.parsers.ParserConfigurationException; import javax.xml.xpath.XPath; import
javax.xml.xpath.XPathConstants; import javax.xml.xpath.XPathExpression; import
javax.xml.xpath.XPathExpressionException; import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document; import org.w3c.dom.Node; import org.w3c.dom.NodeList;
```

```

import org.xml.sax.SAXException; import com.cordys.xml.dom.DOMFactory; import
com.cordys.xml.dom.NOMNode; import com.eibus.xml.nom.XMLException; public class
Example { static { System.setProperty("javax.xml.parsers.DocumentBuilderFactory",
"com.cordys.xml.dom.internal.DocumentBuilderFactoryImpl"); } private static String
xmlStr = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" + "<employees>" +
"<employee>" + " <name>e1</name>" + " </employee>" + " <employee>" + "
<name>e2</name>" + " </employee>" + "</employees>"; private static
com.eibus.xml.nom.Document nomDoc = new com.eibus.xml.nom.Document(); private static
void traverseAndPrint(Node node ) { if(node != null) { if(node.getNodeType() ==
Node.TEXT_NODE) { System.out.println(node.getNodeValue()); } else {
System.out.println(node.getNodeName()); } } } public static Document parseXML(String
xml) throws SAXException, IOException ,ParserConfigurationException {
DocumentBuilderFactory dbfactory = DocumentBuilderFactory.newInstance();
DocumentBuilder parser = dbfactory.newDocumentBuilder(); ByteArrayInputStream is =
new ByteArrayInputStream(xml.getBytes()); Document doc = parser.parse(is);
traverseAndPrint(doc); return doc; } public static void evaluateXPathAndPrint
(Document doc , String xpathExp) throws XPathExpressionException { XPathFactory
xpathFactory = XPathFactory.newInstance(); XPath xpath = xpathFactory.newXPath();
String xpe1 = xpathExp; XPathExpression employeesXPath = xpath.compile(xpe1);
NodeList nl = (NodeList) employeesXPath.evaluate(doc,XPathConstants.NODESET); for
(int i = 0 ; i < nl.getLength(); i++) { System.out.println(nl.item(i).getNodeName()
()); } } public static void wrapDocument() throws
XMLException,UnsupportedEncodingException { int nomNode = nomDoc.parseString
(xmlStr); org.w3c.dom.Node domDoc = DOMFactory.wrapDocument(nomNode); /* 'nomNode'
need not be deleted, it is under DOM-Wrapper memory management. As when docDoc gets
garbage collected, native 'nomNode' also will be deleted */ System.out.println
(domDoc.getNodeName()); } public static void wrapNode() throws
XMLException,UnsupportedEncodingException { int nomNode = nomDoc.parseString
(xmlStr); org.w3c.dom.Node domNode = DOMFactory.wrapNode(nomNode); /* This is just a
DOM representation of the passed in NOM Node. Memory management is still with NOM
and the nomNode needs to be deleted explicitly */ System.out.println
(domNode.getNodeName()); com.eibus.xml.nom.Node.delete(nomNode); } public static
void getNOMNNode(Document doc) throws XMLException { org.w3c.dom.Node fChild =
doc.getFirstChild(); int fChildNOM = ((NOMNode)fChild).getNOMNode();
System.out.println(com.eibus.xml.nom.Node.getName(fChildNOM)); /* Memory management
is still with DOM and the retrieved node need not be deleted*/ } public static void
unwrapDoc(Document doc) throws XMLException { int fChildNOM = ((NOMNode)doc).unwrap
(); /* Now fChildNOM is not under DOM memory management and should be deleted */
System.out.println(com.eibus.xml.nom.Node.getName(fChildNOM));
com.eibus.xml.nom.Node.delete(fChildNOM); } public static void main(String [] args)
{ try { System.out.println("-----PARSING XML-----");
" ); Document doc = parseXML(xmlStr); System.out.println("-----");
" ); System.out.println("-----XPath evaluation -----"); evaluateXPathAndPrint
(doc,"/employees/employee"); System.out.println("-----");
" ); System.out.println("----- Wrapping NOM
node under DOM mem management--"); wrapDocument(); System.out.println("-----");
" ); System.out.println("-----");
" ); wrapNode(); System.out.println("-----");
" ); System.out.println("-----");
" ); System.out.println("-----Unwrapping DOM node-----");
" ); getNOMNNode(doc); System.out.println("-----");

```

XML search patterns

AppWorks Platform includes a library for XML object manipulation. This library is written in C, but has been made available to Java developers as well, through the Java Native Interface. For performance related reasons, XML Object pointers are declared of type Integer in Java code. Object construction in Java is much more expensive than memory allocation in C. Both the Java library and the C library support not only an object model on top of XML documents, but also a powerful document search functionality. The object model exposes functions such as `getTagName()`, `getFirstChild()`, `createElement()` on an XML Document. The search functionality allows complex document queries. For example you can query for the first child with the element name 'author' and an attribute called 'name' with the value 'George Orwell'.

Match patterns

All Match functions have two arguments. The first argument passed is the root node from which the search is started and the second argument is a string containing the search pattern. The table below contains the syntactic elements that are valid for the search pattern. You can use them in any combination. The match function that you use determines whether it should return the first matching node or all matching nodes.

Syntax	Meaning
<code><[name]></code>	Find a child element that has name as tagname.
<code><[name] [attr]></code>	Find a child element that has name as tagname and an attribute with name attr.
<code><[name] [attr]='[val]'></code>	Find a child element that has name as tagname and an attribute with name attr and value val.
<code><[name]>'[val]'</code>	Find a child element with the specified name that has a textnode child that has the specified value.
<code>.</code>	Separation mark for pattern-elements; ?, <, >, ', and ^ are also (implicit) separation marks.
<code>?</code>	Any child node.
<code>^<[name]></code>	Search up for the first parent that has a tagname with name.
<code>parent</code>	Find the parent element.
<code>fChild</code>	Find the first child.
<code>lChild</code>	Find the last child.
<code>right</code>	Find the next (right) sibling.
<code>left</code>	Find the previous (left) sibling

Example

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Header>
        <header xmlns="">
            <sender>
                <component>Anonymous Simple Client</component>
                <reply-to>zipsocket://10.192.61.190:1154/</reply-to>
                <user>cn=jsmith,cn=organizational
users,o=system,cn=cordys,o=acme.com</user>
            </sender>
            <receiver>
                <component>cn=XML Store Service,cn=soap
nodes,o=system,cn=cordys,o=acme.com</component>
            </receiver>
            <msg-id>guidstring</msg-id>
        </header>
    </SOAP:Header>
    <SOAP:Body>
        <GetXMLObject xmlns="http://schemas.cordys.com/1.0/xmlstore">
            <key version="isv">file1</key>
        </GetXMLObject>
    </SOAP:Body>
</SOAP:Envelope>

```

Code	Explanation
<SOAP:Envelope><SOAP:Body><GetXMLObject><key version='isv'>	Find the key node that has value ISVP for the version attribute in the specified pattern. This will return a pointer to the key node.
<SOAP:Envelope><SOAP:Body>?<key>fChild	Find the key node that has any parent, that is a grand child of SOAP:Body and that is a great-grandchild of SOAP:Envelope. This will return a pointer to the textnode file1. You can get to this data with the function getData().
<SOAP:Envelope>?<msg-id>'guidstring'parent.parent	Returns a pointer to the header node
<SOAP:Envelope>?<msg-id>'guidstring'^<SOAP:Envelope><SOAP:Body>	Returns a pointer to the SOAP:Body node
<SOAP:Envelope>?<msg-id>'guidstring'parent.parent.parent.right	Returns a pointer to the SOAP:Body node

Additional features for querying a database

This topic describes the additional features provided in AppWorks Platform for querying a database.

parameterlist - This feature allows the request to contain multiple values for a single parameter. This is useful in case of dynamic IN clause queries. Consider the following implementation that queries Employees Table with a single parameter.

```

<implementation type="DBSQL">
    <constructor language="DBSQL">
        <query>SELECT * FROM Employees WHERE EmployeeID = :EmployeeID</query>
        <parameterlist>
            <parameters>
                <EmployeeID dd="Employees.EmployeeID" dt="i4"/>
            </parameters>
        </parameterlist>
    </constructor>
</implementation>
<SOAP:Body>

```

The request for such a query is as follows.

```

<soap:envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<GetEmployee xmlns="http://schemas.cordys.com/1.0/demo/Eastwind">
    <parameters>
        <EmployeeID>1</EmployeeID>
    </parameters>
    <parameters>
        <EmployeeID>2</EmployeeID>
    </parameters>
</GetEmployee><soap:envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Body>

```

This feature offers the following advantages:

- Network Roundtrips for each execution to the bus request can be avoided
- Lessens the executions and provides results to the user at an optimal time

Dynamic IN Clause- Consider the following query that has an IN clause querying for the Employees table.

```
<query>SELECT * FROM Employees WHERE EmployeeID IN (:EmployeeID)</query>
```

You can execute the query for multiple values of EmployeeID through the OLEWs-AppServer Service Container. The number of values is dynamic and programmatically generated. In such cases, where the IN clause is not used, users have to call a Web service operation for each of the obtained values that results in one request for each execution and more number of network roundtrips. Also, the execution time increases and more time is taken to return the result. This problem can be avoided by passing multiple values in the <parameter> tag as follows.

```

<parameters>
    <EmployeeID>
        <value>1</value>

```

```

<value>2</value>
<value>3</value>
<value>4</value>
</EmployeeID>
</parameters>
```

The request can be fired successfully without containing the <parameters> tags. In the example, the query is executed only once and the result is returned for all the four values in a single execution. This can also be used for more than one IN clause as shown in the following query.

```
<query>SELECT * FROM Employees WHERE EmployeeID IN (:EmployeeID) AND City IN (:City)</query>
```

The parameters are passed for the given query as follows.

```

<parameters>
    <EmployeeID>
        <value>1</value>
        <value>2</value>
        <value>3</value>
    </EmployeeID>
    <City>
        <value>Seattle</value>
        <value>Tacoma</value>
    </City>
</parameters>
```

Parameters with attribute information - The implementation created using Method Generator contains the parameter information for each parameter sent as an attribute to the back-end. Hence, the developer for each request does not need to supply this information. The information about the parameter is generated as follows.

```

<implementation type="DBSQL">
    <constructor language="DBSQL">
        <query>SELECT * FROM Employees WHERE EmployeeID BETWEEN :fromEmployeeID AND
:toEmployeeID</query>
        <parameters>
            <fromEmployeeID dd="Employees.EmployeeID"/>
            <toEmployeeID dd="Employees.EmployeeID"/>
        </parameters>
    </constructor>
</implementation>
```

The implementation type is DBSQL, and the constructor language is DBSQL (for tables) and DBRPC (for stored procedures).

- **Support for table names with spaces**- Tables with spaces in the table names are also supported in AppWorks Platform.
- **Performance improvement with no-reply option** - When working with bulk inserts, the performance is slowed down due to the presence of auto-generated fields in the response or due to responses being generated after every read/insert. The performance of insert operation can be improved by using the reply option. By setting reply='no' attribute for the 'Update' tag in the implementation, you can avoid auto-generated responses and improve the connector performance.

Application status categories

The Applications Signature Verification Status page of Application Installer wizard displays the list of Applications being loaded along with associated status, as follows.

Valid	A legitimate or genuine Application.
Tampered	<p>Signed Application(s) that are altered. This includes addition, removal, and update of the files in the Application. Note:</p> <p>To view the details:</p> <ol style="list-style-type: none"> 1. Click the status term beside the Application name. The Verification Details dialog box opens. This box lists the files that are added, deleted or modified and the location of the files. 2. Point over a file to view its file path. It indicates relative path of the file located within the packaged application.
Untrusted	<p>Application(s) are signed using certificates that are not in the trust store. The signing certificate and its corresponding Certificate chain are not configured in the trust store.</p> <p>To view the details:</p> <ul style="list-style-type: none"> ■ Click the status term beside the Application name. The Certificate Details dialog box opens displaying the signing certificate details.
Unsigned	Application(s) that are not signed using a certificate.
Improper Certificate	<p>Applications are signed by an invalid certificate. The reasons for invalidity are listed below:</p> <ul style="list-style-type: none"> ■ Certificate is Expired ■ Certificate is Revoked ■ Certificate chain is not found in Trust Store ■ Invalid Certificates in its Certificate chain ■ Does not have 'code signing' extended key usage <p>To view the details:</p> <ul style="list-style-type: none"> ■ Click the status term beside the Application(s) name.

	The Certificate Details dialog box opens displaying the signing certificate details.
--	--

Application verification scenarios

The Applications Signature Verification Status page of the Application Installer wizard displays the list of Applications being loaded with their associated status such as Valid, Tampered, Untrusted, Unsigned, and Improper Certificate.

The check box next to the name of the Application is enabled or disabled depending on its validity. The Security settings of the Applications govern the behavior of installer to proceed or abort the procedure.

The following table explains the possible scenarios of Application Installation:

If check box is . . .	Is check box editable?	Result	Reason
Selected	No	Applications are installed uninterrupted	Applications are valid The security option of the security setting, against which the application validation has failed, is set to Allow.
Cleared	Yes	You can select to proceed with the installation	Applications are tampered, untrusted, unsigned or signed with an invalid certificate. The security option of the security setting, against which the application validation has failed, is set to Prompt.
Cleared	No	Applications are not installed	Applications are tampered, untrusted, unsigned or signed with an invalid certificate. The security option of the security setting, against which the application validation has failed, is set to Disallow.

Binding template interface

This topic describes the binding template interface.

General information

Description	optional	Description of the Web service operation being published.
-------------	----------	---

Category bag

The category bag element allows the binding templates to be categorized according to any of the available taxonomy-based classification schemes.

Key Name	optional	Key name further describes the key value.
Key Value	optional	The key value is the identifier within the categorization.
tModel Key	optional	Denotes the information about common forms of categorizing the binding templates.

Cell object property

A cell object is an object that contains the combined properties of a cell, such as properties of the data, the HTML, and the status of the cell.

The following methods are available for the cell object retrieved using the `getCell()` method.

Method	Description
<code>getRow()</code>	Retrieves the row object of the cell.
<code>getColumn()</code>	Retrieves the column object of the cell.
<code>getValue()</code>	Retrieves the value of the cell.
<code>setValue(newValue)</code>	Sets the value of the cell. <ul style="list-style-type: none"> ■ <code>newValue</code>: Refers to the new value of the cell.

The properties of the cell object are as follows:

Property	Description
<code>checkRowColumn</code>	Boolean. Set as true, if the cell is part of the check row column; set as false, if not.
<code>dataNode</code>	Refers to the XML node with the data of this cell.
<code>ref</code>	Refers to the XPath of the cell. The XPath to the data Node is considered from the business Object of the cell.

Configuration parameters for custom application connectors

General tab

Name	Specifies the name of the Application Connector. Example: <code>com.cordys.mypack.CountConnector</code>
------	--

Description	Describes the Application Connector Example: com.cordys.mypack.CountConnector
Implementation Class	Specifies the fully qualified Java class name which contains the implementation of the Application Connector (com.eibus.soap.ApplicationConnector) Example: com.cordys.mypack.CountConnector
Property Page	Specifies the name of the HTML page which accepts configuration inputs for the application connector. You have to upload the related HTML page to refer the details. Example: Appcon.htm
Image	Specifies the name of the image that is displayed in the configuration page. You have to upload the related image file to refer the details. Example: configurationconnector.png
Supports JVM sharing	Specifies whether the application connector can share a JVM or not. If this check box is selected, it indicates that other application connectors can also be hosted in the same Service Container, to which the application connector is configured. <ul style="list-style-type: none"> ■ True - Can share a JVM ■ False - Cannot share a JVM
Use custom classpath	This refers to the Service Container classpath that is identified while defining an application connector. By selecting custom classpath, you can choose only the necessary .jar files and the location of these files can be used as classpath entries. These classpath entries must be relative to the <AppWorks Platform_installdir> separated by commas. Service Containers configured for the application connector use these entries in their classpath. For example, if an entry is abc/xyz.jar,<classfile_location>, the classpath of the Service Container contains <AppWorks Platform_installdir>/abc/xyz.jar;<AppWorks Platform_installdir>/<classfile_location>. Click  and provide the location in the empty row that is added to the table. Example: components/Appconn/Appconn.jar cordyscp.jar

Logging Categories tab

AppWorks Platform provides certain default logger categories and trace levels. Apart from these, you can also specify some custom categories that have to be published.

- Click  and provide the following details:
 - **Name** - Specify the name of the category group, for example, Relay Connector
 - **Category** - Specify the name of the class for which the logs must be recorded, for example, com.cordys.relay.snapper

Start up Dependencies tab

Specifies the list of dependent namespaces for the Service Container to start.	Click  and type the namespace in the empty row that is added to the table. Example: <code>http://schemas.cordys.com/1.0/xmlstore</code>
--	--

To initialize a set of Service Containers in batches, it is necessary to define a dependency list. This list is associated with the application connector, hence is a part of the Application Connector configuration.

Configuring advanced search parameters

Advanced Search refines your search for Web services.

1. In the **Explore Web Services - UDDI Browser** dialog box, expand the **Advanced Options** group box.
The Advanced Options section is displayed.
2. In **Maximum display results**, type a value for the number of results to be displayed.
3. Set the following search criteria.
 - **Exact phrase** - To match the search term exactly.
 - **Case sensitive** - To enable case sensitivity of the search term.
4. From the **Select a categorization scheme** list, select a scheme.

Note: A categorization scheme is a predefined set of categories, derived from an internal or external hierarchy, which can be used to classify a service. Categories under this scheme are displayed in a grid and you can search for Web services in a specific category.

5. Select the categories and click **OK**.
6. From the **Identifier** list, select an identifier and provide a **Value**.

Note: Identification schemes provide an additional means of grouping entities together and enable the discovery of logically grouped entities when searching.

The search criteria are set for an advanced search.

Configuring application connectors

Configuring application connectors will allow a Service Container to connect to multiple applications or back ends.

Before you begin:

- You must have the role of `systemAdmin` or `organizationalAdmin` to configure application connectors.

To configure application connectors:

1. On the Welcome page > My Applications, click  (System Resource Manager). The System Resource Manager window opens and lists the available Service Containers in the Service Containers App Palette.
2. Select the Service Container you intend to modify and double-click it. Alternatively, you can right-click a Service Container and select Properties option. Corresponding properties are displayed in the Properties - <Service Container Name> App Palette. The associated connectors are displayed in the Application Connectors area.
3. Click  in the Application Connectors area. The Select Application Connectors dialog box opens with the list of application connectors.
4. Select an application connector and click **OK**. This is added as <Application Connectors> tab in the Service Properties - <Service Name> App Palette. Additionally, it is added to the list of connectors in the Application Connectors area.
5. You can click the corresponding tab to configure a connector. See the topic *Configuration Parameters for Application Connectors* in the *AppWorks Platform Administration Guide* for information on configuring the application connectors.
6. Alternatively, select the connector from the Application Connectors area and double-click it. The corresponding <Application Connectors> tab is activated.

An Application Connector is configured to a Service Container.

To delete an application connector:

- Select it from the Application Connectors area and click .
- You can add or delete multiple application connectors.

Configuring applications for anonymous usage

Before you begin:

- Determine which Web service operations your application will use.

An application can be used by anonymous users when all the Web service operations the application uses can be used by the user 'anonymous' in AppWorks Platform. This can be done by applying the appropriate ACL at the Web service operation level.

Make sure your application is accessible to anonymous users by creating a virtual directory with anonymous access in the Web server.

1. On the Welcome page > My Applications, click  (System Resource Manager).
The System Resource Manager window opens.
2. Click .
The list of service groups is displayed.
3. Perform the following steps for each Web service operation that your application uses:
 - a. Right-click the <**Service group**> and click **Security**.
 - b. Click **Add**
The Organizational Users/Roles dialog box opens.
 - c. Select anonymous role.
4. Click **OK**.
5. Click **X** to close the System Resource Manager window.

The application is configured for anonymous usage.

Using APIs for database access

Before you begin:

All database requests in AppWorks Platform need to follow a standard protocol. The XQY API is a component in AppWorks Platform that handles the AppWorks Platform database protocol. This XQY API is accessible using class

`com.eibus.applicationconnector.sql.DBConnectionPool`. An instance of `DBConnectionPool` represents a pool of connections to the database via XQY API. This pool exposes different APIs to route requests to the database. This allows the user to contact the database directly in the AppWorks Platform environment. It also decreases the response time and allows users to perform dynamic query execution.

To use APIs for database access:

1. Create a connection pool.

The public Java class `DBConnectionPool` present in the `com.eibus.applicationconnector.sql` package can be used to create a connection pool by supplying all the details required to connect to the DB server in XML form.

```
String dsoXml = null;
dsoXml = <configuration>
<update-connections>10</update-connections>
```

```

<read-connections>10</read-connections>
<dso jdbcDriver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
      driver="JDBC"
      connectionString="jdbc:sqlserver://<<server>>:<<port>>" 
      defaultDB="<<DB name>>" 
      userId="<<user>>" 
      xmlencoding="false" 
      password="<<base64 encoded password>>" 
      update="true" >
<query-cache>
  <size>50</size>
  <refresh-interval>3600</refresh-interval>
</query-cache>
<cursor-cache>
  <size>50</size>
  <refresh-interval>3600</refresh-interval>
</cursor-cache>
</dso>
<connection-pool>
  <min-update-connections>1</min-update-connections>
  <min-read-connections>1</min-read-connections>
  <refresh-interval>3600</refresh-interval>
</connection-pool>
</configuration>";
int dso = xmlDoc.parseString(dsoXml);
DBConnectionPool pool = DBConnectionPool._createInstance(dso, null, null, null);

```

This will create a read and an update connection in the pool. For information about `createInstance`, refer to Java SDK documentation.

