

Kolban's
Book on
IBM Business
Process
Management

APRIL 2014

Neil Kolban

April 2014

Never memorize what you can look up in books.

- Albert Einstein

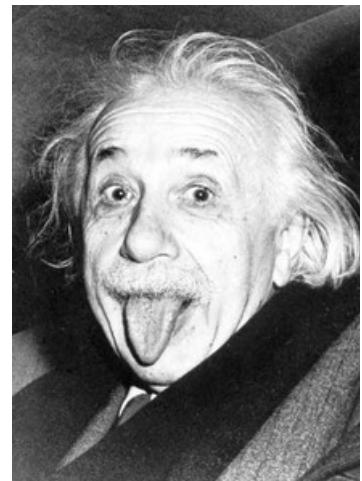


Table of Contents

About this book	30
Book Release History.....	31
What is Business Process Management?.....	33
Overview of IBM Business Process Manager.....	33
History of IBM Business Process Manager.....	33
IBPM Usage.....	34
Modeling a process.....	35
Integrating with IT Systems.....	35
Architecture.....	35
Process Applications.....	35
Process Instances.....	37
Archiving Process Applications.....	37
Process Applications state management.....	38
Changing Process Application settings.....	38
Toolkits.....	39
Tasks.....	41
Task Priority.....	42
IBPM Components.....	42
Component – Process Server.....	42
Component – Process Center.....	43
Starting Process Center.....	46
Component – Process Designer.....	46
Play back sessions.....	47
Starting IBPM Process Designer.....	47
Working in the Designer view.....	51
Working with the Library.....	52
Adding managed files.....	54
Tagging.....	56
Smart Folders.....	59
Validation errors.....	61
Working as part of a team.....	62
Component – Performance Data Warehouse.....	64
Component – Process Center Console.....	64
Component – Process Portal.....	64
Component – Process Admin Console.....	64
IBPM Knowledge.....	65
Product Documentation.....	65
Web Sites.....	65
Public Forums.....	65
IBM RedBooks.....	65
IBM Community.....	66
developerWorks on IBM BPM.....	66
TechNotes.....	68
Requirements and Enhancements.....	80
Product Installation and Configuration.....	82
Part Numbers.....	82
Prerequisites.....	83
Installing Integration Designer.....	83

Installing Business Process Manager Advanced in typical mode.....	90
Performing a custom Install of Process Manager Advanced.....	96
Using BPMConfig to create servers.....	117
Using the PMT to create servers.....	118
Using the command line to create server profiles.....	137
Installing to (DBA) managed databases.....	139
DB2.....	139
MSSQL.....	141
Common for all Databases.....	144
Installing Process Designer.....	146
Starting BPM.....	151
Stopping BPM.....	151
Resetting the databases.....	152
Un-configuring BPM.....	152
Hiding the sample applications.....	152
Frequently Asked installation Installation Questions.....	153
Business Process Definition - BPDs.....	154
Pools and Lanes.....	157
Sequence Lines.....	159
Activities.....	162
Activity Appearance in the BPD diagram.....	165
Data Mapping.....	166
Conditional Activities.....	166
Start Event.....	168
End Event.....	168
Message Start Event.....	169
User Task.....	170
Automatic Flow.....	171
Assigning Activities – Mapping staff to work.....	172
Re-Assigning Tasks and changing group members.....	175
Assignment Examples.....	175
The Four Eyes Example.....	175
Assigning a task to a named user.....	176
Timer Events.....	177
Tracking Intermediate Event.....	180
Ad-hoc Start Event.....	181
Message Intermediate Event.....	182
Message End Event.....	185
Terminate Event.....	186
Error Intermediate Event.....	186
Error End Event.....	188
Gateways, Conditionals and Joins.....	188
Modeling sub-processes.....	194
Modeling Linked Processes.....	195
Modeling Event Sub-processes.....	195
Script Activities.....	197
External Implementation.....	198
Completing an External Implementation – REST API.....	200
Completing an External Implementation – Web Service API.....	202
Application Loops.....	205

Simple Looping.....	205
Multi-Instance Looping.....	206
Looping through a map data type.....	208
BPD Diagram Notes.....	208
Due dates for process and activities.....	209
Time and Holiday Schedules.....	209
Generating Reports.....	210
BPD Settings.....	211
Exposing the Process.....	211
Process Instance Name.....	212
BPD and BPMN.....	212
Concept of a Token.....	212
Deviations from BPMN.....	213
BPD Flow Control.....	215
Rework a set of steps.....	215
Services.....	218
Caching the results of a service.....	220
Human Service.....	221
Exposing the Human Service for starting.....	221
Canceling outstanding Human Tasks.....	224
Informing a user that their task has been canceled.....	226
Timing out a Human Service.....	227
Escalating a Human Task.....	227
Notifying a user that a task is ready for work.....	229
Semantics of closing a browser window.....	229
Ajax Service.....	230
Integration Service.....	235
Advanced Integration Service.....	236
General System Service.....	236
Decision Service.....	236
Business Action Language (BAL) Rule.....	238
Decision Table.....	239
Integrating with WODM.....	241
Service Components.....	241
Coaches.....	243
Server Scripts.....	243
Server Scriptlet.....	244
Modify Task.....	244
Postpone Task.....	245
Decision Gateway.....	246
End Event.....	246
Note.....	246
Error End Event.....	247
Error Intermediate Event.....	247
Invoke UCA.....	248
Catch Exception.....	248
Intermediate Tracking Event.....	249
Stay On Page Event.....	249
Nested Service.....	249
Send Alert.....	250