2. Compose the database request.

The database request, as required, can be composed using the following Web service Operations.

Method	Description
query	Reads data from the database
update	Updates data in the database
getMetaData	Retrieves the metadata of the database
validateCommand	Validates a query against the database
commitCurrentTransaction	Commits the current transaction to the database
abortCurrentTransaction	Aborts the current transaction
executeDDL	Executes the input DDL against the database

To query the database:

```
<dataset>
    <constructor language="DBSQL">
        <query>select * from Employees where EmployeeID = :EmployeeID</query>
        <parameters>
            <EmployeeID dd="Employees.EmployeeID">1</EmployeeID>
        </parameters>
    </constructor>
</dataset>
```

To retrieve the metadata of the database:

```
<dataset>
    <metadata requestType="getTableInfo" tableCatalog="Northwind"
              tableName="Employees" tableSchema="" tableType="TABLE"/>
</dataset>
```

To validate a query against the database:

```
<implementation>
    <validate>
        <query> select * from Employees where EmployeeID = :EmployeeID</query>
    </validate>
</implementation>
```

To execute an input DDL against the database:

```
<implementation>
    <executeDDL>create table temp_table (field1 int primary key, field2 varchar
(20))</executeDDL>
</implementation>
```

To update the database:

```
<update>
    <tuple>
        <old>
            <Employees>
                <EmployeeID>1</EmployeeID>
                <FirstName>Nancy</FirstName>
            </Employees>
        </old>
        <new>
            <Employees>
                <EmployeeID>1</EmployeeID>
                <FirstName>Nancy123</FirstName>
            </Employees>
        </new>
    </tuple>
</update>
```

To execute an input DML against the database:

```
<implementation>
    <executeDML>
        <DML>
            <command> INSERT INTO Employees(FirstName, LastName, Title, City) values
(:FirstName, :LastName, :Title, :City) </command>
            <parameters>
                <FirstName dd="Employees.FirstName">Test</FirstName>
                <LastName dd="Employees.LastName">Test</LastName>
                <Title dd="Employees.Title">Test</Title>
                <City dd="Employees.City">Test</City>
            </parameters>
        </DML>
    </executeDML>
</implementation>
```

3. Get an appropriate connection from the pool.

The connection pool contains both read and write connections. Read connections can be used to process requests like query, metadata, and validate. Write connections can be used to process requests like update records, DDL request, and DML requests where transaction is necessary. Based on the request to be processed, applications can use appropriate APIs to get a read or updateconnection.

```
WCPDBConnection con = null;
con = pool.getReadConnection();
OR
con = pool.getWriteConnection();
```

4. Using appropriate APIs of WCPDBConnection class, send a request to the DB Server.
This class provides the following APIs to execute the request.

Method	Description
query	Reads data from the database. Pass the <constructor> node xml to query API
update	Updates data in the database
getMetaData	Retrieves the metadata of the database
validateCommand	Validates a query against the database
commitCurrentTransaction	Commits the current transaction to the database
abortCurrentTransaction	Aborts the current transaction
executeDDL	Executes the input DDL against the database
executeDML	Executes the input DML against the database

Method	Description
createAuditTable	Creates an audit table from an existing table
purgeAuditTable	Removes the contents of audit table

5. Analyze the return value of the database response.

When the database requests are sent, you will receive database responses. The database responses are in XML format. Based on the return value, you can determine whether a request is processed successfully or an error occurred. If the return value is 0, then the request is processed successfully. If it is less than zero, then the request execution resulted in an error. Based on the return value, the application can either commit or abort the transaction for the operation that needs transaction support.

```
If(ret <0)
{
// error happened
con.abortTransaction(); // for write connections
}
else
{
// Processing is success
con.commitTransaction();
}
```

The following sample responses are received in case of success and in case of failure.

```
Success
<dataset>
<constructor language="DBSQL">
<query>select EmployeeID, FirstName, LastName from Employees where
EmployeeID = :EmployeeID</query>
<parameters>
<EmployeeID
dd="Employees.EmployeeID">1</EmployeeID>
</parameters>
</constructor>
<tuple>
<old>
<Employees>
<EmployeeID>1</EmployeeID>
<FirstName>Nancy123</FirstName>
<LastName>Davolio</LastName>
</Employees>
</old>
</tuple>
</dataset>

Error
```

```

<dataset>
    <constructor language="DBSQL">
        <query>select EmployeeID, FirstName, LastName1 from Employees where
EmployeeID = :EmployeeID</query>
        <parameters>
            <EmployeeID
                dd="Employees.EmployeeID">1</EmployeeID>
        </parameters>
    </constructor>
    <error
        TYPE="Enumeration">
        <elem>ColumnsRowset formatting problem</elem>
        <elem>Failed to setQueryMetaData()</elem>
    </error>
</dataset>

```

6. Place the connection back into the pool.

After the processing is done, it is mandatory to place the connection back into the pool. Otherwise, this connection cannot be used for any other request processing. You can do this by using Web service Operations provided in DBConnectionPool class.

```
Pool.putReadConnection(con); OR Pool.putWriteConnection(con);
```

Conversion methods

The conversion methods that can be defined for globalization component needs to be different from the conventional methods in the sense that the data as well as conversion factors are extensible at run time. A unique method written to handle this case will not be possible, since it depends on various attributes like conversion base, factor of conversion etc. This becomes complex particularly if the conversion is to be extended for more number of data (For example, 100 or more).

To solve this problem, a generic algorithm is written in the component that can convert the data from one form to another form without extending the basic definition.

Currency conversion methods

The currency data can be reached through the property currency of the globalization component. To convert the currency in one format to other format, the syntax defined will be as follows.

Scripting	nConvertedUnit = globalID.currency.convertCurrency(sSourceName, sTargetName, nUnits)
-----------	--

Parameters

sSourceName	Required. String that denotes the source unit name.
sTargetName	Required. String that denotes the target unit name to convert to.
nUnits	Required. Integer that denotes the number of units to convert.

Return Value

Returns an integer that denotes the source unit (nUnits) converted to target unit.

Example

The sample below will convert the Indian currency to Dutch currency.

```
var output = globalID.currency.convertCurrency("INR", "NLG", 100)
```

Other conversion methods

The convention for calling other conversion methods is defined as follows:

Scripting	nConvertedUnit = globalID.sUnitType.convertsSourceUnitToTargetUnits (nUnits)
-----------	---

Parameters

sUnitType	Required. String that denotes the source unit type to convert to. This can be distance, weight, volume or any defined unit of measurement.
sSourceUnit	Required. String that denotes the source unit to convert to. This unit should be defined in the corresponding unit type.
sTargetUnit	Required. String that denotes the target unit to which the source unit is to be converted.
nUnits	Required. Integer that denotes the units to convert.

Return value

Returns an integer that denotes the source unit (nUnits) converted to target unit of the given unitType.

Example

The sample below converts the weight in kilograms to tonnes.

```
var output = globalID.weight.convertkgTotonne(10)
```

The output returned would be 0.01 in this case, as the equivalent of kg in tonnes is 0.001.

The following sample converts the temperature in Celsius to Fahrenheit.

```
var output = global.temperature.convertcelsiusTofahrenheit(36.5)
```

The output in this case would be 97.7.

Creating a custom desktop

Before you begin:

- Ensure that your application is accessible to anonymous users by creating a virtual directory with anonymous access in the Web server.

A custom desktop is a Web page that can be integrated in AppWorks Platform. Custom desktops can have the following levels of integration with AppWorks Platform:

- **No integration:** When the custom desktop is not integrated with AppWorks Platform, you can access AppWorks Platform data in various ways. The desktop can be created with various server and client-side languages. To retrieve the data from AppWorks Platform, you must use SOAP implementations which are located on the client-side, for example, in JavaScript. It is also possible to use a SOAP implementation on the server-side.
- **Medium integration:** Implementing medium integration can be done using a AppWorks Platform JavaScript library on the client-side. This JavaScript library *busdataisland* provides connectivity with AppWorks Platform. On the AppWorks Platform Developer Network (CDN), you can find more documentation on how to call communication functions from *busdataisland*.
- **Full integration:**
 - The desktop is a customized version of the AppWorks Platform desktop that is `cusp.caf`. Applications must be AppWorks Platform application as defined in the Web Toolkit. This level of desktop integration has all the capabilities of the Explorer. To implement this, you can copy the installed `cusp.caf` to a new location and customize it.
 - Use the default Cordys `cusp.caf` as desktop, and add the file entry in XMLStore Explorer - Collection\CORDYS\WCP\Desktop\Application to start your custom desktop applications automatically or check the Launch on StartUp of its XForm.

You cannot associate applications (or the desktop) to users or an anonymous user by default. The user identity is checked when a Web service operation is accessed or when some data is retrieved. So by default, anything is accessible to the anonymous user until a Web service operation is called that contains an ACL specifying that anonymous access is not allowed. By not defining such an ACL, all the data is available for anonymous access. If an ACL is defined on the first Web service operation called, then nothing is accessible anonymously.

Creating a Java class metadata

Before you begin:

- Ensure that a Java Archive (.JAR) file or a Java class (.CLASS) file is available on the machine where AppWorks Platform is installed.

This option helps you map your custom Java application files into AppWorks Platform in a compatible format. After this mapping (generation of Java Class Metadata), you can generate Web services on them.

To create a Java class metadata:

1. Select a starting point and click  to create a Java Class Metadata.
The Java Class Metadata window opens.
2. Do one of the following:
 - To map a Java Archive (.JAR) file to this model, provide the full path of its location, in the space provided.
 - To use individual Java class files (.CLASS), provide their fully qualified classname in the space provided.
3. Ensure that the class files are placed the classpath.
4. Click **Finish**.
The Save Document dialog box opens.
5. Provide a Name, Description, and a location to save the model, and click **Save**.
The Save Document dialog box closes.
6. Close the Java Class Metadata window.

The Java Class Metadata is created and is available for viewing at the saved location.

After you complete this task:

1. If you modify the contents of .JAR, you need to update the Java Class Metadata.
 - a. Right-click  (Java Class Metadata).
 - b. Select **Reload Java Class Metadata**.
 - c. Publish the Java Class Metadata to an organization.

Creating an XML

Creating an XML helps you to bundle XML content with the application that you are building. You need to create the document using the XML structure that you have and save it with .xml extension.

The interface supports both writing the XML structure or copying and pasting a predefined XML structure from an external source. However, ensure that the file name with which you save the document ends with .xml extension.

You can create an XML file at the project level or in a folder within the project. If you create an XML file within a folder, you can set Qualified Name (QN) on that folder to exclusively publish its contents to XML Store.

To create an XML:

1. Select a starting point and click  to open the XML editor.
2. Type the XML content in the provided space and click .

CAUTION: Ensure that the XML is valid. Otherwise it may not work as expected.

The Save Document dialog box opens.

3. Provide the **Name** and **Description**, and click to browse and select the location to save the XML file.
The folder path where the XML will be created appears in the Save in Folder field.
4. Click **Save**.
The Save Document dialog box closes.
5. Close the XML window.

The XML is created with the provided name and is stored at the selected location.

After you complete this task:

- Create an XML Store Definition that bundles the XML in a suitable format so that it can be packaged with the application and deployed to be used at run time. It can then be viewed using the XML Store Explorer.

Creating an XML store definition

Before you begin:

- You must have valid XML files in your project.

To create an XML store definition:

1. Select a starting point and click  to open the XML Store Definition editor.
2. Enter the following information.

Name	Provide a Name of the XML Store Definition file.
Description	Provide a Description of the XML Store Definition.
XML Store-content folder	<p>The folder where the XML files are available and need to be published to run time. These XML files will be used in the Application Package.</p> <p>Next to the field, click  to browse and select the source content project/folder.</p> <p>Note: The documents in this folder will be validated and published as per the publish logic of XML Store Definition.</p>

3. Click .
- The Save Document dialog box closes.
4. Close the XML Store Definition window.

The XML Store Definition is created with the provided name and is stored at the selected location. When you publish it, the included XML files are made available in the XML Store. When you create an Application Package, the XML files are packaged along with and are available for use at run time when the package is installed.

Note: When deploying an application that contains an XML Store Definition (or when publishing such an artifact), the documents in the corresponding XML Store content folder will be deployed in the XMLStore. The exact location at which the documents will be deployed is based on their qualified name. Only valid XML documents will be published to the XMLStore.

Creating AppWorks Platform toolbar

To create a AppWorks Platform toolbar:

1. Right-click the toolbar and select the **New > Toolbar** option.
The New Toolbar - Toolbar dialog box opens.
2. Type the name, caption, and description for the new toolbar and click **OK**.
The new toolbar can be seen in the Explorer.
You can customize a toolbar by adding new items.

The context menu of the toolbar contains multiple options for customization.

Custom connector configuration interface

The parameters required to configure Custom Connectors appear while creating a Service Group.

To configure custom connectors:

1. In the New Service Group wizard, in the Select Application Connectors page, select **Other**.
2. To save the details provided for the Data field in the XML format, select **Store as XML**.

Implementation Class	The fully qualified Java class name that contains the implementation of the Application Connector.
Data	The additional directories or jar files for the Service Container classpath to start the Service Container. The OS-specific classpath delimiters must separate these entries.

Customizing the AppWorks Platform toolbar

You can customize the AppWorks Platform toolbar by adding a new toolbar item or a toolbar.

To customize the AppWorks Platform toolbar:

1. Right-click the toolbar and select **New > Toolbar Item**.
The New Toolbar Item window opens.
2. To create a new toolbar, right-click **New Toolbar**.
The New Toolbar Item window opens.
3. Specify the required information and click **OK**.

A new toolbar item is added.

Customizing validation messages

In applications that require the user to input data, you can define validations on controls to ensure that correct data is provided. If appropriate data is not provided, validation fails, and an error message or notification message is displayed.

You can modify the textual content of these messages to customize them according to your application requirements. It is possible to customize these messages.

Customizing form level messages

At the XForm level, an alert message displays when you try to close the XForm without saving the changes.

To customize the message:

- Type the new textual content in the Form Messages pane of the Form - Properties window. For information about the Form - Properties window see [Setting Properties of an XForm](#).

You can only specify form-level messages here.

Customizing messages related to models

When you delete a record in an XForm, a message displays asking you to confirm the deletion of the record.

To customize the message:

- Type the new textual content in the Model Messages tab of the Model Properties dialog box. For more information, see [Model Properties dialog box](#).

You can only specify messages related to data models in the [Model Properties dialog box](#).

Customizing control level messages

It is possible to specify validation constraints to be applied to a control and the corresponding messages to be displayed for each constraint if its validation fails. You can use the `onvalidate` event to apply custom validations.

An error message or a notification message displays when the validation of a constraint fails.

To customize the message:

- Type the new textual content for the corresponding constraint in the Messages tab of the Formatting Options dialog box. For information about the Formatting Options dialog box, see [Formatting and Validating Controls](#).

You can specify these messages for all controls and for all data types.

Debugging processes at runtime

You can debug processes at runtime from the Process Instance Manager (PIM).

The `DebugProcess` method is used to debug processes at runtime. This method allows you to debug processes in different modes, set breakpoints, and run applications in real-time while debugging them simultaneously.

To debug processes at runtime:

1. In the My Applications App Palette, click  (Process Instance Manager). The Process Instance Manager window opens displaying a list of business processes that have process instances.
2. Click the link to process instances of a business process model in the **Total** column. The Instances by Process Definition window opens displaying a list of all process instances available for the selected business process model.
3. Right-click on a process instance whose **Status** is other than **Complete** and select **Debug**. Alternatively, select the check box against a required process instance whose Status is other than Complete and click  (Open Debugger). The selected process instance appears in a graphical view displaying the Process Debugger toolbar.
4. Select any one of the options from the Process Debugger toolbar to debug the business process instance.

External messages

An external message is a message or element that is part of an application service outside the context of the current business process.

You can drag external messages of any application service from the project content tree in Workspace Documents (Explorer) view on to a Start event, Intermediate Message event,

End event, Sub-process and, Send Message Activity constructs in a business process, without actually using the application service.

The following table displays the result of attaching external messages to BPMN constructs in a process model.

Action	Results
Start event	The input message displays the root name of the message that is defined in the WSDL of the application service. The input message changes to the newly created message in the Message Map . See Start Event Properties Interface for more information.
Receive Message event	The input message displays the root name of the message that is defined in the WSDL of the application service. The input message changes to the newly created message in the Message Map . See Receive Message Event Properties Interface for more information.
Independent Sub-process	Prompts the user to select either an input message or an output message. After the selection, the input or the output message displays the root name of the message that is defined in the WSDL of the application service. See Independent Subprocess Properties Interface for more information.
Send Message	Displays the root name of the message that is defined in the WSDL of the application service. This part is to be sent to the main process. See Send Message Event Properties Interface for more information.
End event	The output message displays the root name of the message that is defined in the WSDL of the application service. The output message changes to the newly created message in the Message Map . See End Event Properties Interface for more information.

Fields for configuring MDM service groups

This topic describes the fields required to create the MDM service groups.

Backend Details tab

Name	Type a name for the service group. Do not provide multibyte characters in the name of the service group.
Service Group	Type common name for the service group. Do not provide multibyte characters in the name of the service group.
Create Database Content	Select this option to create MDM database content.

Initial Settings tab

Hub Publisher	When selected, marks this as a Hub publisher. If this option is not selected the application connector is configured as a spoke publisher.
Network Configuration	<p>Machine name and Port number together are used to form a syncup ring between all the service containers within this service group for exchanging cache information, load balancing, and failover messages. To provide or modify the Machine name and Port number manually, you can select the Manual check box and do one or both of the following as per your requirement:</p> <ul style="list-style-type: none"> ■ Modify the Machine Name or IP Address value with the required machine name or IP address. ■ Provide an unused free port number in the Port Number field. <p>By default, the Manual option is not selected and the framework automatically assigns the machine name and an unused free port number. You must restart the service container to apply the change.</p>
Use different database for application	When this option is selected the application content is not placed in the same database as the MDM content. Hence, you must provide the database details for the application content in the Application tab.

Repository tab and Application tab

1. On the **Repository** tab, provide the database details where you want the MDM content to be placed.
2. Similarly, in the **Application** tab provide the database details where you want the application content to be placed.
3. Click  to save the changes.

Name	Name of the database configuration
Description	Description of the database configuration
JDBC	If you choose JDBC as the database driver, the associated fields are filled.
Advanced Options	<p>See Advanced Properties for more information.</p> <p>If the Hub or Spoke application database is PostgreSQL, then enable Support Special Characters in XML property. Support Special Characters in XML property needs to be enabled for the corresponding WS-AppServer service container also.</p>
Select Database Configuration	<p>To create a database configuration:</p> <ul style="list-style-type: none"> ■ Select the New Database Configuration option from Select Database Configuration.

File synchronization

All application content that is created in the workspace is stored on a repository.

These are further replicated and placed on the computer file system, which is treated as another location of content storage and development. The repository and file system are linked through synchronization. Application content may undergo changes on the repository (through workspace), as well as directly on the file system. Synchronization is the process of constantly keeping in sync both the changes made to the repository and the changes made to the file system.

Synchronization is a prerequisite for validate, publish, and package processes. Whenever you validate, publish, or package, the CWS framework first internally checks for the synchronized state of application contents and then proceeds with the required activity. This is to avoid any discrepancy between the contents in the repository and the contents on the file system. For example, sometimes developers prefer to use dedicated editors outside AppWorks Platform environment that provide special features to develop an application. In such cases, developers may not use workspace and its editors. As the content they develop needs to be packaged along with other contents in the repository, synchronization is mandatory. Therefore, before packaging, the content in repository and the content in the file system must be in sync with each other. The synchronizer operates bidirectionally - repository to file system and vice-versa.

Important: To meet High Availability requirements, you may clone the CWS processor on several nodes to enable application development in a shared/distributed environment. In such a case, ensure that the synchronization folder paths set on the AppWorks Platform installations on all nodes point to the same physical sync folder. As per the High Availability principles, the sync folder must not be located at one of the AppWorks Platform nodes itself.

Synchronization process and different states

As mentioned earlier in this topic, synchronizer works on two content stores - repository and file system. For each content store, it maintains two states - state of the previous synchronization (old-state) and the state of the current synchronization (new-state). The various combinations of the old state and new state make synchronization either a success or a failure. The probable combinations for successful synchronization and the action that synchronizer takes are mentioned in the following table.

Old State	New State	Action
-	exists	Places a new item
exists	exists	Based on the most recent modification, updates an item
exists	-	Removes an item

The comparison of the states also detects if items need to be merged. For example, an item is changed simultaneously in the repository and in the file system. There is a need to merge these two changes into one. When the merge activity fails, it results in synchronization conflicts.

To understand these states, consider the following real-time activities in an application development scenario.

Activity on the Repository (through Workspace)	Activity on the File System	Action by Synchronizer
A new document is created		Place the document on the file system
-	A new document is created	Place the document on the repository
An existing document is modified or renamed	-	Update or rename the document on the file system
-	An existing document is modified or renamed	Update or rename the document on the repository
A document is deleted	-	Delete the document from the file system
-	A document is deleted	Delete the document from the repository
An existing document is modified or renamed	An existing document is modified or renamed	Detects and displays synchronization conflicts

Completing search details

Before you begin:

- You must have the AppWorks Platform Certification Authority or the eNotary application installed for the Search Details pane to be visible.

Every Certificate Authority (CA) has a repository, in which the details (attributes) of the certificates issued by the CA are stored. This repository is used when searching for certificates, and hence the location of the repository must be specified in AppWorks Platform.

The attribute names of the certificates stored in the repository may be different in every CA. For example, in one CA, the common name of the subject, to whom the certificate is issued, may be stored as 'Name', while in another it may be stored as 'cn.' Hence, the name of the attributes must be mapped.

To complete search details:

1. If you do not want to provide the facility to search for certificates issued by a CA then select **No Search for this certificate** option, else do not select this option.
2. In the Search Details pane, select **Search Plugin**.
AppWorks Platform supports the LDAP search technology.

3. In the **Connection Parameters** pane, provide the following details.

Host	Type the location of the certificate repository. For example, srv-ind-esec0.
Port	Type the port number for accessing the LDAP service. For example, 389.
Version	Select the current version of the LDAP protocol.
Anonymous bind	Select this option to access the LDAP service anonymously. Ensure that the LDAP service allows anonymous access.
UserName	Type the user name for accessing the LDAP. This option is required only if you did not select Anonymous bind.
Password	Type the password corresponding to the user name. This option is required only if you did not select Anonymous bind. Both UserName and Password are required to access the LDAP.

4. In the Query Parameters pane, provide the **SearchBase** details.

Searchbase indicates the location within the LDAP, where AppWorks Platform must search for the certificates. For example, o=CAUserCertificates,DC=Cordys,DC=Com.

5. Specify the individual query parameters in the table.

This information is used to map the attribute names in the repository to the generic certificate attributes. The following table provides information related to query parameters available in the AppWorks Platform Certificate Authority Repository (CCAR).

Query Parameter	Value
State	st
organization	o
serialnumber	uid
email	mail
cn	cn
location	l
organizationalunit	ou

6. To use additional parameters, click  and specify the information in the new field that appears.
7. If you want to delete a parameter, select the check box beside the parameter and click .
8. Click .
- This saves the details specified for searching certificates.

Flow of control while loading welcome page

The AppWorks Platform application page starts with cusp.htm as its initial page. When the URL for the AppWorks Platform site is typed in the browser, the default page loads the IFRAME inside it, which in turn contains the cusp.htm page. The OpenText AppWorks Platform user welcome page shows all the tasks that the user has access to (based on the role).

When you open the welcome page, the system component is loaded. The welcome page thus functions as the root for all applications. The welcome page body registers a frame, called main, in which all the applications are loaded.

This is followed by loading the Application UI, which is the container for applications; that is all applications are loaded inside Application UI. The container itself is surrounded by a set of GUI representations that handle all the GUI-related activities of an application's container. The common GUI-activities include maximize, minimize, show, and hide. When the container initializes, the guiRepresentation object for containers also initializes.

Requests are then sent to LDAP to retrieve user details. User details include the role and organizations to which the user belongs. These user details are authenticated and a request is sent to the XDS repository to retrieve content for the user. The retrieved content comprises assigned tasks, shortcuts, and automatic applications that are loaded and displayed in the welcome page.

The applications loaded in the welcome page are Web pages that the AppWorks Platform Ajax Toolkit application framework can recognize and read. Once initialized, the default browser settings for the welcome page are then retrieved from saved user preferences in the XDS repository.

Free-form controls

Form controls that can be directly dragged on to the User Interface Editor are known as free-form controls. They are neither linked to a transactional nor a non-transactional model.

Free-form controls do not have any references. You can use User Interface with free-form controls in a business process model by publishing the User Interface. A user does not have to attach a dummy model to the UI controls and assign dummy references.

The data in the free-form control is fetched from the business process and when changes are made to the data, the modified data will be sent back to the business process.

In the Message Map free-form controls can be found within the element **forminputdata** and **formoutputdata** under **freeformcontrols**.

Caution: Do not create a model with the name FreeFormData as this will conflict with the control of the same name in the Message Map.

Locales

The following locales are available in AppWorks Platform. You can use the codes mentioned for each to set these programmatically.

Locale	Code
Afrikaans - South Africa	af-ZA
Albanian - Albania	sq-AL
Arabic - Algeria	ar-DZ
Arabic - Bahrain	ar-BH
Arabic - Egypt	ar-EG
Arabic - Iraq	ar-IQ
Arabic - Jordan	ar-JO
Arabic - Kuwait	ar-KW
Arabic - Lebanon	ar-LB
Arabic - Libya	ar-LY
Arabic - Morocco	ar-MA
Arabic - Oman	ar-OM
Arabic - Qatar	ar-QA
Arabic - Saudi Arabia	ar-SA
Arabic - Syria	ar-SY
Arabic - Tunisia	ar-TN
Arabic - United Arab Emirates	ar-AE
Arabic - Yemen	ar-YE
Armenian - Armenia	hy-AM
Azeri	az
Azeri (Cyrillic) - Azerbaijan	az-AZ-Cyril
Azeri (Latin) - Azerbaijan	az-AZ-Latin
Basque - Basque	eu-ES
Belarusian - Belarus	be-BY
Bulgarian - Bulgaria	bg-BG
Catalan - Catalan	ca-ES

Locale	Code
Chinese - Hong Kong S.A.R.	zh-HK
Chinese - Macau S.A.R.	zh-MO
Chinese - China	zh-CN
Chinese - Singapore	zh-SG
Chinese - Taiwan	zh-TW
Croatian - Croatia	hr-HR
Czech - Czech Republic	cs-CZ
Danish - Denmark	da-DK
Divehi - Maldives	div-MV
Dutch	nl
Dutch - Belgium	nl-BE
Dutch - The Netherlands	nl-NL
English	en
English - Australia	en-AU
English - Belize	en-BZ
English - Canada	en-CA
English - Caribbean	en-CB
English - Ireland	en-IE
English - Jamaica	en-JM
English - New Zealand	en-NZ
English - Philippines	en-PH
English - South Africa	en-ZA
English - Trinidad and Tobago	en-TT
English - United Kingdom	en-GB
English - United States	en-US
English - Zimbabwe	en-ZW
Estonian - Estonia	et-EE
Faroese - Faroe Islands	fo-FO
Farsi - Iran	fa-IR
Finnish - Finland	fi-FI

Locale	Code
French	fr
French - Belgium	fr-BE
French - Canada	fr-CA
French - France	fr-FR
French - Luxembourg	fr-LU
French - Monaco	fr-MC
French - Switzerland	fr-CH
Galician - Galician	gl-ES
Georgian - Georgia	ka-GE
German	de
German - Austria	de-AT
German - Germany	de-DE
German - Liechtenstein	de-LI
German - Luxembourg	de-LU
German - Switzerland	de-CH
Greek - Greece	el-GR
Gujarati - India	gu-IN
Hebrew - Israel	he-IL
Hindi - India	hi-IN
Hungarian - Hungary	hu-HU
Icelandic - Iceland	is-IS
Indonesian - Indonesia	id-ID
Italian - Italy	it-IT
Italian - Switzerland	it-CH
Japanese - Japan	ja-JP
Kannada - India	kn-IN
Kazakh - Kazakhstan	kk-KZ
Konkani - India	kok-IN
Korean (Korea)	ko-KR
Kyrgyz - Kyrgyzstan	ky-KZ

Locale	Code
Latvian - Latvia	lv-LV
Lithuanian - Lithuania	lt-LT
Macedonian - FYROM	mk-MK
Malay	ms
Malay - Brunei	ms-BN
Malay - Malaysia	ms-MY
Marathi - India	mr-IN
Mongolian - Mongolia	mn-MN
Norwegian (Bokmal) - Norway	nb-NO
Norwegian (Nynorsk) - Norway	nn-NO
Polish - Poland	pl-PL
Portuguese	pt
Portuguese - Brazil	pt-BR
Portuguese - Portugal	pt-PT
Punjabi -India	pa-IN
Romanian - Romania	ro-RO
Russian - Russia	ru-RU
Sanskrit - India	sa-IN
Serbian	sr
Serbian (Cyrillic) - Serbia	sr-SP-Cyril
Serbian (Latin) - Serbia	sr-SP-Latn
Slovak - Slovakia	sk-SK
Slovenian - Slovenia	sl-SI
Spanish	es
Spanish - Argentina	es-AR
Spanish - Bolivia	es-BO
Spanish - Chile	es-CL
Spanish - Colombia	es-CO
Spanish - Costa Rica	es-CR
Spanish - Dominican Republic	es-DO

Locale	Code
Spanish - Ecuador	es-EC
Spanish - El Salvador	es-SV
Spanish - Guatemala	es-GT
Spanish - Honduras	es-HN
Spanish - Mexico	es-MX
Spanish - Nicaragua es-MX	es-NI
Spanish - Panama	es-PA
Spanish - Paraguay	es-PY
Spanish - Peru	es-PE
Spanish - Puerto Rico	es-PR
Spanish - Spain	es-ES
Spanish - Uruguay	es-UY
Spanish - Venezuela	es-VE
Swahili - Kenya	sw-KE
Swedish	sv
Swedish (Finland)	sv-FI
Swedish (Sweden)	sv-SE
Syriac - Syria	syr-SY
Tamil - India	ta-IN
Tatar - Russia	tt-RU
Telugu - India	te-IN
Thai - Thailand	th-TH
Turkish - Turkey	tr-TR
Ukrainian - Ukraine	uk-UA
Urdu - Pakistan	ur-PK
Uzbek	uz
Uzbek (Cyrillic) - Uzbekistan	uz-UZ-Cyril
Uzbek (Latin) - Uzbekistan	uz-UZ-Latn
Vietnamese - Vietnam	vi-VN

Manually synchronize interface

The fields on the Manually Synchronize Interface are as follows:

Data Store	The specified name of the data store appears, by default.
Name	The specified name of the entity in the datastore appears, by default. There is 1 entry for each entity in the selected datastore.
Client Entity	The specified name of the spoke entity appears, by default.
Backend Entity	The specified name of the hub entity to which the spoke entity is connected appears, by default.
Bucket Count	The number of records present in the corresponding buckets table for the selected data store with respect to the Client Entity appears, by default.

Mapping variables

You must map the variables given under **Previous Messages** in the **Message Map** to the respective variables under **Input Messages** column after changing the path of the elements in the general file and regenerating the WSDL.

You can map any of the following:

- [Single variables](#)
- [Group variables](#)

Mapping single variables

After designing the business process model, you have to map single variables to the Target messages. For example, you have to map ContactTitle and Address to displayName and emailAddress respectively.

To map single variables:

1. From the **Message Map** tab, select the **Special offer for all customers** activity.
2. Select the **Pre Assignments** tab.
3. Map Contact Name under **Source** to displayName under **Target**, and map Address under **Source** to emailAddress under **Target**.

The variables in the Message Map are mapped to the Target.

Mapping group variables

After changing the path of the elements in the general file and regenerating the WSDL, you must map the element that contains the repeating element in the Source (in this example,

GetProductsResponse) to the new non-repeating element in the Target (in this example, Marketing).

To map group variables:

1. From the **Message Map** tab, select the **New Range of Products** activity.
2. Select the **Pre Assignments** tab.
3. Map the output of GetProductsResponse to the Marketing element in the **Target** in any of the following ways:
 - Drag the **GetProductsResponse** element from **Source** to the left box in the **Assignments** and drag the **Marketing** element from **Target** to the right box in **Assignments**.
 - Right-click on the **Marketing** element from **Target** and select **Create Assignment** from the context menu.
 - Select the **GetProductsResponse** element from **Source**, press CTRL and select the Marketing element from **Target**.
4. If you are creating an assignment for the Marketing element, copy the XPath /GetProductsOutput/GetProductsResponse into the left box in the **Assignments**.
5. Click on the left box in the **Assignments** to see the list of operations and select **Replace Content With, Children**.

The variables in the Message Map are mapped to the Marketing element in Target.

Matched instance details interface

There may be situations where you want to modify the values of an instance and then manually trigger an event.

You can edit the business object instances and manually trigger an event through the Match Instance details interface.

The interface lists the matched instances in a particular state. It displays the match criteria, state information, and the attributes of the instance under **Column**, **Matched**, and **Incoming** columns.

To use the Toolbar operation, Available Operations > ** <button that represent the event>

1. Click the button that represents the event that you want to trigger.
All the possible transitions, that you have defined while [designing a state model](#) appear under the Available Operations section.
2. The event is fired on the business object instance with the attribute values from the Proposed column.
If you have cleared the **Show proposed object** check box, the event is fired on the business object instance with the attribute values from the **Incoming** column.

3. The business object instance with the new values moves into a state that is a result of the event triggered.
A message appears that the event is triggered successfully.
The next events, if they exist, appear as buttons.

To use the Toolbar operation  (Refresh):

- Click to refresh the Match Instance details window.

To use the Toolbar operation, Show proposed object check box:

1. Select to edit or modify the attribute values of the business object instance. The attributes of the instance appear under Proposed column.
2. Edit attributes under the Proposed column and click one of the following arrows:
 - Click to reset the attribute value under Proposed with the attribute value under Matched.
 - Click to reset the attribute value under Proposed with the attribute value under Incoming.
3. Clear the check box if you do not want to edit or modify the attribute values of the business object instance.

Matched object instance interface

An instance can go into any state depending on the events that have occurred. The Match Object Instances displays the state history of a business object instance.

To use the Toolbar operation,  (Refresh page):

- Click to refresh the Match Object Instances window.

To use the Toolbar operation,  (View selected match object history):

1. Select the instance (row) for which you want to view the history of the business object instance click  (View selected match object history).
2. The State History window appears.
See [Matched Object Instance State History Interface](#) for more details.

To use the Toolbar operation,  (View selected match object):

1. Select the instance (row) for which you want to view the data, and click  (View selected match object).
2. The Match Instance details window appears with a list of records that have matched with the incoming business instance. See [Matched Instance Details Interface](#) for more details.

You can view the matched objects of the instance only if the you have enabled Data Quality in the State Engine for the model, while creating a data entity.

Matched object instance state history interface

An instance can go into any state depending on the events that have occurred. The State History interface displays the state history of a matched object instance.

To use the Toolbar operation, (Refresh):

- Click to refresh the State History window.

Modifying a task

Before you begin:

- [Create a task](#).

You can change the general and configuration details of a task such as name, description, application URL, input variables and so on in the Task editor window.

1. In the **My Recent Documents** or **Explorer** view of the Workspace Documents window, double-click the task. For information about changing the view, see [Working with Workspace Documents](#).
The <Taskname>-Task window opens.
2. In the <Taskname>-Task window, you can change the task details, as follows:
 - a. Click  (Quick Access Menu) > Properties.
The Properties - Document Properties dialog box opens.
 - b. In the **General** tab, change the necessary details and click **OK**.
3. In the <Taskname>-Task window, you can change the configuration details such as Application URL, Input Variables, Output Variables, caption and so on.
4. Click .
5. Publish the updated task to make it available for the current organization. You can publish in one of the following ways:
 - In the Quick Access Menu, select Publish option.
 - In the Explorer view of the Workspace Documents window, right-click the task and select Publish to Organization option.

The Task details are modified.

To delete a task, do one of the following:

- In the My Recent Documents view, point to the task and click .
- Select **Delete** on the context menu.
- In the **Explorer** view, go to the task, right-click it and select **Delete**.

Modifying the system attributes of a task

Users can modify the start date of a task, so that they can plan to take that task at a later point of time. For Case related Activities, if users have the necessary permissions, they can also change the due date of an activity.

Note:

- The Start Date of an In-Progress task or activity cannot be modified.
- Start Date should be earlier than the Due Date of the selected tasks.
- Due Date should be later than the Start Date of the selected tasks.
- Due date of a task can be modified only if:
 - users have change due date permission set on their roles, while [configuring the authorization for a case activity](#).
 - that particular activity allows Due Time Change in Inbox, while setting the [Duration property of the activity](#) in the business process mode.

To modify the system attributes of a task:

Users must have the change due date permission set on their roles while configuring the [authorizations for a case activity](#).

1. On the Welcome page, double-click  (My Inbox).
The My Inbox window opens, displaying the personal tasks, by default.
2. Click the required task from the work list and click  (Modify System Attributes) on the Task Toolbar.
The Modify System Attributes dialog box opens.
3. Beside Start Date, click  and select the required date.
The selected date is displayed in New Start Date.
4. Beside Due Date, click  and select the required date.
The selected date is displayed in Due Date.
5. Click **OK**.

The System Attributes of the task are modified.

To view Calendared Tasks:

When the start date of a task is modified, it is moved to the Calendared Tasks view.

- From the View list (in the top left corner of Personal Tasks view), select **Calendared Tasks**.
All the tasks that must be started at a future date are displayed in the grid below.

To view future tasks:

- Select **All Tasks** view on the top left corner of the Inbox.
All the tasks that you can work upon, except for the Completed or Skipped tasks, are

displayed in the grid.

Modifying an existing document

You may need to edit a document, modify its properties, or recreate it, depending upon the business requirement. You may also create more documents of the same type from the editor. AppWorks Platform facilitates modifying an existing document from Workspace Documents (Explorer) as well as Workspace Documents (My Recent Documents).

As you modify a model, the changes are being tracked by the system and a ** appears on the title bar of the document. If you do not want to retain the changes, you have an option to refresh the document and take the document to the last saved state.

To refresh the document:

1. Click on the toolbar of the editor.
2. Alternatively, if you are in Workspace Documents (Explorer) view, right-click the document that you modified and select  (Refresh). You will see the message - "Refreshing <document name> will discard all changes. Do you want to continue?"
3. To revert to the last saved state, click **Yes**.
4. To retain the in-progress changes, click **No**.

Important: In a team setup, it is possible that other developers work on the same documents on which you are working, or update or add other documents to the projects. In either case, CWS takes care of updating all the open workspaces containing those documents, as and when a change occurs in the CWS Repository. This activity is handled internally by event service that continuously checks for any modification in the workspace content. The findings are rendered on the UI in the form of confirmation messages that need a user action in response.

In the following table are details of the confirmation messages along with the situations in which they appear.

Document in CWS Repository	Document in Workspace Documents	User Action (in response to the prompt)
Modified, Renamed, Deleted, Moved	Opened but not modified	<ul style="list-style-type: none"> ■ Refresh - Receive the updates ■ Cancel - Ignore the updates and close editor. CWS will prompt for refresh the next time it is opened.
Modified, Renamed, Deleted, Moved	Modified by User	<ul style="list-style-type: none"> ■ Discard - Lose all recent changes and revert to the last saved state ■ Save As - Create a new copy with the modifications. Original

Document in CWS Repository	Document in Workspace Documents	User Action (in response to the prompt)
		<p>document will be obsolete.</p> <ul style="list-style-type: none"> ■ Cancel - Ignore the updates and close editor. CWS will prompt for refresh the next time it is opened.
Modified, Renamed, Deleted, Moved	Not opened	NA. Workspace automatically refreshes the document.
Added	NA	NA. Workspace automatically adds the document to the project.

To modify an existing document from Workspace Documents (Explorer):

1. In the Workspace Documents (Explorer), open <solution> > <project> and do one of the following:
 - Double-click a document.
 - Right-click a document and select **Open**.
The document opens in its editor in a separate window.

If you are working in Classic View, the document opens in the relevant editor to the right side of Workspace Documents

2. Do one of the following:
 - Edit or modify the document and click .
 - On the toolbar of the editor, open the  (Quick Access Menu) to perform several tasks.
 - On the toolbar of the editor, click  to create a new document of the same type.

To modify an existing document from Workspace Documents (My Recent Documents):

1. On Workspace Documents (My Recent Documents), click the document that you want to modify. The document opens in its editor in a separate window.
2. Do one of the following:
 - Edit or modify the document and click .
 - On the toolbar of the editor, open the  (Quick Access Menu) to perform several tasks.
 - On the toolbar of the editor, click  to create a new document of the same type.
The modified document is saved.

An existing document is modified or updated or a new document is created.

Mutex and override rules

Mutex rules

You can set a particular rule (source rule) to be mutually exclusive (mutex) to one or more rules. This way, if the source rule happens to trigger first, and the condition is satisfied, the mutex rule will not trigger. Likewise, if the rule that is set to be mutually exclusive happens to trigger first, the source rule will not trigger.

Example of Mutex rules

A customer has placed an order, and you have designed your application to send a request for the ordered items to two different warehouses. The process of procuring items from each warehouse differs. For this purpose, you have designed two separate rules to handle the operations. According to the application logic, if a particular warehouse responds to the request, you would not want the other rule to trigger. In such cases, you can use the mutex feature.

- Rules that are mutually exclusive to each other should be of the same type.
- When you set a rule to be mutually exclusive to another rule, the rule priority is ignored.
- You cannot set a constraint rule having internal actions, to be mutually exclusive with constraint rules having external actions, and vice versa.

Overriding rules

You can set a rule to override another rule. This way, even if a rule is acting on an object, you can choose to override that rule with another rule.

Example of overriding rules

You have designed your application to accept orders from customers and internal users. You want to design separate rules to handle each of these entities. But you want to give preference to the rule which is triggered when an order is placed by a customer. From this perspective, you have designed two rules: one rule would handle the orders received by the customer, and the other would handle the orders received by internal users. You can use the override feature so that in the event of both the entities (customer and internal user) placing an order at the same time, the rule that processes the order placed by the customer gets preference and will be implemented. The rule processing the orders received by the internal users will be overridden and the corresponding action will not be carried out, even though the rule is successfully executed. In case the rule placing the orders for the customer does not execute or the rule fails, the rule that places the orders for the internal customer will be executed.