Web Service Integration.....	251
Java Integration.....	253
Stay on Page Event.....	253
Variables – Process and Service.....	254
System of Record data vs Business Intelligence data.....	255
Creating new Data structures.....	256
Simple Types.....	257
Business Object Types.....	258
List Variables.....	260
Setting defaults on variables.....	261
Making variables search-able.....	261
Defining and using shared objects.....	262
BPD Variables and Service Variables - Mapping.....	265
Variable identity and UUID.....	266
Exposing Business Data for Searches.....	268
Accessing variables from JavaScript.....	269
Environment Variables.....	270
Exposed Process Values (EPVs).....	271
Pre and Post Assignments.....	273
Variables and XML.....	274
Using Variables.....	275
Determining the type of a variable.....	276
The case of the mysteriously changing variables.....	276
User Interfaces and User Interaction.....	280
Coach Views.....	281
General Coach View settings.....	281
Visibility Coach View Settings.....	282
IBM supplied stock Coach Views Controls.....	284
Button.....	284
Checkbox.....	285
Date Time Picker.....	285
Decimal.....	287
Horizontal Section.....	287
Image.....	289
Integer.....	290
Output Text.....	291
Radio Buttons.....	291
Select.....	292
Select Sample 1 – Selection based on other selection.....	294
Table.....	297
Tabs.....	299
Text.....	299
Text Area.....	300
Vertical Section.....	301
Content Box.....	303
Custom HTML.....	305
Styling with CSS.....	305
Making Left Labels.....	309
Adding a TextBox icon.....	310
Styling a Dojo button for BPM.....	310

Data Validation.....	311
Boundary Events.....	315
Dynamically controlling visibility based on user.....	315
IBM sample Coach Views.....	316
Data Change Boundary Trigger.....	316
Data Change Boundary Trigger - Updating one list when another changes.....	317
Building new Coach View types.....	317
Coach View Event Handlers.....	322
Load event handler.....	323
View event handler.....	323
Unload event handler.....	324
Change event handler.....	324
Validate event handler.....	324
Coach View Context.....	325
context.bindingType().....	325
context.bpm.system.....	325
context.bpm.team.....	325
context.broadcastMessage().....	326
context.cancelBoundaryEvents().....	326
context.containDiffLoopingContext().....	326
context.createView(domNode, index, parentView).....	326
context.deleteView(domNode).....	326
context.element.....	326
context.getDomNode().....	326
context.getInheritedVisibility().....	326
context.getSubview(viewId, requiredOrder).....	326
context.htmlEscape(<string>).....	326
context.isTrustedSite().....	326
context.parentView().....	326
context.refreshView().....	326
context.setDisplay(isDisplay, domNode).....	326
context.setUniqueId().....	327
context.subscribeValidation().....	327
context.subview[viewid].....	327
context.trigger(callback).....	327
context.triggerCollaboration(payload).....	327
context.unsubscribeValidation().....	327
context.viewid.....	327
Coach View global data.....	328
com_ibm_bpm_global topLevelCoachViews.....	328
com_ibm_bpm_global coachView.byIdControlId().....	328
com_ibm_bpm_global coachView.byIdDomId().....	328
Coach View data binding and configuration.....	328
Working with Coach View list data.....	331
Dis-allowed Coach Binding data types.....	333
IBM supplied Coach View utility functions.....	333
Generated HTML.....	333
Custom JavaScript in Coach Views.....	334
Using JavaScript libraries other than Dojo.....	334
Generated content in a control.....	335

Making Ajax and REST requests from a Coach View.....	335
Navigating and access to Coach Views on a Coach page.....	337
Learning resources to build new Coach Views.....	338
Debugging Coach Views.....	339
Coach View Construction Tips.....	342
DOM root access in a Coach View.....	342
Editing JavaScript for Coach Views.....	342
Adding a label.....	343
Working with visibility.....	343
Using a Dijit Widget in a Coach View.....	344
Including a custom Dijit Widget in a Coach View.....	344
Describing a Dojo based page declaratively.....	346
Accessing images by CSS.....	347
Commonly used AMD loaders.....	348
Sample Coach Views.....	348
Sample Coach View – trivial "hello world".....	348
Sample Coach View – Data Change Boundary Trigger.....	351
Coach View Templates.....	352
Dashboard Coach Views.....	352
Batch Modify Dialog Control.....	353
Chart Control.....	354
Chart with Time Selector Control.....	361
Dialog Control.....	362
Floating Layout Control.....	364
Gantt Chart Instance Details Control.....	364
Gantt Chart Process Overview Control.....	365
Navigation Controller Control.....	366
Process Diagram Control.....	367
Process Due Date Control.....	368
Process Instances List Control.....	368
Process Summary Control.....	369
Search Control.....	370
Stream Control.....	371
Task List Control.....	371
Tasks Due Control.....	374
Team Roster Control.....	374
Team Summary Control.....	376
Zoom Control.....	376
Exposing a Human Service as a portlet.....	377
Page Flow/ Screen Flow Solutions.....	378
Screen Flow using BPD.....	380
User Interface fragments.....	380
Embedding another HTML page.....	380
Error Handling.....	381
Error Handling in JavaScript	383
BPD Events.....	384
Undercover Agents (UCAs).....	385
Event Initiated UCAs.....	386
Schedule initiated UCAs.....	388
UCAs and queued events.....	389