You cannot override a constraint rule having internal actions with constraint rules having external actions, and vice versa.

Operators to build conditions

Operators

The operators are a combination of the arithmetic, Boolean and logical types. You can use these operators in framing conditions. The following sections describe the operators that are used in defining the build conditions.

Operator Type	Description
()	To define the first condition in a statement
(())	To define the second condition in a statement
((()))	To define the third condition in a statement
=	Equality operator
!=	Inequality operator
>	Greater than operator
>=	Greater than or Equal to operator
<	Less than operator
<=	Less than or Equal to operator
And	And operator to combine conditions in a statement
Or	Or operator to provide one or more conditions

Icons

Icons are the symbols used when inserting and deleting rows.

Icon Type	Description
+	To insert a new row
X	To delete a row

Process monitoring

Process monitoring tracks instances of a process through the lifespan of the instance. Monitoring involves updating and tracking instances of a particular process at Instance and Activity levels. Monitoring can be critical in some cases, but may also be a performance overhead at times.

Monitoring can be enabled or disabled during design time, at a particular process/activity level.

Levels of monitoring

The monitoring attribute can be set at three levels:

1. Instance Level
2. Process Level
3. Activity Level

At the Instance level

If the monitor property is . . .	Then . . .
Enabled at the instance level (when RequestProcess SOAP Request is set to 'true')	Activity level property is considered and the Process definition property is ignored. If the Disable Monitoring check box is selected, then Activity level information is not monitored. If the Disable Monitoring check box is cleared, then Activity level information is monitored.
Disabled at the instance level (when RequestProcess SOAP Request is set to 'false')	No monitoring is carried out at any level.

At the Process level

If the Disable Monitoring (at the process level) check box is . . .	Then . . .
Cleared	Activity level property is considered. If the Disable Monitoring check box is selected, then Activity level information is not monitored. If the Disable Monitoring check box is cleared, then Activity level information is monitored.
Selected	No monitoring is carried out.

Important: If neither the Instance level nor the Process level monitor attribute is found, then it will use the Enable Admin property from the CoBOC SOAP Processor configuration (which is the default behavior). If the Process Instance is running in debug mode, the Disable Monitoring (at the activity level) attribute will be ignored and the Activity data will be published.

At the Activity level

At the Activity level, for a Long Lived process, the Activities defined inside a Transaction can have the **Disable Monitoring** check box selected or cleared.

Crash Recovery can be enabled even if the **Disable Monitoring** check box is selected, but recovery data will not be persisted with unless monitoring is enabled.

For a Short Lived process, the **Disable Monitoring** check box will be disabled with value as 'On' and Crash Recovery option will not be shown to the user.

Process instance

A process model, when instantiated, is stored in the CoBOC as XML. The XML structure of a process instance is as follows.

```
<processinstance>
  <processmodel>
    [Process Model with status of each activity.]
  </processmodel>
  <messagemap>
    [All the properties and request & response messages.]
    [Implicit properties of process-instance]
    [Details of Fault-occurred, if any]
  </messagemap>
  <schemamap>
    [Readymade instances of all the schema elements defined in the WSDL for generating
    messages .]
  </schemamap>
</processinstance>
```

The following table describes the elements in the XML structure of a process instance.

processmodel	This element stores the executable process model. For each instance, this element contains the process model along with the status of each BPML activity. It is useful to know the status of an activity when resuming a process after it had transitioned to the waiting state. The process model is not the same as defined in BPML but a formatted version that is suitable for proper implementation by the Process Engine.
messagemap	This element stores instance specific data, request and response messages, and details of faults (if any).
schemamap	This element contains empty instances of XSD schema elements and complex types defined in WSDL. The empty instances are used to generate messages for action activities. While creating messages, these empty elements are filled by picking values from the properties in the Message Map. The schema map is generated when the process model is read and loaded from the repository.

Process instance manager interface

The following table describes the fields in the Process Instance Manager interface.

Folder	Displays the name of the folder in which business process model of a process instance is located.
Process Name	Displays the name of the business process model to which the process

	instances belong.
Published To	Displays the name of the space - organization or ISV to which a process instance is published.
Total	Displays the total number of process instances as a linked reference that are created for a business process model. Click the hyperlink to drill down and view all the process instances for a selected business process model.
Aborted	Displays the number of process instances that are aborted for a business process model as a linked reference. Click the hyperlink to drill down and view all the aborted process instances for a selected business process model.
Waiting	Displays the number of process instances that are in waiting for a business process model as a linked reference. Click the hyperlink to drill down and view all the waiting process instances for a selected business process model.
Complete	Displays the number of process instances that are completed for a business process model as a linked reference. Click the hyperlink to drill down and view all the completed process instances for a selected business process model.

Restart process instances interface

The Restart Process Instances dialog appears when you select a process instance for restarting it. This dialog box helps you to restart a process instance from a selected activity.

Process Name	Name of the business process to which the selected process instance belongs.
Version	Version number of the business process.
Description	Description of the business process to which the selected process instance belongs.
Restart from Activity	<p>Displays a list of all aborted activities that belong to the selected process instance. Select an aborted activity to indicate where from you want to restart the process instance and click Start.</p> <p>The progress bar appears and the Pause button is enabled.</p> <ul style="list-style-type: none"> ■ Click Pause, if required. When you click Pause, the Continue button is enabled and the Restart button appears. ■ Click Continue to continue with restarting or click Restart to restart the process instance again. ■ Click Cancel to cancel the current operation and exit from

	the Restart Process Instances dialog.
Original Aborted Instances	This field displays the number of process instances that were initially aborted.
Restarted Instances	This field displays the number of process instances that were restarted.
Not Restarted Instances	This field displays the number of process instances that are not yet restarted.

Process options menu

The following table describes the options available on the Process Options menu of the process modeling environment.

Suspend	This option suspends an active process instance. It is available for a process instance that is running, waiting, or ready for debug.
Resume	This option resumes an aborted or suspended process instance.
Run	This option reverts the business process from the Debug mode back to normal mode.
Run Interactively	This option is used to call the Run Interactively mode of operation in the Process Debugger. It is available for running process instances. If no instance is available, a new instance can be started.
Debug	This option calls the Debug mode of operation in the Process Debugger. It is available for process instances, which are running. If no instance is available, a new instance can be started.
Activity by Activity	This option debugs each activity in the business process. It executes the next activity in the sequence of the business process.
Step by Step	This option debugs each step (assignments and activities) in the business process.
Terminate	This option terminates a running process instance. It is available for all process instances that are not terminated or completed.

AppWorks Platform AJAX toolkit architecture

AJAX Toolkit contains a set of rich Web components built using JScript. These Web components are simple, lightweight components that encapsulate specific functionality (behavior) on a page. When applied to a standard HTML element on a page, a Web component enhances that element's default behavior. Web components enable extending the functionality of existing tags, extending HTML by creating custom tags, and encapsulation and componentization of code.

Within AppWorks Platform Ajax Toolkit architecture, the AJAX toolkit components (behaviors) communicate with the Web server using Web Gateway. The Gateway communicates with the Service Groups.

SOAP messages can be sent and received from the Gateway. The rendering of data is done using the various behaviors provided by the AJAX Toolkit. When the user requests for data from the back end, the behaviors (toolkit components) communicate the user requests to the gateway.

Communication between the browser and the gateway is performed by a behavior called the BusDataIsland. BusDataIsland exposes a set of properties, methods, and events to enable the developer to obtain data from the transaction object and manipulate the transaction object. It can also change the organizational context or receiver Service Group at run time.

The BusControl behavior presents the data, in a transaction object, in HTML format to the user. It also ensures that all the changes made to this HTML are correspondingly reflected in the transaction object in the BusDataIsland. The changes can be addition, deletion, or update of data. Thus, it maintains the correlation between the SOAP Request in the BusDataIsland and the HTML tree.

The AppWorks Platform environment thus enables generating Web applications using existing Web components.

Projects

Business models can be grouped into projects to meet different customer needs. Grouping is particularly relevant for:

- Reference modeling projects when you need to define a model for a type of business such as ETO (Engineer to Order), ATO (Assemble to Order).
- Business verticals such as Automotive, Electronics.
- Customer projects.
- Business Administrators when they need to export models based on projects.

Because projects are always stored in a particular version, they can be used for configuration management when a number of business models are grouped together for a specific customer, a specific vertical, and so on. Let us understand this with an example.

Consider a software consultancy that uses AppWorks Platform to design business models for a number of its customers who belong to the Automotive vertical. The consultancy can create business models in three different versions.

The following table describes each of the three versions.

Business Model Version	Description
Standard Version	This version can be used to store business models that are used by several customers belonging to any vertical industry such as

Business Model Version	Description
	Automotive or Electronics.
Automotive Version	This version can be used to store business models customized for the Automotive industry. This version can be derived from the Standard version.
Acme Version	This version is used to store business models customized for Acme, since Acme could have some very specific business processes that may not necessarily be the same as its competitors in the Automotive industry. This version is derived from the Automotive version.

Properties of an activity within a transaction

It is essential to understand the different properties that you can assign to an Activity, so that you can use them effectively to build transactions.

An Activity (or a sub-process wherein the Short Lived Process property is selected) will participate in the transaction if the Participate in Transaction property is selected. If there is an exception at any stage, all activities that are marked Participate in Transaction are rolled back. A method will not be a part of the transaction if this property is modified and set to false, although it forms a part of the transaction.

If a short-lived process consisting of WS-AppServer activities is called from a transaction of a long-lived process, the WS-AppServer activities in the short-lived process should be grouped as part of the transaction to maintain the transaction context. By default, for a WS-AppServer activity that is outside the transaction, the Participate in Transaction property is set to false and disabled.

Example:

Suppose there are four update activities defined in a transaction:

- Transaction Start
 - Activity A WS-AppServer method, Participate in Transaction =true
 - Activity B WS-AppServer method, Participate in Transaction =false
 - Activity C Web Service, Participate in Transaction =false
 - Activity D WS-AppServer method, Participate in Transaction =true
- Transaction End

If Activity D throws an exception, then Activity A is rolled back. However, Activities B and C are rolled back only if the service groups handling the WS-AppServer or Web Service methods share the same OS process as the Platform service container, that is, if they run inside the TomEE web server.

An End event within a transaction has an additional option Abort. Ensure that the End event is inside the transaction before you set this option. When an exception occurs, the transaction is automatically aborted. Hence, you do not need to model an End event of type Abort after the exception handling.

Properties of the attribute element

An attribute represents a field in database table (primary key) or a class.

To create a BusObject:

- Define values for the properties described in the following table.

unique	Determines whether the attribute is part of the unique identifier for the class. The acceptable values are: true, false
type	<p>Determines the type of the attribute. The acceptable values are as follows:</p> <ul style="list-style-type: none"> ■ relation - indicates relation to another model: <pre><attribute type="relation"> <name>ORDERIDObject</name> <type occ="1">ORDERHEADER</type> <match> <local type="foreignkey"> <attribute>ORDERID</attribute> </local> <remote type="primarykey"> <attribute>ORDERID</attribute> </remote> </match> </attribute></pre> ■ aggregation - Represents an existing class that has an independent existence. <pre><attribute type="aggregation"> <name>header</name> <layout>nested</layout> <type>Distribution::ORDERHEADER</type> </attribute></pre> ■ composition - Indicates that the attribute holds a class as a component within it. <pre><attribute type="composition"> <name>Father</name> <layout>nested</layout> <type def="class"> <class></pre>

<pre> <name>Father</name> <type>custom</type> <attributes> <attribute> <name>Name</name> <type>string</type> </attribute> <attribute type="composition"> <name>Son</name> <layout>nested</layout> <type def="class"> <class> <name>Son</name> <type>custom</type> <attributes> <attribute> <name>Name</name> <type>string</type> </attribute> </attributes> </class> </type> </attribute> </attributes> </class> </type> </attribute> </pre>	
<name>	Indicates the name of the attribute
<java><name>	<p>Indicates the name that is used to generate the accessor and mutator methods of an attribute. For example, in an attribute which has CustomerID as the <name> and CustID as the <java><name>, the following accessor and mutator methods are generated:</p> <ul style="list-style-type: none"> ■ public void setCustId(String value) ■ public String getCustId() ■ public String getCustId() <p>Note: The <java><name> values take precedence over <name> values, while generating these methods.</p>
<type>	<p>Represents the data type of the attribute. The acceptable values belong to the following categories:</p> <ul style="list-style-type: none"> ■ BasicType ■ ClassType
<size>	Based on the data type, this element indicates the size of the attribute. Use this element only if the data type requires it (for example, string).
<maxLength>	Based on the type, it indicates the maximum length of the attribute.
<length>	Based on the type, it indicates the exact length of the attribute.

<minInclusive>	Indicates the lower bound of the value, with the value included.
<maxExclusive>	Indicates the lower bound of the value, with the value included.
<derived>	Indicates the class from which this attribute is derived.
<scale>	Indicates the number of digits to the right of the decimal point.
<precision>	Indicates the total number of digits used.
<pattern>	Indicates the pattern that the attribute must match, based on the type of attribute.
<initial>	Indicates the initial value of the attribute. The initial value will be applicable only for validate requests.
<changeability>	Indicates the changeability of the attribute. The acceptable values are as follows: <ul style="list-style-type: none"> ■ changeable - Value can be modified at any point of time ■ frozen - Value assigned during creation of the object but cannot be modified subsequently ■ add only - Value can be assigned anytime (during insert or update operation) but only once, and cannot be modified subsequently
<persistence>	Determines the mode of persistence. The acceptable values are: <ul style="list-style-type: none"> ■ Blank (Persistent) - Default mode of mapping to a database column ■ Transient - Attribute does not map to a database column and is not persisted
<required>	Indicates the necessity of providing a value for the attribute. The acceptable values are: <ul style="list-style-type: none"> ■ True - the attribute must have a value ■ False - the attribute does not require a value. <p>This property allows you to determine whether to allow null values for a particular attribute or not . Based on the specification, WS-AppServer generates Java code to create the necessary constraint that will be used during validation. If nothing is specified, False is the default value.</p>

Ranges

For ranges, you can track the KPI and specify ranges either as a percentage or as an integer or duration (second or minute or hour or day or week or month or year). For each range, you specify the lower limit and the upper limit.

If you define your KPI as Average time to approve an order for the last 30 days and your target as 2 days, set several ranges as follows.

Name	Lower Limit	Upper Limit
Excellent	0	1
Good	1	2
Fair	2	3
Poor	3	5

You can have lower limit of the first range and upper limit of the last range as unbounded (you can leave the lower limit of the first range and the upper limit of the last range empty - the boundaries for these are filled at runtime accordingly). For each set of ranges, you can specify a name.

References

The references for business modeling are:

- [Business Process Modeler Interface](#)
- [Crash Recovery](#)
- [Reliable Messaging](#)

Crash recovery

Crash recovery is the ability to restore a process instance from the point of execution failure without having to restart. If a business process encounters an issue due to a system or processor failure or due to the stoppage of a service container, you can restore the system using the crash recovery feature at runtime.

Need for crash recovery

To understand the need for crash recovery, consider the following example.

A business process is scheduled to run 400 iterations, but the Business Process Management Service Container crashes after 200 iterations. In such a case, if the Business Process Management Service Container is restarted, the entire process has to begin again (i.e., from the first iteration).

However, if crash recovery is enabled, the business process restarts from the point at which the process was terminated (in this case, from iteration 201).

When you enable crash recovery at the business process model level, the crash recovery feature is automatically applied to all the activities with that business process even if it is not enabled at the activity level.

When the crash recovery feature is enabled only at the activity level, crash recovery is applicable only for that particular activity. However, when crash recovery is enabled at both the business process model and activity levels, then the property set at the model level takes precedence.

Reliable messaging

Reliable messaging uses a queue to ensure that Web service message calls are not lost due to unavailability of the back-end. For example, if the back-end is down the Web service calls will stay in the queue until the back-end is up. As a result, the Web service call can take a long time if the Web service call is not handled directly. Therefore, Web services called via reliable messaging are asynchronous by nature. So, in a business process the output message can neither be used nor can exceptions be handled for Web services called via reliable messaging.

Hence, reliable messaging is recommended for Web services for which it is essential, that they are always executed and executed only once; and for those Web services for which the functionality cannot go wrong. For example, update Web services calls that can never have primary key violations and do not contain invalid data.

Regions

Regions are placeholders that host sub applications known as App Palettes in an application. An application can be divided into multiple regions, and a region can hold multiple App Palettes. Multiple App Palettes in a region are displayed as tabs. You can select a tab to display the respective App Palette.

The region definition can be set in two ways:

- Inline in the HTML structure of an application.
- As an XML in the Application Definition or in the application itself. In this scenario, the regions defined in the Application Definition take precedence over the region XML definition specified in the application itself.

The following methods are available for regions.

Method	Description
getOffset()	Gets the offset specified for the region.
getType()	Returns a string that specifies the type of region.
setOffset(value)	Sets the offset for the region. Values can be in percentage (%) or in pixels (px). If no unit is specified, it is taken as pixels by default.
setType(sType)	Sets the orientation of a region dynamically, when the region is loaded at runtime. Possible values are top, bottom, left, or right.

Region sets with multiple regions

In case more than two regions need to be defined for an application, they must be wrapped inside a regionset. The concept of region and regionset is similar to the frame and frameset concept in HTML. Different attributes need to be specified depending on if it is a region or a regionset.

Defining regions inline in the HTML structure of an application

To define regions, set class="region" or class="regionset" for the DIV objects defined in the HTML document body.

When two regionset or regions are defined, the attributes of only the first declared region or regionset are considered for dimensions and type.

The following attributes are available for defining region-sets and regions for an Application.

Attribute	Description
appPaletteDefinitions	Denotes the XML Dataisland IDs of application definitions of the App Palettes to be loaded in the region by default. The IDs must be separated by spaces if more than one App Palette is to be loaded.
class	Required. Optional values are regionset and region.
id	Denotes the unique identifier given to an App Palette, which is used to refer to it after loading an App Palette. If not specified, an auto-generated id is assigned.
regionName	Denotes the name of the region. It will be displayed in the layout structure of the regions while dragging the App Palettes from one region to another region. If nothing is provided for the regionName, an empty layout structure of the regions will be displayed in the runtime.
multipleAppPalettes	Denotes whether multiple App Palettes can be loaded in a region. Available options are true and false(default).
offset	Mandatory; defined as offset = "<value>%". Denotes the percentage value (%) of offset required for the region.
orientation	Denotes the position of tab headers in case of multiple App Palettes. Possible values are: <ul style="list-style-type: none"> ■ left: Tab headers are positioned along the left edge of the tabbed multiple App Palettes. ■ right: Tab headers are positioned along the right edge of the tabbed multiple App Palettes. ■ top: Tab headers are placed at the top of the tabbed multiple App Palettes. ■ bottom: Default. Tab headers are placed at the bottom of the tabbed multiple App Palettes.
resizable	Denotes whether the region is resizable or fixed, with reference to the offset. It holds the value yes (default) or no.

Attribute	Description
textDirection	Denotes the orientation of text displayed in tab headers (in case of multiple App Palettes). Possible values are: <ul style="list-style-type: none"> ■ vertical: This is the default text direction for left- and right-oriented App Palettes. ■ horizontal: This is the default text direction for top- and bottom- oriented App Palettes.
type	Required. Optional values are left, right, top, and bottom.
visibility	Available options are hidden(default), display, maximized, and minimized. The default value is hidden; if no content is specified in the region, the region is hidden from view.

HTML App Palettes in regions

It is possible to load more than one App Palettes in a region. The App Palettes can be loaded using the API application.loadAppPalette(appPaletteDefinition).

While defining inline HTML App Palettes inside a region, the static HTML must be wrapped in a DIV container. If multipleAppPalettes="true" for the region, the attributeclass="apppalette" has to be set to the same.

Other attributes that can be set to an App Palette are:

- caption = <Caption>
- title = "true/false". The default value is true.

While setting the HTML content in a region, use the getTab() method of the region object to get the GUI representation of the App Palette. The content can be set on the App Palette guirepresentation using the setContent() method.

You can define inline regions with App Palette as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"> <html> <head> <title>Test AppPalette</title>
<script src="/cordys/wcp/application.js"></script> <script type="cordys/xml"
id="testCollapsible"> <Application focus="true" display="visible" title="false"
inPreferences="true"> <id>testCollapsible</id>
<url>/cordys/wcp/test/library/ui/testcollapsible.htm</url>
<description>Collapsible</description> <caption>Collapsible</caption>
<icon>/cordys/wcp/theme/default/icon/task/classicapplication.png</icon> <region
docked="true" width="200px" height="400px">rightRegion</region> </Application>
</script> <script type="cordys/xml" id="testCalendar"> <Application focus="true"
display="visible" title="false" inPreferences="true"> <id>testCalendar</id>
<url>/cordys/wcp/test/library/ui/testcalendar.htm</url>
<description>Calendar</description> <caption>Calendar</caption>
<icon>/cordys/wcp/theme/default/icon/task/classicapplication.png</icon> <region
docked="true" width="200px" height="400px">rightRegion</region> </Application>
```

```
</script> </head> <body scroll="no" leftmargin="0" topmargin="0" rightmargin="0" bottommargin="0"> <div style="width:100%;height:100%;overflow:auto"> <div class="regionset" style="width:100%;height:100%;"> <div class="region" id="leftRegion" regionName="Left Region" type="right" visibility="display" offset="70%" resizable="yes"> <div> <!-- No multiple AppPalettes. So class="apppalette" is not necessary --> <h1>Left Region</h1> </div> </div> <div class="region" id="rightRegion" regionName="Right Region" multipleAppPalettes="true" appPaletteDefinitions="testCollapsible testCalendar"> <!-- Multiple AppPalettes. So class="apppalette" is necessary --> <div class="apppalette" caption="Static HTML" title="false"> <div class="mainbody"> <h1>Right Region</h1> </div> </div> </div> </div> </body> </html>
```

Defining regions in the application definition

In the Application Definition, you can define a region as follows:

```
<script type="cordys/xml" id="testAppPaletteUIApplication"> <Application focus="true" display="visible"> <id>testAppPalette</id> <url>/cordys/wcp/test/library/ui/testappaletteui.htm</url> <description>App Palette UI</description> <caption>App Palette</caption> <icon>/cordys/wcp/theme/default/icon/task/licensemanager.png</icon> <frame docked="false">main</frame> <info>This is a sample text that can be shown as Info</info> <regionsetId="parentSet"> <region id="leftRegion" type="left" visibility="display" offset="30%" resizable="yes" /> <region id="rightRegion"/> </regionset> </Application> </script>
```

The above definition divides an application as below.

If an application contains elements in the body and a regionset is defined in the application definition, the framework will identify the area where the body HTML is to be displayed. For example, if the region type is 'left', the body HTML is displayed on the right side and if the region type is 'right', the body HTML is displayed on the left side. Similarly, if the region type is 'top', the body HTML is displayed at the bottom and so on.

The following attributes are available for defining regions in the application definition.

Attribute	Description
id	Required. Denotes the unique identifier given to a region, which is used to refer to the region while loading an App Palette.
type	Required. Specifies whether the region should be placed to the left, right, top, or bottom.
offset	Mandatory; defined as offset = "<value>%". Denotes the percentage value (%) of offset required for the region.
visibility	Available options are hidden(default), display, maximized, and minimized. The default value is hidden; if no content is specified in the region, the region is hidden from view.

Attribute	Description
resizable	Denotes whether the region is resizable or fixed, with reference to the offset. It holds the value yes(default) or no.
appPaletteDefinitions	Denotes the XML Data island IDs of application definitions of the App Palettes to be loaded in the region by default. The IDs must be separated by spaces if more than one App Palette is to be loaded.

Request and response messages

The Message Map stores instance specific data, messages, and properties. Whenever a new message is received by the process instance, it is added to the Message Map, forming a repository of all request and response messages generated.

The following table lists the different types of request and response messages, and describes how they are stored in the Message Map.

If the request/response message is a...	Then...
SOAP message to invoke an external SOAP enabled application	All parameters of the request and response messages are stored in the Message Map. These parameters are encapsulated in the respective message names as specified in their WSDL definitions.
Process-to-process or inbox communication	The message is encapsulated using the message names specified in the WSDL definition

Resizing a swimlane

Before you begin:

- You must have the Business Analyst role in AppWorks Platform application package to resize a swimlane.

A user can resize the height or width of a swimlane to remove excess space or add space as required.

To resize a swimlane:

1. Click <**Workspace Documents (Explorer)**> > <Solution> > <Project> > <business process model>.
- The business process model appears in the AppWorks Platform business process modeling environment.
2. Select the swimlane you want to resize, and click the right boundary of the swimlane (for a vertical swimlane) and the bottom boundary of the swimlane (for a horizontal swimlane) and drag in or out to adjust its width or height respectively.

Retrieving the template with installation inputs

Before you begin:

- You must have the role of a developer to retrieve the template.
- You must be in the context of System Organization.

While installing applications, you have an option to record the complete web-based Application installing inputs to a template. This is created and saved in the AppWorks Platform Repository with a key. Below is the procedure to retrieve the template.

To retrieve the template:

1. On the Welcome page > My Applications, click  (Service Test Tool). The Service Test Tool window opens.
2. Select the monitorsoapnode@<machinename> from **Service Groups**.
3. Select the MethodSet ISVPackage from **Web Service Interface**.
4. Select GetISVPTemplate from **Web Service Operation** and click **Compose the Request**.
The request SOAP message for the selected Web service operation is composed and is displayed in the Test Request section.
5. Replace the default text KEY parameter in the **Task Request** section with **Cordys/WCP/Installation Templates/default/<File name>**.
6. Click **Test Operation**.
The Result Messages section is updated with the request and response messages.
7. Click the response to view the associated XML code.
8. Click the **Copy to Clipboard** icon to copy the code block.
9. Copy the code snippet within <ISV Command> tags and save the content in a .txt file.
This template is used as an input for the Silent installation of Applications.

Starting a batch of service containers

An Application Developer plays a vital role in representing the dependencies. A dependency list is determined during the Application creation. This list refers to a set of namespaces on which the application is dependent upon at the start up (namespaces used to invoke any Web service in the open () Web service operation of the Application connector).

- A dependency list is associated with the application connector, hence must be a part of the Application Connector configuration.
- The list must not contain namespaces of LDAP and Monitor.
- In case of no dependency list, Monitor treats it to be no dependency to start the Service Container, which implies that such a Service Container would be in the initial batches.

- Add the startup dependency list as part of the application connector definition similar to 'classpath' addition. The application definition is bundled as a file at installation under <CORDYS_INSTALL_DIR>\XMLStore\collection\Cordys\WCP \Application Connector. As part of the Service creation during installation, include the startup dependency.

Sample configuration

A sample Application connector configuration with startup dependency is listed below. The XML element 'startupDependency' contains all the 'namespaces' that it is dependent on.

```
<configuration
implementation="com.eibus.applicationconnector.myApplicationConnector">
.....
<startupDependency>
<namespace>http://schemas.cordys.com/namespace1</namespace>
<namespace>http://schemas.cordys.com/namespace2</namespace>
</startupDependency>
</configuration>
```

Run-time documents

The documents created and modified during design time can be seen and worked with using Workspace Documents. However, when the documents are validated and published, each document is stored at a different location for use at run time. This topic lists the documents and their associated run-time locations after they are published.

Document type	Runtime location after publish activity
Action Template ()	CoBOC Browser () > <system>/<ISV>
Application Connector ()	XMLStore Explorer () > Collection > Cordys > WCP > Application Connector
Condition Template ()	CoBOC Browser () > <system>/<ISV>
Content Map ()	CoBOC Browser () > <system>/<ISV>
Data Transformation models ()	CoBOC Browser () > <system>/<ISV>
Decision Table ()	Decision Tables can be seen in the database to which the Rule Service is configured. It is stored in <Database> > BizRule table - You need to have access to the database. Rule Test Tool > <Object Template> - You need to know the name of the Object Template on which the Decision Table is built.
Email Model ()	XML Store Explorer () > Collection > Cordys > notification > emailmodels > models > Human Task ID > <Email Models> Email models can be seen in

Document type	Runtime location after publish activity	
	the database to which the Notification Service is configured. It is stored in <Database> > email_template table - You need to have access to the database.	
External User Interface (	User Manager > Roles-Tasks or Users-Tasks view	
Inbox model (	Inbox Models can be seen in the database to which the Notification Service is configured. It is stored in <Database> > message_model table - You need to have access to the database.	
Java Archive Definition (	<AppWorks Platform_installdir>\<instance name>\bsf\runtime\deploy\<organization>.jar - for JAR files generated by WS-AppServer along with other WS-AppServer content <AppWorks Platform_installdir>\<instance name>\cws\build\<organization>\<workspace name>\cws\<project>.jar - for JAR files that were not generated by WS-AppServer	
MDM Data Steward Cockpit (	The business objects in their respective states are displayed.	
MDM Model Browser (	Displays all the published MDM models. Click  > 	CoBOC Browser () > <system>/<ISV>
Role (	LDAP Explorer > Cordys > <Organization> > Organization Roles > Role or User Manager > Users-Roles or Roles- Roles or Roles-Users view	
Rule (	Rules can be seen in the database to which the Rule Service is configured. These are stored in <Database> > BizRule table - You need to have access to the database. Rule Test Tool > <Object Template> - You need to know the name of the Object Template on which the Rule is created.	
Rule Group (	Rule Groups can be seen in the database to which the Rule Service is configured. It is stored in <Database> > Rule Group table - You need to have access to the database.	
Schedule (	Schedule Manager > Active Schedules/ Inactive Schedules	

Document type	Runtime location after publish activity
User Interface (XForms) (	XML Store Explorer () > Collection > Cordys > WCP > XForms > runtime
Web Library Definition () Note: The Web Library Definition does not get published to the run time. It is the Web file that it contains gets published to run time.	<AppWorks Platform_installdir>\<instance name>\Web* *Note: The Web file will appear depending upon the Qualified Name you set on the project or the folder containing it.
Web service operation () Web service interface ()	System Resource Manager > Web Service Explorer ()
WS-AppServer Package ()	XML Store Explorer () > Collection > Cordys > WS-AppServer > runtime > classregistry \ .cmx file.

Runtime Translation Repository for XForms

During design time, you can specify the appropriate translated labels for translating an XForm. The selected labels and their translations in each language are stored in XForm's Multi-Language Mapper (.mlm) files for each language, which is used to create translated XForms at run time.

This means that the XForms are not translated at design time, but the.mlm file in respective language applies only when you view the XForm at run time. Thus, you can create a single XForm, define its translation in various languages, and at run time, choose the language in which to view it. This also allows the re-use of labels and their translations across various applications.

If there is no translation available for a specific label in the MLM, the label will be displayed in the Project Language.

Sample HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html onapplicationselect="initialize()">
<head>
<script type="text/javascript" src="/cordys/wcp/application.js"></script>
<style>
input{font-size:11; font-family:verdana;}
label{font-size:11; font-family:verdana;}
```

```

</style>
</head>
<script type="text/javascript"
src="/cordys/wcp/admin/behavior/applicationconnector.js"></script>
<script type="text/javascript">
// Creates the xml
function createConnectorConfiguration(configurationNode)
{
    //Create & add XML nodes to configurationNode
    return true;
}

// This function fills in the values when used as a property page
function fillInPropertyScreen(configurationNode)
{
    //Parse the data from configurationNode XML and fill in the property screen
}
</script>
<body leftmargin="20">
<table height="70%" width="100%" cellSpacing="0" cellPadding="0" align="middle">
<tr height="30%">
<td>
<label>Sample Application Connector</label>
<br />
</td>
</tr>
</table>
</body>
</html>

```

Sample Java code snippets

This topic provides sample Java code snippets for ApplicationConnector and ApplicationTransaction classes.

ApplicationConnector Class

```

package com.cordys.countconnector;

import com.eibus.soap.ApplicationConnector;
import com.eibus.soap.ApplicationTransaction;
import com.eibus.soap.Processor;
import com.eibus.soap.SOAPTransaction;

public class CountConnector extends ApplicationConnector {
    // This method gets executed when a service container is started
    public void open(Processor processor) {

```

```

// Check processor configuration, validate it and throw back exceptions
// if any
}

// This method gets executed when a service container is reset
public void reset(Processor processor) {
// Any reset operations like cache reset, variables reset etc
}

// This method gets executed when a service container is requested to
// process a soap request
public ApplicationTransaction createTransaction(
SOAPTransaction soapTransaction) {
// Returns the transaction class that processes the soap requests sent
// to this service container
return new CountTransaction();
}
}

```

ApplicationTransaction Class

```

package com.cordys.countconnector;

import com.eibus.soap.ApplicationTransaction;
import com.eibus.soap.BodyBlock;

import com.eibus.xml.nom.Node;

public class CountTransaction implements ApplicationTransaction {
    static int count = 1;

    public CountTransaction() {
        // Count transaction constructor
    }

    // This function informs whether a request can be processed or not. The
    // implementation type of the webservcie operation is checked against the
    // list of types that can be processed by this connector.
    public boolean canProcess(String type) {
        return "Count".equals(type);
    }

    // Actual processing happens here
    public boolean process(BodyBlock request, BodyBlock response) {
        int responseNode = response.getXMLNode();
        // Creating status tag in response which is returned as the response to the user
        Node.getDocument(responseNode)
            .createTextElement("status", "This is Method No " + count++,
        responseNode);
    }
}

```

```

        return true;
    }

    //Code that Commits the transaction must come here
    public void commit() {
    }

    //Code that aborts the transaction must come here
    public void abort() {
    }
}

```

Scheduler Service Interface

The following tables describe the fields on the **Scheduler** tab of the **Scheduling** service container.

Select Database Configuration	To create a Database Configuration, select the New Database Configuration option from the list and provide the required information in the dialog box that appears. To continue with an existing Database Configuration - Select the required database configuration from the list. The associated fields are automatically filled.
Advanced Options	<p>Expand the group box and provide the necessary details.</p> <p>Caution: Setting the Cursor Cache Size to a high value would cause high memory consumption. Based on the application usage, the administrator must set an optimal value. This also applies to the Query Cache size.</p> <p>For example, the optimal cursor cache size can be between 50 - 100, subjective to the application usage.</p>

Scheduler Engine tab

When the schedule service container is stopped due to network failure or stopped manually, the schedule instances during the downtime are termed as Missed Schedules.

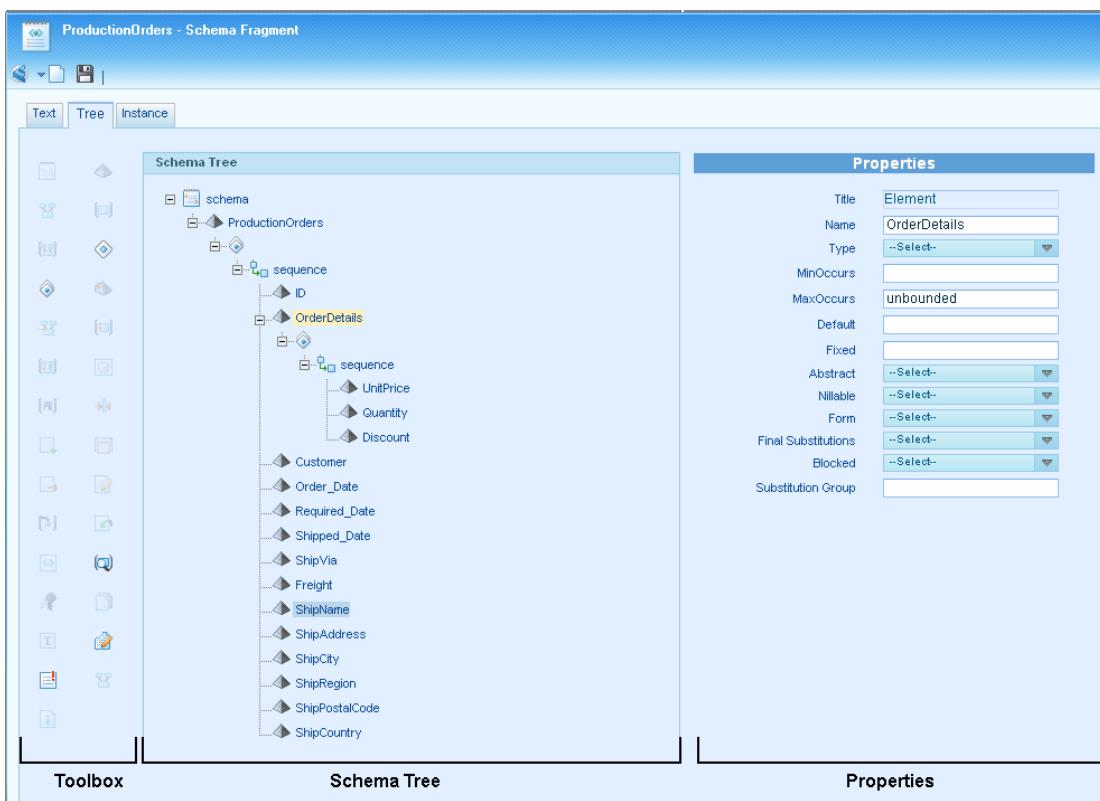
For example: Let us assume that a task is scheduled to be triggered hourly at 15th, 30th, 45th, and 59th minute, that is 4 times in an hour. If the **Scheduling** service container is stopped on the 10th minute of the hour and is restarted on the 40th minute, the 15th and 30th minute schedules are not triggered. So, in such situations, an Administrator can configure to trigger none/one/all of the missed schedules by selecting one of the following options in the **Scheduler Engine** tab.

Trigger none of the schedule instances on start-up which are missed during downtime	Select this option, if you do not want any missed schedules to be triggered upon restarting the Scheduling service container.
---	---

Trigger one schedule instance per schedule on start-up which are missed during downtime	Select this option, if you want only one instance of the missed schedules to be triggered upon restarting the Scheduling service container. As per the example, the schedule will be triggered once at the 40th minute and the remaining schedules will follow the plan.
Trigger all the schedule instances per schedule on start-up which are missed during downtime	Select this option, if you want all the instances of the missed schedules to be triggered upon restarting the Scheduling service container. As per the example, the 15th and 30th minute schedules will be triggered at the 40th minute (that is, two times) and the remaining schedules follow the plan.

Schema editor

This topic describes the various sections and tools in the Schema Editor frame.



The sections in the Schema Editor Frame are as follows.

Section	Description
Toolbox	The Schema Editor toolbox. The toolbox hosts a variety of tools you can use to build complex schemas. Refer to the table below for more information.
Schema Tree	This is the section where the graphical view of the schema is displayed. To start building the schema fragment: <ul style="list-style-type: none"> ■ Right-click the schema element, select Add and select the required child element.
Properties	This area enables you to define properties of each element that you include in the schema.

The following table describes the tools in the Schema Editor toolbox:

Tool	Description
 (Schema)	Defines the root element of a schema.
 (Element)	Defines an element.
 (Attribute)	Defines an attribute.
 Group (Group)	Defines a group of elements to be used in complex type definitions.
 (Attribute Group)	Defines an attribute group to be used in complex type definitions.
 (Simple Type)	Defines a simple type and specifies the constraints and information about the values of attributes or text-only elements.
 (Complex Type)	Defines a complex type element.
 (Reference to Element)	Defines reference to an Element.
 (Reference to Attribute)	Defines reference to an Attribute.
 (Reference to Group)	Defines reference to a Group.
 (Reference to Attribute Group)	Defines reference to an Attribute Group.
 (Content)	Defines Content type for Elements. Elements can have different Content types.
 (Model Group)	Defines Group, which can be nested inside an element. After you create a group node or a complexType node, you can add a model group node as a child.

Tool	Description
 (Restriction)	Defines restrictions on a simpleType, simpleContent, or a complexContent.
 (Aggregator)	Defines an Aggregator.
 (Facet)	Define restrictions on XML Elements.
 (Extension)	Extends an existing simpleType or complexType element.
 (Notation)	Describes the format of non-XML data within an XML document.
 (Include)	Adds multiple schemas with the same target namespace to a document (include).
 (Import)	Adds multiple schemas with different target namespace to a document (import).
 (Redefine)	Redefines simple and complex types, groups, and attribute groups from an external schema.
 (Identity Constraint)	Defines a set of elements as Identity Constraints.
 (Selector Key)	Specifies an XPath expression that selects a set of elements for an identity constraint.
 (Text)	Defines the text for documentation.
 (Documentation)	Defines text comments in a schema (must go inside annotation).
 (Annotation)	Specifies the top-level element for schema comments.
 (Comment)	Defines the text comments.
 (any Attribute)	Extends the XML document with attributes not specified by the schema.
 (AppInfo)	Specifies information to be used by the application (must go inside annotation).

XML schemas comply with the standards set by the W3C.

Select X-Axis field

Field Name	Displays the output fields of the Business Measure.
Field Type	Select one of the following options: <ul style="list-style-type: none"> ■ Primary: The data of the field selected as Primary is rendered as x-

	<p>axis labels in the chart.</p> <ul style="list-style-type: none"> ■ Secondary: The data selected for this field type appears as a Legend for the x-axis. The Data is grouped by using both Primary and Secondary data. <p>The Field Type is displayed for only Line Chart, Bar Chart, Stacked Bar Chart, Combinational Chart, and Combinational Dual-Y Chart View types because the other View types can support only one X-axis field.</p>
XPath	Displays the Xpath of the field.

Select Y-Axis field

Field Name	Displays the output fields of the Business Measure.
Y-Axis	<p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ Primary: The data of selected field is plotted along the as the Primary y-axis in the chart. ■ Secondary: The data of the selected field is plotted as the Secondary y-axis in the chart. ■ The Y-Axis field appears only for the Combinational Dual-Y Chart view type. ■ Select at least one Primary before you select Secondary.
Render As	<p>Select one of the following options:</p> <ul style="list-style-type: none"> ■ Line: The data of the field selected is rendered as Line in the chart. ■ Bar: The data of the field selected appears as Bar view in the chart. ■ Area: The data of the field selected is rendered as Area in the chart. <p>The Area field appears only for the Combinational Chart and Combinational Dual-Y Chart view types.</p>
XPath	Displays the Xpath of the field.

Sending email

To send an email:

1. On the Define Actions page, click the **Send Email** tab.
The Send Email page opens in entry mode, with all the mandatory fields.
2. In the **To** column, type a mail Id, to whom the action is intended.
3. In the **Subject** column, type the subject of the Process Event.
4. Type the message that is intended to be sent.
5. Click **OK** to complete the process.

The information entered is saved.

6. Click **Next** to define the next action, if required.

By default, the Invoke Business Process action page appears.

Setting additional properties of a task

This task is performed in the process of setting additional properties such as:

- Custom Variables - the additional information, like metadata, that you want to add to the task.
- Task Identifiers - the properties or attributes that are used as conditions when searching for tasks in the AppWorks Platform Inbox.

To set properties of a task:

1. Click  next to Custom Variables tab.
The <Custom Variables> schema editor appears.
2. Provide the custom variables in XML format in the <schema editor> and click **<OK>**.
The XML is displayed on the Custom Variables field.
3. To select task identifiers, click in  the Task Identifiers section.
The Task Identifiers dialog box opens, displaying all the available task identifiers.
4. Select the attributes on which you want to query the tasks in the Inbox by selecting the check box for each and click **OK**.
The XPath of the task identifier and its display name are displayed in the Task Identifier table.
5. Click  on the tool bar.

The additional properties of the task are set.

Setting special attribute properties

Special attributes provide the ability to query the business object with the specified attribute value. This task is performed while creating an object template.

To set special attribute properties:

1. On the Object Template page, click the **Special Attributes** tab.
2. Click the required attribute and drag it to the **Selected Special Attributes** pane to set its properties.
The selected attributes along with their XPaths are displayed in the Selected Special Attributes pane. The attribute name is displayed in the Special Attributes Properties pane.

3. Type an alias for the attribute name, if required.
It is recommended to give an alias for the attribute. This name is referred as a parameter for the GetObjectsbyAttribute SOAP request.
4. Select the required check boxes to set the attribute as a primary key or data type as numeric.
The icon of the special attribute is changed.
See [Special Attributes Status](#) for more information on the icons of special attributes. The prefix and namespace of the attribute are displayed in the Prefix and Namespace table below.
If special attribute is set on an object with multiple occurrences, only the first occurrence is retrieved.
5. Click .

Special attribute properties are set on the attributes of the object.

Setting the properties of a sub-context model

To set the properties of a sub-context model:

1. From the Workspace Documents, expand <Project> and select an existing <Business Context Model>.
The business context model appears in the business context modeling environment.
2. Double click the sub-context model object to view its properties. Alternatively, right-click the sub-context model and select **Properties**.
The Sub Context - Sub Context Properties pane appears under the business context modeling environment.
3. Click the **General** tab and do the following:
 - a. Type a description for the sub-context model in the Description field.
 - b. To apply a font size to the sub-context model object, select a font size from **Font Size**.
 - c. To apply a color scheme to the sub-context model object, click  and choose a color.
 - d. Select **Associate Business Context Model**.
 - e. Click  to select an existing business context model from the Select Business Context Model dialog to attach a business context model to the sub-context model.
4. Click the **Annotation** tab to type additional notes or comments on the selected sub-context model.

The sub-context model is configured.

Simple client interface

Number Of Messages	Optional	The number of messages to be sent.
Log Rate	Optional	The rate at which the messages of type 'Session' must be displayed. Thus, if the log Rate is 2, then for every second message sent, the event of type 'Session' will be generated
Send Rate	Optional	The rate at which the messages of type 'Sent' must be displayed. Thus, if the Send Rate is 2, then for every second message sent, the event of type 'Sent' will be generated.
Receive Rate	Optional	The rate at which the messages of type 'Received' must be displayed. Thus, if the Receive Rate is 2, then for every second message sent, the event of type 'Received' will be generated.

Configuration option for SOAP message

This topic describes the configuration option on the AppWorks Platform Simple Client interface.

To invoke a SOAP request without expecting a reply:

- Select the No Reply check box.

For example, few notification messages can be sent and need not be acknowledged on their receipt. Hence, a reply is not expected in such cases

Simple object access protocol

Simple Object Access Protocol (SOAP) is a way to create widely distributed, complex computing environments that run over the Internet using existing Internet infrastructure. SOAP is about applications communicating directly with each other over the Internet in a very effective way.

SOAP is an XML-based messaging protocol. It defines a set of rules for structuring messages that can be used for simple one-way messaging but is particularly useful for performing Remote Procedure Call (RPC)-style request-response dialogues. Messages are routed along the message-path, irrespective of the protocol to which SOAP is bound. This allows processing messages at one or more intermediate nodes in addition to the ultimate destination.

An advantage of SOAP is that program calls are much more likely to get through fire wall servers that screen out requests other than those for known applications (through the designated port mechanism). HTTP requests are usually allowed through fire walls. Hence, the applications can communicate with other applications using SOAP over HTTP.

SOAP enables message exchange over a variety of underlying protocols. As the SOAP messaging framework is independent of the underlying protocol, each intermediary can choose to use a different communication protocol without affecting the SOAP message. However, standard protocol bindings are necessary to ensure high levels of interoperability across SOAP applications and infrastructure. A concrete protocol binding defines exactly how SOAP messages must be transmitted with a given protocol.

A Service Container processes these request or response SOAP messages according to the formal set of conventions defined by SOAP. It is responsible for enforcing the rules that govern the exchange of SOAP messages and accesses the services provided by the underlying protocols through SOAP bindings.

Single sign-on connection parameters interface

Check to enable auditing for SSO	If this option is selected, audit messages are logged for each assertion issued and for each error encountered while validating the identity. Wrong password, expired assertion, and so on are examples of authentication-related details that go into the audit logs.
----------------------------------	--

Single sign-on for developers

The following information is available for application developers that want to use the functionality of the Single Sign-On component in their client applications.

- [Client Side Single Sign-On Library](#)
- Form based Authentication
- [Creating a Custom Desktop](#)
- [Configuring Applications for Anonymous Usage](#)
- [Server Side Single Sign-On API](#)

Sorting notifications

Based on your requirements, you may want to arrange your tasks in a particular order. Inbox has the facility to sort the tasks in ascending or descending order.

To sort notifications:

1. Click **<User> > <Organization> > Inbox > My Inbox**.
The My Inbox page appears.
2. Right-click the column header that you want to sort, select **Sort Ascending**.
The notifications will be sorted in an ascending order based on the order of the column header that you choose. For example, you want to sort the notifications in an ascending order of the alphabets based on the column **Sender**, right-click the header **Sender**, and select **Sort Ascending**.

3. Alternatively, if you want to sort in descending order, right-click the column header and select Sort Descending. The tasks will be sorted in the descending order.

State machine

A Finite State Machine (FSM) or finite state automaton or simply a state machine is a model of behavior composed of a finite number of states, transitions between those states, and actions.

The state machine is used synonymously with state engine.

A state machine represents the sequence of states that an object goes through during its lifetime in response to the events. It captures the unique characteristics of an object at distinct periods of its lifecycle and the events that trigger the changes.

Diagrams of the state machine in Unified Markup Language (UML) depict the various states that an object could be in and the transitions between those states. In many modeling languages, it is common for this type of a diagram to be called a state-transition diagram or even simply a state diagram.

A state represents a stage in the behavior pattern of an object, and like UML Activity Diagrams, it is possible to have initial states and final states. An initial state, also called a creation state, is the one that an object is in when it is first created. A final state is the one in which no transitions lead out. A transition is a progression from one state to another and is triggered by an event that is either internal or external to the object.

A state machine does the following:

- Specifies state changes, which an object performs triggered by events or signals received from other objects.
- Specifies the reactions of the object on events.
- Specifies actions in states.
- Defines protocols, that is, legal sequences of operation calls of a class or interface or of interactions by signals.

The state machine has the following components:

- **State** - A state is an instance in an object's life cycle with unique characteristics. For example: A new customer applying for a credit card would be in a state 'Registered' initially and later on move to 'Verified', 'Approved' or 'Rejected'.

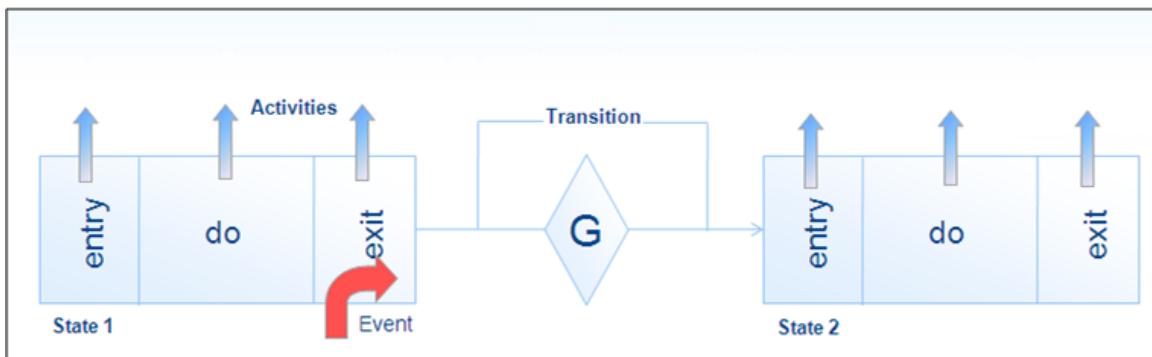
The following are the stages in a state:

- **Entry** - The action performed when an object enters a state. For example, sending a welcome email when the user objects come into a 'Registered' state.
- **Do** - The action performed when an object is in a state. To wait in a particular state is also a Do. For example, long running tasks can be invoked from here, like a business process model

- **Exit** - The action performed when an object leaves a state, generally you can perform cleanup sort of activities here. For example, consider a customer requesting a bank to a loan, the bank after all the verifications rejects the customers application, so before the final state we could model an activity inside the 'Exit' region to send an e-mail indicating the status of the application.
- **Transition** - The movement of an object from one state to another and the line linking the two states is the transition. An object moves from one state to another state in response to some events that are fired on this object's state. For example, in a customer credit card application model, when the application is verified for credit history and if it is found unsatisfactory, the concerned role or person sends a reject event to move the customer's application into a Rejected state.
- **Transaction** - Any transition between two states is in a transaction. For example, consider two states State 1 and State 2, when the object is transiting from State 1 to State 2, this transition, which includes the exit of State 1 and entry of State 2 is performed inside one transaction. While taking the transition, if the entry of State 2 fails, the object returns to the previous State 1 or rather the transition never happened so the object remains in the old state. When a transition fails for some reason, state engine raises an event called state.error. In cases where the user wants to handle this exception case and may want to undo the work done on the exit or entry, he or she can do so by drawing a new transition from this State 2 to a state say 'error' on the event state.error and use the entry or do activities.
- **Guard** - A guard is like a condition on a transition, which allows or not allows a transition. For example, in the ATM withdrawal example, when the customer enters an invalid pin, the transition condition (guard) checks for the correct pin and accordingly accepts or rejects the transition.
- **Activity** - Activity is some work done at some predefined regions inside a state. For example, send a welcome e-mail when a new user registers
- **Effect** - Effect is the result of a transition. Effect is again an activity that can be performed when a transition is chosen. Consider the following example:

In the above example (only a portion of the state model is shown here), a new customer comes in through two different paths. In the first path, a suspect is found and the data steward resolves the suspect and sends for update, in the second path since there are no duplicates found, the customer is directly updated into the database. Since both of them come into an updated state through two different paths, we can model an effect on the second path, where in the customer is updated automatically to send a notification to the data steward about this and this ensures that the data steward is aware of the new entries flowing in automatically.
- **Event** - Event is what triggers the transition of an object from one state to another. You can name an event, 'reject' on the credit card application model. Here trigger and event mean the same thing. A trigger is an event that initiates a transition from one state to another. 'Trigger' is the cause of the transition. An event is something that happens in time. For example: the bus has arrived. The following are the types of events:

- **Signal Event** - An object that is dispatched (thrown) and received (caught). A signal event represents a specific message that initiates a transition when an object receives the expected message (or signal).
- **Call Event** - A call event represents the dispatch of an operation. A call trigger is an event that represents the receipt of a request to invoke an operation. A transition with a call trigger initiates when the called operation is invoked. The difference between a signal event and a call event is that the transition does not start until the called operation is completed. Example - Consider a car with cruise control mode to manage the vehicle speed and other factors while driving on a lane which has strict speed limits (see the following graphic as a sample state model, only a portion of it is shown here).
- **Time Event** - Time event represents the passage of a certain amount of time. For example: You could model a behavior when a new user registers himself with the web site (application) and the applications want to activate this user after 24 hours, in such cases we can model a timer event with a delay of 24 hours, upon expiry of this time, the user is automatically made active and can even send an email confirming the same.
- **State Model** - State model depicts the lifecycle of an object as it passes through each state. State model is the graphical representation of the behavior that is modeled for the object under consideration. Most of the terms explained above are shown in the picture below, please look into the pictorial notations of these terms, the same notations are used inside the state modeler:



After understanding the concepts of state machine, you can now move on to understanding a state model. For information on state models, see State Model.

State model notation using SCXML

State Chart XML (SCXML) is used for denoting State Machine Notation for Control Abstraction 1.0.

SCXML provides a generic state-machine based execution environment based on CCXML and Harel State Tables. Refer this link for more details:

<http://www.w3.org/TR/2005/WD-sxml-20050705/>

MDM feature of AppWorks Platform uses SCXML to represent the state models. Below is an example of a simple state model (source: A Simple State Machine section in <http://www.w3.org/TR/2005/WD-sxml-20050705/>).



The equivalent XML representation of the model follows:

```

<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml">
<state id="S1">
<transition event="Event1">
<target next="S2"/>
</transition>
</state>
<state id="S2">
<transition event="Event2">
<target cond="X>0" next="S1"/>
</transition>
<transition event="Event2">
<target cond="X<0" next="S3"/>
</transition>
</state>
<state id="S3"/>
</scxml>
  
```

Synchronization settings interface

The fields on the Fields on the Synchronization Settings Interface are as follows.

Data Store	The specified name of the data store appears, by default.
Data Entity	The specified name of the selected data entity for which you want to change synchronizing settings appears, by default.
Subscribed Data Entity	The specified name of the hub entity to which the spoke is connected appears, by default.
Subscribed Data Store	The specified name of the hub data store to which the spoke entity is connected appears, by default.
Direction of Subscription	The direction of the subscription appears, by default.
Sniff Changes	Selecting this check box ensures that the differences between the data at the hub and at the spoke are sniffed and buckets are prepared.
Synchronize	Selecting this check box ensures that data between the hub and the spoke is synchronized. The Sniff Changes and Synchronize check boxes are disabled if you have selected the Enable Business Object Life Cycle check box for the hub entity while Creating a Data Entity . In such

cases, these properties have no impact on synchronization during run-time.

Tabular interface of a deployed MDM model

The fields in the Data Stores table are as follows.

Data Stores	The names of the data stores in the MDM model appear. On clicking the name of the data store, the corresponding data entities of the data store appear in Data Entities table; the related subscriptions, spoke entities, hub entities and the direction of the subscriptions of the selected data store appear in another table.
Type	The type of the data store - Hub or Spoke , appear.

Other fields.

Subscription	The names of the subscriptions defined in the selected data store of the MDM model.
Spoke Entity	The names of the spoke entity to which the subscription is configured in the selected data store of the MDM model.
Hub Entity	The names of the hub entity to which the subscription is configured in the selected data store of the MDM model.
Direction	The direction of the subscription that is configured - Spoke to Hub or Hub to Spoke .

Task definition

A Task refers to an activity that can be performed independently and it has two parts as listed below. Task definition will help you with the code form of the task.

- Task Part - An activity that is part of the Task and there can be multiple task parts within a task.
- Related Task - A Task that is referred by this task.

The code snippet below shows the task definition:

```
<Task id="GUID of task " name="name of task" type="UI_TASK"
xmlns="http://schemas.cordys.com/task/1.0/">
<Description>description about the task</Description>
<Properties>
<Property name="showOnStartPage" value="true"/>
</Properties>
<Input/>
```

```

<Output/>
<LaunchingInfo/>
<TaskParts>
<TaskPart id="id of task part" name="name of task part">
<Description/>
<ACL/>
</TaskPart>
</TaskParts>
<RelatedTasks>
<Task id="id of task" name="name of task"/>
</RelatedTasks>
<ACL/>
</Task>

```

Parameters of the Task Definition:

- Shows Application details
- Shows installation details
- Shows content of Application
- Installing Applications
- Uninstalling Applications
- Upgrading Applications

Field	Description	Example
Task	ID - The unique identifier of the task. Name - Name of the task. Type - The type of task.	<ul style="list-style-type: none"> ■ <GUID> ■ Application Management ■ UI task
Description	The description about the task.	Managing Applications
Properties	Any property that a task requires can be added.	showOnStartPage is set to True, to list on the Welcome page.
Input	The Input required to start the task.	
Output	The result obtained on execution of task.	
LaunchingInfo	Information required to start the task.	See below
TaskPart	ID - The unique identifier of the TaskPart. Name - Name of the TaskPart. Description - The description of the TaskPart. ACL - The access rights required to execute TaskPart.	<ul style="list-style-type: none"> ■ cordys_adminui_show_isvp_details ■ Show Application Detail ■ Displays Application Details
Related Task	ID - The unique identifier of the task.	■ cordys_adminui_load_

Field	Description	Example
	Name - Name of the task.	isvp ■ Load Applications
ACL	The access rights that are required for execution of this task.	See below

An example of the LaunchingInfo field is as follows:

```
<LaunchingInfo>
  <Application class="ISVPRegistry" display="hidden" isfolder="false"
  taskbar="false">
    <id>baDaemon</id>
    <url>/cordys/wcp/admin/daemon.htm</url>
    <description>ISV Package Registry</description>
    <caption>ISV Package Registry</caption>
    <icon>/cordys/wcp/theme/default/icon/task/applicationmanager.png</icon>
  <Application class="ISVPRegistry" isfolder="false" menu="false">
    <id>isvRegistry</id>
    <url>/cordys/wcp/isvp/registry.htm</url>
    <description>ISV Package Registry</description>
    <caption>ISV Package Registry</caption>
    <icon>/cordys/wcp/theme/default/icon/task/applicationmanager.png</icon>
  </Application>
  </Application>
</LaunchingInfo>
```

An example of the ACL field is as follows:

```
<ACL>
  <ACObjectTree context="http://schemas.cordys.com/1.0/ldap"/>
</ACL>
```

Task properties interface

The properties mentioned below are applicable for the creation of a User Interface (UI) through a Web page URL.

Field Name	Property	Description
Application URL	Mandatory	Specify the Web page URL of the external UI.
Input Schema	Optional	<p>Input schema serves as the launching information for the UI. An XML schema is provided as an input variable.</p> <ol style="list-style-type: none"> 1. Click  to browse and select the XML schema. The schema definitions that are created appear in the Schema Fragment dialog box.

Field Name	Property	Description
Output Schema	Optional	<p>2. Select a schema fragment and close the dialog box.</p> <p>Output schema is the expected outcome of the UI. This outcome can serve as an input variable to launch another application.</p> <ul style="list-style-type: none"> ▪ Click  to browse and select the XML schema.
Query String Parameters	Optional	Specify the name value pairs that are appended to a Web page URL.

Note: The XML Schema that has more than one schema fragment is not shown in the Schema Fragment dialog box.

Properties pane

The properties mentioned below are applicable for both the Web page URL and User Interface (XForms).

- You can provide these properties by clicking  (UI Task Properties) in the Untitled External User Interface - External User Interface or Untitled User Interface - User Interface window.

Caption	Specify the caption that must appear along with the UI. The title that appears on the titlebar of the application is the combination of Show Title and Caption fields.
Height	Specify the height of the window in pixels (px), em, and percentage (%).
Icon URL	Click  to browse and select the uploaded icon. This icon represents the UI on the Welcome page. Use this option to select an uploaded icon from the local repository. Ensure to meet the design requirements for the Icon selected.
Information	Provide the information about this page that must appear below the title.
Is Internal	Select this option, if the UI will not be used by the end users and is meant for internal consumption. For example, it could be a part of other UIs and need not be exposed. This option is hidden when the Show on Desktop option is selected.
Launch on Startup	Launch the UI on the Welcome page automatically.
Left	Specify the left margin of the window in pixels(px), em, and percentage(%).
Mode	Specify the mode of the UI window appearance. The options are as follows:

	<ul style="list-style-type: none"> ■ Normal - You can resize the window as per your convenience. ■ Fixed - The window dimensions are fixed and the window cannot be resized. However, the title appears on the titlebar of the window. ■ Fixed without Title - The window dimensions are fixed and the window cannot be resized. However, the window appears without any title.
Public Icon URL	Select this option to use a public icon and provide the URL of the icon in the Icon URL field. The value of the Icon URL must point to the absolute HTTP URL of the Icon.
Show on Desktop	The icon of the application is displayed on the Welcome page. By default, this is not selected.
Show Title	Display the title of the UI on its title bar. By default, this is selected.
Tag Names	<p>Tag the UI document with tag labels.</p> <ul style="list-style-type: none"> ■ Multiple tags can be defined with a comma separator. ■ All the tags added to the UI will be displayed as bread crumbs below the text field. ■ When a new tag is added, a prompt for selecting the folder to save the tag will be displayed ■ When the UI is published, the tags will be available in the Tags Cloud.
Top	Specify the top margin of the window in pixels(px), em, and percentage(%).
Width	Specify the width of the window in pixels(px), em, and percentage (%).
XML Data	The XML Data field is used to provide the user defined XML data for the external UI.

Taskbar context menu

The taskbar, similar to the Windows taskbar, contains all the applications that are open at any point of time.

- Right-click on any task in the taskbar to display a shortcut menu with the following options.

Show	Opens the application
Hide	Hides the application
Close	Closes the application

The anchoring feature

In common terms, Anchoring refers to securing the position of any object with respect to its surroundings.

In AppWorks Platform XForms, Anchoring involves specifying fixed positions for user interface (UI) elements on a free layout. This feature enables you to define the absolute positioning of a control with reference to the edges of its immediate parent element.

AppWorks Platform supports anchoring of controls on an XForm, in a composite control, or in a split area. The anchoring feature is available for all freely-positioned controls in a layout. By default, all controls are anchored to the parent control at the top and the left side. The anchoring feature in AppWorks Platform XForms emulates the behavior specified by the W3C consortium. For information about the standards defined by W3C, see the Absolute Positioning module of the W3C- CSS 2.1 specification for Visual formatting model.

You can anchor controls on all sides, or specify varying anchoring distances for each side. Additionally, you can also dock a control such that its edges are equidistant from the parent control or such that there is no space between the control and its parent control. The dimensions of a control that is anchored on all sides change, when the parent control is resized.

On anchoring a control to an edge of its parent control, the corresponding anchor bar is highlighted. Place the cursor on the anchor bar to view the anchor size in an editable field.

Consider the following while anchoring controls:

- A control that is anchored on two sides does not resize when you resize the XForm. When anchoring at two sides, you cannot anchor the control at two opposite sides.
- The default anchor unit is pixel (px).
- The anchor size is distance from the parent's edge to the anchored control's edge. You can specify the anchor size in pixels (px), points (pt), em, ex, millimeters (mm), centimeters (cm), inches (in), and percentages (%). When specified in percentage, the anchor size is calculated based on the visible size of the parent control.
- The controls that cannot be resized vertically such as Input, Password, Output, Check, and Select, cannot be anchored on more than three sides. You also cannot anchor these controls to both the top and the bottom of the XForm simultaneously.
- You cannot anchor a collapsible Groupbox on more than three sides. The collapsible behavior is useful in saving space in an XForm with a vertical layout, automatically repositioning the controls below a collapsible Groupbox. Although these concerns are not relevant to a free layout, anchoring a collapsible Groupbox on all sides is not supported.
- The anchoring feature is not supported for the Toolbar and Element bar buttons, and the controls in a Table control. For these controls, you can anchor the control's container or parent control, as applicable.
- A control that is anchored at the bottom or the right side may not be displayed when the size of the XForm is reduced. The XForm does not display scroll bars to view such controls if they are anchored on all sides. To avoid this, AppWorks Platform

recommends that you do not leave the top or the left side of a control unanchored, especially when the run-time and design-time size of an XForm vary greatly.

The find feature

The Find feature enables you to identify and view selected records in an XForm. You can access the Find feature through the pagination bar of a control. This feature is available for grouping controls (such as Table and Groupbox) attached to a model.

The parameters available in the Find feature depend on the method of the parent control's model. In addition to defining parameters, you can customize your search according to scope and method.

The scope for a search can be set as **Default**, **Local**, or **Global**.

- When set as **Default**, the application searches the local dataset; if no matching record is found, it sends a request to the backend. This is the default scope.
- When set as **Local**, the application searches the local dataset only. No results display if no matching record is found.
- When set as **Global**, the application sends a request to the backend. If no matching record is found, it retrieves the closest possible match.

By default, all methods use the Get, Next, and Previous methods to retrieve and display data. However, when the scope is set as **Global**, you can also use alternate methods to search for records. Consider the following while using the Find feature:

- The Find feature is not supported for non-AppWorks Platform protocol layouts and non-transactional models.
- The Find feature is available only for methods that contain parameters to retrieve data.
- In the case of associated models, this feature is available only if the parent model contains a reference attribute and the associated model contains one or more parameters to retrieve data.
- You cannot edit the reference parameters used in associated models while using the Find feature. You can edit only the non-referenced parameters for an associated model.
- You can define multiple parameters for searching data. Use the xforms-onfind event exposed in the data model to customize search properties.

The type library

The Type Library is an advanced version of Libraries provided by the AppWorks Platform framework. Using Type Libraries, you can easily and effectively enhance the behavior of any control or object.

The Libraries earlier provided by AppWorks Platform also help enhance the default behavior of a control or UI element. When used in multiple controls in a form or multiple forms in an application, a library is loaded only once in the AppWorks Platform Desktop and each control

or form refers to the library. In case of association between two libraries, the multiple reference instances of a library get associated with only a single instance of the second library.

In other words, the AppWorks Platform framework supports multiple instances of references to a library loaded in the AppWorks Platform Desktop. For the association between libraries, only one instance of the library being referred to is used. However, while using Libraries some drawbacks were observed. These are as follows:

- When multiple instances of a library refer to another library, disassociation of a single reference breaks all the references between the two libraries.
- It is not possible to create and extend a library from an existing library.
- Only one Library can be defined per feature. In other words, a feature cannot use multiple libraries. This is because each library is loaded in a separate iframe in the AppWorks Platform Desktop, and communication between two iframes is not possible.

To address these issues, Type Libraries have been introduced. Type Libraries are extended versions of Libraries that provide support for all the above-mentioned features that are not available in Libraries. A unique NameSpace identifies each Type Library, and a dot separated unique identifier (namespace) identifies each reference object in a Type Library.

For example, if the URL of a Type library is cordys/wcp/library/util/autosuggest.htm, its namespace would be wcp.library.util.AutoSuggest. This helps you define any number of Type Libraries inside the same util package.

The validation feature

In most cases, the data that you enter in an XForm needs to be validated before saving it to the database. Validating data ensures that only correct data is sent and saved in the database. Validations can also be used as a means to sequence the appropriate course of action in a process in which varied possibilities exist. For example, depending on user selection, you can use validation to decide which screens to display in a wizard. You can also use validation to provide conditional access to information such as limiting access for a particular role profile.

Validation types

AppWorks Platform provides the following types of validations:

- Data-type based validation
- Server side validation
- Custom validation

Data-type based validation

This type of validation is done on the client side and specifies the type of data to be displayed in a control. For example, you can set validation on a control to receive only date values.

You can apply Data-type validation in the following ways:

Automatic: When using Web services to create an XForm, the validations defined in the WSDL are automatically applied to the respective controls. For example, if the WSDL defines the input values for a control as integer, the control will not accept any value other than an integer.

Manual: By specifying the data type of the control in the Formatting Options dialog box. For information about specifying data types for controls, see [Formatting and Validating Controls](#).

Server side validation

The integration of WS-AppServer with XForms makes it possible to define validations on the server side. You can use the business logic defined in the server to validate data in XForms. You can control and customize the validation constraints using the events provided for integration. For information about the XForms-WS-AppServer integration, see [Integrating with WS-AppServer](#).

Custom validation

In addition to the validations specified by data type and business logic, you can also define custom validations. Custom validations are applied after the client and server validations are performed. The use of custom validations gives you greater control and flexibility while working with data.

Execution order of validations

These validation types are executed in the following order:

1. Data-type based validation: Data type validation is applied when an XForm is loaded and data is rendered in it.
2. Server side validation: Server side validation is applied when the data is rendered, after the client-side validations are applied.
3. Custom validation: Custom validation is applied when you change data in an XForm. Only if the modified data is valid, the onchange event is fired for the control.

Additionally, all validations are also executed before synchronizing data with the backend. If any validation fails, no data is saved to the backend.

Validation messages

Messages regarding failed validations display as notification messages or error messages

Notification messages display in case of failure of data-type based validations and custom validations. An error message displays in case of failure of server side validations or if any type of validation fails when saving data to backend.

AppWorks Platform supports the customization of the following error message types:

System error messages: These refer to default messages that display when a specific activity is performed on the XForm, such as closing an XForm without saving data.

Validation error messages: These refer to validation messages that display for formatting and validation inconsistencies.

As validation errors can occur for various reasons in different scenarios, application users expect to see validation messages specific to their situation. To achieve this, you can customize error messages and notification messages as relevant to your application. For information about the customization of validation messages, see [Customizing Validation Messages](#).

Time zones

The following time zones can be set in AppWorks Platform.

Africa	Abidjan
Africa	Accra
Africa	Addis_Ababa
Africa	Algiers
Africa	Asmera
Africa	Bamako
Africa	Bangui
Africa	Banjul
Africa	Bissau
Africa	Blantyre
Africa	Brazzaville
Africa	Bujumbura
Africa	Cairo
Africa	Casablanca
Africa	Ceuta
Africa	Conakry
Africa	Dakar
Africa	Dar_es_Salaam
Africa	Djibouti

Africa	Douala
Africa	El_Aaiun
Africa	Freetown
Africa	Gaborone
Africa	Harare
Africa	Johannesburg
Africa	Kampala
Africa	Khartoum
Africa	Kigali
Africa	Kinshasa
Africa	Lagos
Africa	Libreville
Africa	Lome
Africa	Luanda
Africa	Lubumbashi
Africa	Lusaka
Africa	Malabo
Africa	Maputo
Africa	Maseru
Africa	Mbabane
Africa	Mogadishu
Africa	Monrovia
Africa	Nairobi
Africa	Ndjamena
Africa	Niamey
Africa	Nouakchot
Africa	Ouagadougou
Africa	Porto-Novo
Africa	Sao_Tome
Africa	Timbuktu
Africa	Tripoli
Africa	Tunis
Africa	Windhoek

America	Adak
America	Anchorage
America	Anguilla
America	Antigua
America	Araguaina
America	Aruba
America	Asuncion
America	Atka
America	Bahia
America	Barbados
America	Belem
America	Belize
America	Boa_Vista
America	Bogota
America	Boise
America	Buenos_Aires
America	Cambridge_Bay
America	Campo_Grande
America	Cancun
America	Caracas
America	Catamarca
America	Cayenne
America	Cayman
America	Chicago
America	Chihuahua
America	Cordoba
America	Costa_Rica
America	Cuiaba
America	Curacao
America	Danmarkshavn
America	Dawson
America	Dawson_Creek
America	Denver

America	Detroit
America	Dominica
America	Edmonton
America	Eirunepe
America	El_Salvador
America	Ensenada
America	Fort_Wayne
America	Fortaleza
America	Glace_Bay
America	Godthab
America	Goose_Bay
America	Grand_Turk
America	Grenada
America	Guadeloupe
America	Guatemala
America	Guayaquil
America	Guyana
America	Halifax
America	Havana
America	Hermosillo
America	Indianapolis
America	Inuvik
America	Iqaluit
America	Jamaica
America	Jujuy
America	Juneau
America	Knox_IN
America	La_Paz
America	Lima
America	Los_Angeles
America	Louisville
America	Maceio
America	Managua

America	Manaus
America	Martinique
America	Mazatlan
America	Mendoza
America	Menominee
America	Merida
America	Mexico_City
America	Miquelon
America	Monterrey
America	Montevideo
America	Monticello
America	Montreal
America	Montserrat
America	Nassau
America	New_York
America	Nipigon
America	Nome
America	Noronha
America	North_Dakota
Center	
America	Panama
America	Pangnirtung
America	Paramaribo
America	Phoenix
America	Port-au-Prince
America	Port_of_Spain
America	Porto_Acre
America	Porto_Velho
America	Puerto_Rico
America	Rainy_River
America	Rankin_Inlet
America	Recife
America	Regina

America	Rio_Branco
America	Rosario
America	Santiago
America	Santo_Domingo
America	Sao_Paulo
America	Scoresbysund
America	Shiprock
America	St_Johns
America	St_Kitts
America	St_Lucia
America	St_Thomas
America	St_Vincent
America	Swift_Current
America	Tegucigalpa
America	Thule
America	Thunder_Bay
America	Tijuana
America	Toronto
America	Tortola
America	Vancouver
America	Virgin
America	Whitehorse
America	Winnipeg
America	Yakutat
America	Yellowknife
Antarctica	Casey
Antarctica	Davis
Antarctica	DumontDUrville
Antarctica	Mawson
Antarctica	McMurdo
Antarctica	Palmer
Antarctica	Rothera
Antarctica	South_Pole

Antarctica	Syowa
Antarctica	Vostok
Arctic	Longyearbyen
Asia	Aden
Asia	Almaty
Asia	Amman
Asia	Anadyr
Asia	Aqtau
Asia	Aqtobe
Asia	Ashgabat
Asia	Ashkhabad
Asia	Baghdad
Asia	Bahrain
Asia	Baku
Asia	Bangkok
Asia	Beirut
Asia	Bishkek
Asia	Brunei
Asia	Calcutta
Asia	Choibalsan
Asia	Chongqing
Asia	Chungking
Asia	Colombo
Asia	Dacca
Asia	Damascus
Asia	Dhaka
Asia	Dili
Asia	Dubai
Asia	Dushanbe
Asia	Gaza
Asia	Harbin
Asia	Hong_Kong
Asia	Hovd

Asia	Irkutsk
Asia	Istanbul
Asia	Jakarta
Asia	Jayapura
Asia	Jerusalem
Asia	Kabul
Asia	Kamchatka
Asia	Karachi
Asia	Kashgar
Asia	Katmandu
Asia	Krasnoyarsk
Asia	Kuala_Lumpur
Asia	Kuching
Asia	Kuwait
Asia	Macao
Asia	Macau
Asia	Magadan
Asia	Makassar
Asia	Manila
Asia	Muscat
Asia	Nicosia
Asia	Novosibirsk
Asia	Omsk
Asia	Oral
Asia	Phnom_Penh
Asia	Pontianak
Asia	Pyongyang
Asia	Qatar
Asia	Qyzylorda
Asia	Rangoon
Asia	Riyadh
Asia	Saigon
Asia	Sakhalin

Asia	Samarkand
Asia	Seoul
Asia	Shanghai
Asia	Singapore
Asia	Taipei
Asia	Tashkent
Asia	Tbilisi
Asia	Tehran
Asia	Tel_Aviv
Asia	Thimbu
Asia	Thimphu
Asia	Tokyo
Asia	Ujung_Pandang
Asia	Ulaanbaatar
Asia	Ulan_Bator
Asia	Urumqi
Asia	Vientiane
Asia	Vladivostok
Asia	Yakutsk
Asia	Yekaterinburg
Asia	Yerevan
Atlantic	Azores
Atlantic	Bermuda
Atlantic	Canary
Atlantic	Cape_Verde
Atlantic	Faeroe
Atlantic	Jan_Mayen
Atlantic	Madeira
Atlantic	Reykjavik
Atlantic	South_Georgia
Atlantic	St_Helena
Atlantic	Stanley
Australia	ACT

Australia	Adelaide
Australia	Brisbane
Australia	Broken_Hill
Australia	Canberra
Australia	Darwin
Australia	Hobart
Australia	LHI
Australia	Lindeman
Australia	Lord_Howe
Australia	Melbourne
Australia	NSW
Australia	North
Australia	Perth
Australia	Queensland
Australia	South
Australia	Sydney
Australia	Tasmania
Australia	Victoria
Australia	West
Australia	Yancowinna
GMT	
GMT+0	
GMT+1	
GMT+2	
GMT+3	
GMT+4	
GMT+5	
GMT+6	
GMT+7	
GMT+8	
GMT+9	
GMT+10	
GMT+11	

GMT+12	
GMT-0	
GMT-1	
GMT-2	
GMT-3	
GMT-4	
GMT-5	
GMT-6	
GMT-7	
GMT-8	
GMT-9	
GMT-10	
GMT-11	
GMT-12	
GMT-13	
GMT-14	
Greenwich	
Europe	Amsterdam
Europe	Andorra
Europe	Athens
Europe	Belfast
Europe	Belgrade
Europe	Berlin
Europe	Bratislava
Europe	Brussels
Europe	Bucharest
Europe	Budapest
Europe	Chisinau
Europe	Copenhagen
Europe	Dublin
Europe	Gibraltar
Europe	Helsinki
Europe	Istanbul

Europe	Kaliningrad
Europe	Kiev
Europe	Lisbon
Europe	Ljubljana
Europe	London
Europe	Luxembourg
Europe	Madrid
Europe	Malta
Europe	Minsk
Europe	Monaco
Europe	Moscow
Europe	Nicosia
Europe	Oslo
Europe	Paris
Europe	Prague
Europe	Riga
Europe	Rome
Europe	Samara
Europe	San_Marino
Europe	Sarajevo
Europe	Simferopol
Europe	Skopje
Europe	Sofia
Europe	Stockholm
Europe	Tallinn
Europe	Tirane
Europe	Tiraspol
Europe	Uzhgorod
Europe	Vaduz
Europe	Vatican
Europe	Vienna
Europe	Vilnius
Europe	Warsaw

Europe	Zagreb
Europe	Zaporozhye
Europe	Zurich
Indian	Antananarivo
Indian	Chagos
Indian	Christmas
Indian	Cocos
Indian	Comoro
Indian	Kerguelen
Indian	Mahe
Indian	Maldives
Indian	Mauritius
Indian	Mayotte
Indian	Reunion
Pacific	Apia
Pacific	Auckland
Pacific	Chatham
Pacific	Easter
Pacific	Efate
Pacific	Enderbury
Pacific	Fakaofo
Pacific	Fiji
Pacific	Funafuti
Pacific	Galapagos
Pacific	Gambier
Pacific	Guadalcanal
Pacific	Guam
Pacific	Honolulu
Pacific	Johnston
Pacific	Kiritimati
Pacific	Kosrae
Pacific	Kwajalein
Pacific	Majuro

Pacific	Marquesas
Pacific	Midway
Pacific	Nauru
Pacific	Niue
Pacific	Norfolk
Pacific	Noumea
Pacific	Pago_Pago
Pacific	Palau
Pacific	Pitcairn
Pacific	Ponape
Pacific	Port_Moresby
Pacific	Rarotonga
Pacific	Saipan
Pacific	Samoa
Pacific	Tahiti
Pacific	Tarawa
Pacific	Tongatapu
Pacific	Truk
Pacific	Wake
Pacific	Wallis
Pacific	Yap
UCT	
UTC	
Universal	
Zulu	

TimeFrame composite control properties interface

The TimeFrame Composite Control Properties Interface helps in configuring properties of the composite control.

Static check box	<ul style="list-style-type: none"> ▪ Select to render the TimeFrame composite control with static timeframe UI at run-time. The Start Time and End Time fields appear under the Static Time Frame section on selecting the Static check box.
------------------	--

	<p>By default, the Static check box is cleared. If the check box is not checked, Timeframe is a rolling timeframe and is rendered with Range and Period selection UI at run-time. By default, Range and Period fields appear under the Rolling Time Frame when the check box is cleared.</p>
Static Time Frame > Start Time	Click  next to Start Time to select the start date to fetch data.
Static Time Frame > End Time	Click  next to End Time to select the end date to fetch data.
Rolling Time Frame > Period	Enter the time period in the text box next to Range to get data from a repeating time period of a specific length that moves continuously, which can be hour, day, week, or month.
Rolling Time Frame > Range	Select the duration in the range of Hour(s) , or Day(s) , or Week(s) , or Month(s) .

Events for the TimeFrame Composite Control are as follows.

ondataclick	<p>Fires when clicked on a data point on the chart. The following example describes its functionality.</p> <pre>function Form_InitDone(eventObject) { <timeframeid>.addListener(ondataclicklistenerFn); } function ondataclicklistenerFn(eventObject) { //eventObject.businessObject will give the businessobject as a JSON object for the clicked datapoint }</pre>
-------------	--

Toolbar context menu

The table below describes the options available to configure a toolbar.

Open	Opens the application referred to
Cut	Cuts the item, for use elsewhere
Copy	Copies the item for further use
Paste	Pastes the item on the target toolbar
Hide	Hides the toolbar item
Delete	Deletes the toolbar item
New	Helps create a new item or toolbar

Toolbar Items	Displays the list of toolbar items currently in the toolbar
Toolbars	Displays the list of toolbars currently in the Explorer
Edit XML	Displays the application definition in an XML editor, where changes can be made and saved
Properties	Displays the property window of the application, which can be modified

Transactional and non-transactional web service

MDM is capable of consuming two types of Web services; the default Web services generated over database metadata, and any external Web service that you can import into AppWorks Platform.

Transactional web service

The Web service that confirms to the AppWorks Platform protocol is classified as Transactional Web service. Each data object in Transactional Web service represents an object that can be used for transactions such as creating, editing, and synchronizing with the back-end.

The following sample illustrates the transactional Web service protocol in AppWorks Platform:

```
<tuple>
<old>
<Employees>
<EmployeeID>1</EmployeeID>
</Employees>
</old>
</tuple>
```

In case of Transactional Web service, each data object (such as 'Employees' in the above sample) is wrapped in a `<tuple>` protocol, which is a AppWorks Platform standard, and contains the state of the transactional object. The following are some transaction formats used in AppWorks Platform.

Format	Description
<code><tuple> <new>...</new> </tuple></code>	This format is used when Web service is created in the client. If the format also carries the <code>sync_id</code> attribute, it denotes that the data is yet to be synchronized with the back-end.
<code><tuple> <old>...</old> </tuple></code>	This format is used when Web service is retrieved from back-end.
<code><tuple> <old>...</old> <new>...</new> </tuple></code>	This format is used when Web service is updated in the server. If the format also carries the <code>sync_id</code>

Format	Description
	attribute, it denotes that the data is yet to be updated to the back-end.

Non-transactional web service

The Web service that does not confirm to the AppWorks Platform standard protocol is referred to as Non-transactional Web service. The Non-transactional Web service can be any of the following:

- Web service from Java Call services
- Web service generated over Business Process Model
- Web service from an external Web service
- Any custom Web service (such as XML Web service)

The following sample illustrates non-transactional Web service.

```
<items>
<item>
<id/>
...
</item>
<item>
<id/>
...
</item>
</items>
```

Triggering a new Process Instance

Before you begin:

- You must have the Developer role to perform this task.

The ExecuteProcess SOAP request is used to invoke a new process instance.

1. From My Application palette, click **Service Test Tool**.
The Service Test Tool dialog box opens.
2. From the **Service Group** list, select **Business Process Management**.
All available web service interfaces for the Business Process Management Service are populated in the **Web Service Interface** list.
3. From the **Web Service Interface** and **Web Service Operation** lists respectively, select the required web service interface and web service operation.
Based upon the selected web service interface, all available web service operations for it are populated in the **Web Service Operation** list.
4. Click **Compose Request**.
The SOAP Request for the selected SOAP operation appears in the SOAP Request pane.

5. In the **SOAP Request** text pane, type the required parameter values and click **Invoke**. The SOAP request and response appears in the **SOAP Response** pane.
6. Type the process name for the receiver element.

A new process instance is triggered.

Trigonometric functions of rule engine

Trigonometric functions allow you to perform trigonometric calculations on data.

The following table describes the various trigonometric functions supported by Cordys, their functionality, and the parameters that have to be specified.

Note: The trigonometric function block accepts only integers as valid inputs.

Trigonometric Functions	Description	Usage
sin	Returns the sine of a number.	$\sin(e)$, where e represents the angle (in radians) for which you want to calculate the sine.
cos	Returns the cosine of a number.	$\cos(e)$, where e represents the angle (in radians) for which you want to calculate the cosine.
tan	Returns the tangent of a number.	$\tan(e)$, where e represents the angle (in radians) for which you want to calculate the tangent.
sinh	Returns the hyperbolic sine of a number.	$\sinh(e)$, where e represents the angle (in radians) for which you want to calculate the hyperbolic sine.
tanh	Returns the hyperbolic tangent of a number.	$\tanh(e)$, where e represents the angle (in radians) for which you want to calculate the hyperbolic tangent.
asin	Returns the arcsine of a number.	$\text{asin}(e)$, where e represents the angle (in radians) for which you want to calculate the arcsine. Note: The argument must be in the range $-1 \leq e \leq 1$, and the return value is in the range $-\pi/2 \leq \text{asin} \leq \pi/2$.
acos	Returns the arccosine of a number.	$\text{acos}(e)$, where e represents the angle (in radians) for which you want to calculate the arccosine. Note: The argument must be in the range $-1 \leq e \leq 1$, and the return value is in the range $0 \leq \text{acos} \leq \pi$.
atan	Returns the arctangent of a	$\text{atan}(e)$, where e represents the angle (in radians) for which you want to calculate the arctangent.

Trigonometric Functions	Description	Usage
	number.	Note: The return value is in the range $-\pi/2 \leq \text{atan} \leq \pi/2$.

Note: e is a numeric expression of type float.

type

This is an optional property that sets the manner of display and selection of values in a Slider component. Three type of Slider components are supported:

- value slider: Both the labels and the selected values are numerical, and bound by the maxValue and minValue properties defined for the slider.
- description slider: Both the labels and the values are predefined. A limited set of alphanumeric options are available for selection, for example, disagree, neutral, and agree.
- combination slider: The labels are predefined and are alphanumeric, whereas the values are numerical and bound by the maxValue and minValue properties. This slider is a combination of the value and description sliders. The description and combination slider components require predefined values. These can be defined through child nodes or using the addOption() or addOptions() methods.

Syntax

Inline HTML:

```
<div cordysType="wcp.library.ui.Slider" id="sliderId" type=sType>
...
</div>
```

Where, the possible values for sType are:

- value(default option)
- description, and
- combination.

Remarks

At design time, you can add the options to be displayed in the Description or Combination slider components. This is demonstrated in the following example, where the Disagree, Neutral, and Agree options are added to the slider.

Additionally, options can also be added and removed dynamically using the addOption(), addOptions(), removeOption(), and removeOptions() methods.

Example:

The following example shows the use of each of the three slider components:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>slider - type property</title>
<script src="/cordys/wcp/application.js"></script>
<style>
.sli
{
width:40%;
}
</style>

</head>
<body >
<p>This page shows three examples of the possible slider types: value, description, combination</p>
<div cordysType="wcp.library.ui.Slider"
class="sli"
type="value"
step="5"
tickStep="10"
subTickStep="5"
labelStep="20"
minValue="0"
maxValue="100"
snap="true"
showTicks="true"
showLabels="true"
showTooltip="true"
showRoller="true"></div>

<div cordysType="wcp.library.ui.Slider"
class="sli"
type="combination"
step="1"
 minValue="0"
 maxValue="10"
 showTicks="true"
 showLabels="true"
 showTooltip="true"
 showRoller="true"
 showSelectedRegion="true">
<div>Disagree</div>
<div>Neutral</div>
<div>Agree</div>
</div>
```

```

<div cordysType="wcp.library.ui.Slider"
  class="sli"
  type="description"
  showTicks="true"
  showLabels="true"
  showTooltip="true"
  showRoller="true">
  <div value="bad">--</div>
  <div value="unsatisfactory">-</div>
  <div value="average">+-</div>
  <div value="good">+</div>
  <div value="perfect">++</div>
</div>
</body>
</html>

```

Types of queries

AppWorks Platform supports creating two types of queries for generating custom Web service operations:

- **Nested Queries:** Built based on a primary table and its relations. The method generated will have an outer query and a collection of zero or more inner queries. The parameters used in the inner query must be from the outer query. Nested queries are also called Associated Queries.
- **Composite Queries:** Built based on multiple tables and their fields. This will be an aggregate of several tables put together; hence, it is called Aggregated Query. There is no hierarchy or nesting involved and all the tables are at the same level.

The format of the response received for a Composite Query may vary with the database servers and JDBC drivers.

Uploading documents

Before you begin:

- [Create a project](#) in CWS.

In the <Application Connector Name> - Application Connector page, the following fields refer external pages or images:

- **Property Page** - HTML page that accepts configuration inputs for the application connector.
- **Image** - Image that is displayed in the configuration page.
- **Page** - (Metadata Details) - HTML Page that knows the protocol of the metadata service and exposes applications over it. CWS provides an option to upload these documents

(HTML pages or images) and then use them.

Documents refer to image or .htm files.

To upload and deploy documents:

1. Switch to the **Explorer** view of the Workspace Documents window. For more information on changing the view, see [Working with Workspace Documents](#).
2. In the **Explorer** view, right-click the project and select **Upload Document**. The <Foldername>- Folder wizard opens.
3. Click  beside File. The Choose File dialog box opens.
4. Browse and upload the intended document. The details of the uploaded document are displayed in the Name, Description, and Extension fields.
5. To use the uploaded document in the interface, do the following:
 - a. Click  beside Property Page or Image or Page depending on your requirement. Associated dialog boxes appear and display the uploaded documents. HTML pages are displayed for Property and Metadata Page fields, while image files are shown for Image field.
 - b. Select the necessary page or image and click **OK**.

The required document is uploaded

Usage of standard attributes

The following table describes the Usage of the Standard attributes.

Lead-Time	For Order-to-Cash process, you can define a Business Measure as Average Lead-time for Process where Customer is ABC Inc. Similarly for activity, you can define, Average Lead-time for Activity (Approval by Manager) where Customer is ABC Inc.
Process name or Activity Name	You can use these attributes in the e-mail template in case you want to trigger email if specified objective is met for the KPI.
Process Start or Process End time	You can use these attributes in Business Measure wherein for a query, Process start or end time input is required.
Activity Start time or Activity End time	You can use these attributes in Business Measure wherein for a query. Activity start/end time input is required.
Status	For Order-to-Cash process, you can define, Business Measures as Total number of orders where Status = COMPLETE, WAITING or ABORTED. Similarly for activity, Count of Orders shipped, you can define Status = COMPLETE, WAITING or ABORTED.

Usage report

A usage report indicates the total number of uniquely named users of a AppWorks Platform product within an organization. The number is calculated per role in the Application package.

The usage report has to be sent at periodic intervals to the AppWorks Platform License Service (which is the licensing authority) for generating invoice. The AppWorks Platform License Service sends valid license keys to all AppWorks Platform installations at the customer site on receipt of these usage reports.

Using a web service in a case model

Before you begin:

- [Create a business process model with a Web service activity.](#)

You can use a Web service in a case model by using a business process model that has a Web service activity in it. You can also use a Web service in a XForm or External User Interface and use that XForm or External User Interface in a case model.

1. [Select a starting point](#) and select  (Case Model) to open an existing case model.
2. From the **Workspace Documents (Explorer view)** > <Solution> > <Project>, drag the required <business process model> (that has a Web service activity) on to the case model.
3. To message map the business process activity to the case model, select the business process activity and click **Message Map** at the bottom.
The Message Map window opens.
4. Do one of the following to create an assignment:
 - Drag the source element from the **Source** column and drop it on the left box in the **Assignments** column. Likewise, drag the target element from the **Target** column and drop it on the left box in the **Assignments** column.
 - Right-click the element or message in the **Target** column, and select **Create Assignment**.

Note: By default, this creates a Fixed Value assignment.

5. Click .
6. [Set the properties of other constructs used in the case model.](#)
7. Right-click in the case modeler and select **Execution > Validate**. Alternatively, go to Workspace Documents <project>, right-click the <case model> and select **Execution > Validate and Build**.
 - If there are no warnings, a status message appears indicating that there are no warnings.
 - If there are warnings resolve them and re-validate the case model.

8. Right-click in the case modeler and select **Execution > Publish to Organization**. Alternatively, go to Workspace Documents <project>, right-click the <case model> and select **Publish to Organization**.
The case model is published to organization.
9. Right-click in the case modeler and select **Execution > Run**.
The case model is instantiated.

You have successfully used required Web service in a case model.

Sample case model using a web service

PATIENT REGISTRATION CASE MODEL

To create the UpdatePatientRecord business process model with Update Patient Record Web service activity:

1. Create a business process model UpdatePatientRecord as shown below using Update Patient Record Web service activity as shown below.
2. In the Workspace Documents (Explorer view), go to <Solution> <Project> expand the tree view of <Update Patient Record> Web service definition set.
3. Drag-drop the Update Patient RecordInput message on to the Start construct of the UpdatePatientRecord business process model as shown below.
4. Double click the Start construct and in the **General** tab view, select **Trigger Type as Message**.
5. Drag and drop the Update Patient RecordOutput message on to the End construct of the UpdatePatientRecord business process model as shown below.
6. Double click the End construct and in the **General** tab view, select **End Type as Message**
7. Click .

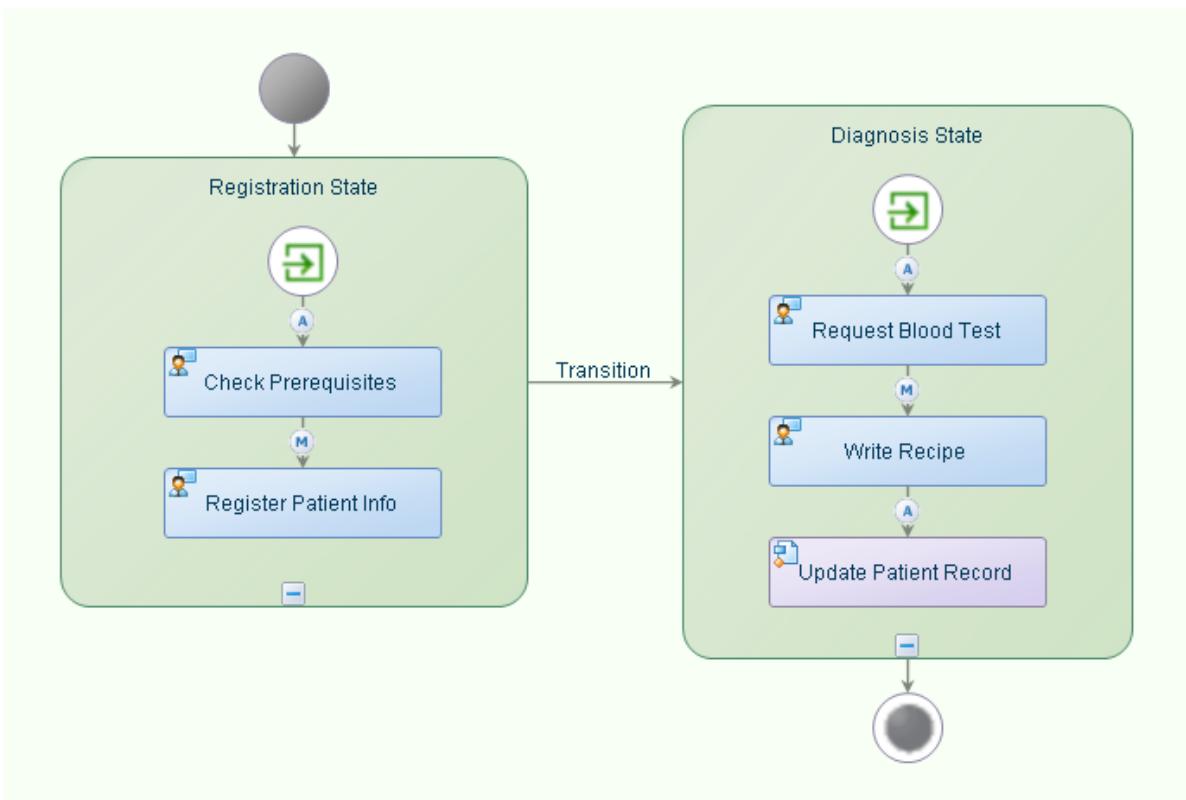


To create the PatientRegistration case model:

1. **Create and Design** a case model as shown below.
2. From the **Workspace Documents (Explorer view)** > <Solution> > <Project> drag the <UpdatePatientRecord> (that has a Web service activity) on to the case model as shown below.
3. To message map the business process activity to the case model, select the UpdatePatientRecord and click the **Message Map** tab.
4. In the Target column of the **Pre Assignments** tab view, expand Update Patient RecordInput message, right-click ns0:PatientID and select **Create Assignment**.

5. In the highlighted text area of the **Assignment** column enter PatientID number and click .

The business process activity is message mapped to the PatientRegistration case model.



Using Intellisense

Intellisense is a feature that prompts you with options when you create or edit the XPath, based on your current position in the XPath.

Using the Intellisense feature, you can insert the appropriate option in the XPath. The XPath text area supports Intellisense. By default, elements and attributes are included in the context menu, if they fit at the position of the cursor. If the start of a function name is already written on that position, the matching function names will also be included as an option.

The following table describes the keyboard controls that can be used in the Intellisense menu.

Select the... Key	To...
Down Arrow	Select the option below the currently selected option. If the last option is currently selected, the first option will be selected.
Enter	Insert the currently selected option into the XPath and close the

Select the... Key	To...
	Intellisense menu.
Esc	Close the Intellisense menu.
Left Arrow/ Right Arrow/ Home/End	Navigate the cursor through the XPath while keeping the Intellisense menu open at the new location of the cursor.
Page Down	Select the last option.
Page Up	Select the first option.
Tab	Insert the currently selected option into the XPath. The Intellisense menu will be moved to the end of the just inserted XPath.
Up Arrow	Select the option above the currently selected option. If the first option is currently selected, the last option will be selected.

In addition, the mouse can also be used to select an option. When an option is clicked, it will be inserted and the Intellisense menu will be closed.

Scrolling in the Intellisense menu

If there are more options available than fit on the screen, a bar appears on the top and the bottom of the Intellisense menu. If you point over these bars, the Intellisense menu will scroll up or down.

If you point over the options in the Intellisense menu, the mouse wheel can also be used to scroll through the options.

If you select an option with the Up or Down arrow keys, then the Intellisense menu will always scroll to a position where the option is visible.

Using the execute condition

The Execute Condition enables business analysts to specify whether the activity should be executed during run time. The associated message map assignments are performed only if the activity is executed. This feature is useful when only one path contains the Activity and the other is modeled as Default in a process model. You can see a small Decision icon at the bottom left of the construct to indicate that the activity includes a condition to be executed.

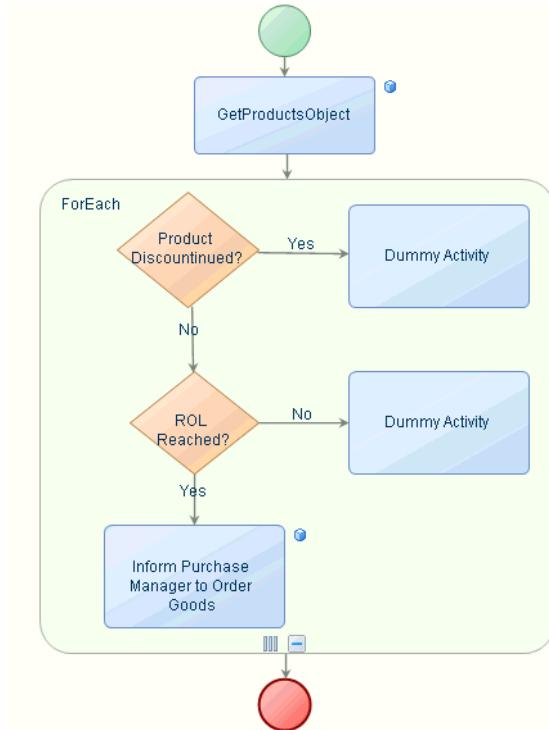
The constructs for which the Execute Condition can be specified are as follows:

- Activity
- Send Message event
- Delay
- Independent Sub-processes
- While
- Until

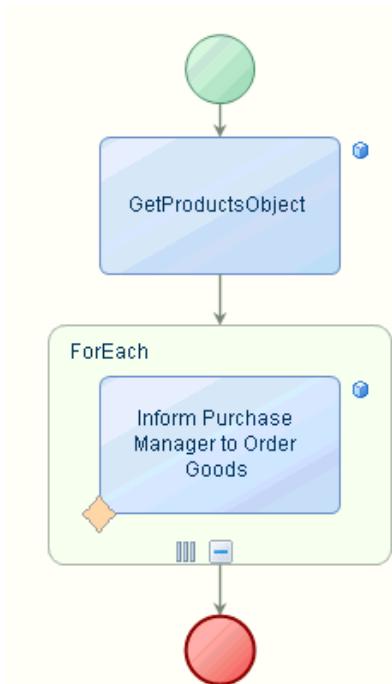
- For Each
- Embedded Sub-Process

For example, suppose you are modeling a business process model where the process checks the re-order level for each product. When the re-order level for a product is reached, a notification is sent to the Purchase Manager to order stock.

In the above example if a business process model is modeled without an Execute Condition, the business process model appears as follows:



If you are modeling the above business process using the **Execute Condition**, the business process model appears as follows:



Specify the Execute Condition in one of the following ways:

1. In the properties of the above constructs, you can specify the condition as /count > 0 in **Execute Condition**.
2. Alternatively, you can enable **Read Execute Condition from Message** so that the **Execute Condition** is picked up dynamically at runtime from the Input Message element of the process; it can also be read by a Web service and be used for all the specified constructs in the business process model. For example, the condition element must contain a condition count > 0 on run time in the **Message Map** and the condition will be evaluated, based on which the activity is executed.

At runtime:

If the Execute Condition is evaluated as...	Description
True	The specified construct is executed.
False	The specified construct is skipped and will not be executed.
Blank (no value is specified)	The construct is always executed.

If the message is not defined in the XPath of the condition, the condition is evaluated as "false". For example, consider the condition /GetemployeesResponse/Employee/EmployeeBalance > 100. If the XPath is not found in the Message Map, this condition will be marked as "false" even though the condition has not been evaluated. This is done to ensure that the process does not wait endlessly for the message to be present in the Message Map. The Wait for Message attribute applies to While, For Each, Until, Decision, Execute Condition, and Debug conditions.

Versioning in XMLStore

AppWorks Platform allows customizations of the objects in the XMLStore. This means that different versions of the same object can exist in the XMLStore. Using this feature, it is possible to define standard XML objects that are available across all organizations, customized XML objects for a specific organization, or even customized XML objects for a specific user.

The possible versions are:

- **isv** - These are the original objects provided by the Application, without any customization. These can be read by all the users of the AppWorks Platform.
- **organization** - These are the Application objects that have been customized for an organization, or objects that have been created for an organization. An object with version as 'organization' can be read only by the users in that organization.
- **user** - These are the Application or Organization objects that have been customized for a user, or objects that have been created for a user. These objects can only be accessed or updated only by that user.

Note: These are the Application or Organization objects that have been customized for a user, or objects that have been created for a user. These objects can only be accessed or updated by that user.

Web service definition set options

The following table describes the Web service Definition set options that you can choose while building an expression for a KPI.

New	<ol style="list-style-type: none"> 1. Select the New option, if you want to provide a new Webservice Definition set to the KPI. This option is selected by default. The Web service Definition Set Name and Web service Definition Set Namespace fields are enabled. 2. Enter a name and namespace in the respective fields. By default, these are populated by the KPI name. The Web service Definition Set Name and Web service Definition Set Namespace fields are mandatory and should not begin with a number. Blank spaces or special characters are not allowed.
Existing	<ol style="list-style-type: none"> 1. Select the Existing option, if you want to apply any existing Web service Definition set. The Existing Web service Definition Set Name field is enabled with  The Existing Web service Definition Set Name is a mandatory field and it should not begin with a number. Blank spaces or special characters are not allowed.

-
2. Click , and select the Web service Definition Set name from the Select Web service Definition Set dialog box.

The **Existing Web service Definition Set Namespace** field remains disabled. Depending on the Web service Definition Set you select, this field is populated with the Namespace automatically.

XForms - A W3C standard

XForms is a set of recommendations for web forms specified by the World Wide Web Consortium (W3C). Adherence to these specifications ensures multi-platform and multiple device support (including hand-held devices, mobile phones, and desktop computers) for such web forms. Web forms built using the W3C XForms recommendations are called XForms.

XForms differ from the traditional HTML forms as they differentiate and separate content from presentation. The separation of data, logic, and presentation simplifies the process of creating forms and reduces the maintenance effort. For more information, see [Model-View-Controller Concept in AppWorks Platform XForms](#).