Disabling UCA processing.....	389
UCAs and Toolkits.....	389
Token Management.....	391
Security.....	396
Security Groups.....	396
Teams.....	398
Securing Access to the repository.....	402
Securing development of a Process Application.....	402
Securing the ability to start an application.....	405
Securing ability to work with tasks (Human Services).....	405
Securing Process Portal capabilities.....	405
Securing access to publish Integration Designer projects.....	408
Lightweight Directory Access Protocol - LDAP.....	409
Tivoli Directoy Server.....	409
Apache Directory Server.....	414
Installation of Apache Directory Server.....	414
Apache Directory Server Studio.....	419
Installation of Apache Directory Studio.....	419
Installation of Apache Directory Server Studio Eclipse Plugins.....	421
Installation of JXplorer.....	426
Getting locked out of WAS.....	428
Configuring WAS for LDAP.....	428
Configuring LDAP for IBPM.....	432
Debugging LDAP.....	433
User Attribute Definitions.....	434
Process Admin Console.....	437
Process Admin - IBM BPM Admin.....	438
Task Cleanup.....	439
Process Admin - User Management.....	439
Group Management.....	441
Bulk User Attribute Assignment.....	441
Process Admin - Monitoring.....	442
Process Admin - Event Manager.....	443
Event Manager > Monitor.....	443
Event Manager > Blackout Periods.....	443
Event Manager > Synchronous Queues.....	444
Event Manager > Event Managed JMS Error Queue.....	444
Admin Tools > Manage EPVs.....	444
Process Admin – Saved Search Admin.....	446
Process Admin – Process Inspector.....	447
Process Admin - Installed Apps.....	449
Customizing the Process Admin Console.....	449
Process Portal.....	451
Process Portal – My Tasks.....	454
Process Portal – Saved Searches.....	460
Starting Ad-Hoc entry points.....	461
In-line Tasks in process portal.....	462
Team Performance.....	463
Team Tasks.....	464
Process Performance.....	465

Quick Stats.....	466
Turnover Rate.....	466
Average Duration.....	467
Instances In Progress.....	467
Diagram View.....	468
Gantt View.....	468
Set Path View.....	469
Searching in Process Portal.....	469
The data used to populate the Process Performance dashboard.....	470
Customizing Process Portal look and feel.....	470
Customizing the login screen.....	473
Versioning Solutions.....	475
Snapshots.....	475
Managing Snapshots.....	476
BPMDelteSnapshot.....	476
BPMSnapshotCleanup.....	476
BPMShowProcessApplication.....	477
Migration of in-flight BPMN process instances.....	478
BPD Integrations.....	483
Outbound Web Services.....	483
Setting up security for outbound Web Services.....	491
Inbound Web Services.....	491
Testing an Inbound Web Service with soapUI.....	494
Invoking a BPD as a Web Service.....	497
Invoking an IBPM Web Service from a Java POJO.....	498
Web Services and data types.....	501
Simple Data Type.....	502
Complex Data Type.....	503
Nested Complex Data Type.....	504
Array of Complex Data Type.....	505
Java Message Service – JMS.....	513
JMS – Sending and receiving from queues.....	513
JMS – Triggering a UCA.....	515
JMS Client Tools.....	517
Hermes JMS.....	523
WebSphere Default Messaging.....	530
REST Integration.....	531
The REST Functions.....	532
Handling REST errors.....	534
The REST API Tester.....	534
Working with REST Search Queries.....	537
Working with REST Task Instance Queries.....	541
Working with REST Task Instances.....	543
Getting Task and Instance details through REST.....	543
Getting a template for a task.....	546
Finishing a Task.....	547
Starting a Task.....	547
Claiming a task.....	548
Re-assigning a task.....	549
Getting Task variable values.....	549

Getting the client details of a task.....	549
Working with REST Processes.....	550
Starting a process from REST.....	550
Getting details of a process instance using REST.....	551
Getting details of a process model from REST.....	552
Starting a UCA from REST.....	553
Moving a token within a process using REST.....	553
Retrying a failed instance.....	553
Listing and starting Ad-Hoc Events from REST.....	554
Working with REST Services.....	554
Starting a service from REST.....	554
Getting data from a Service.....	555
Setting data into a Service.....	555
Getting a Service Model using REST.....	556
Working with Exposed Items.....	559
Getting Exposed Processes.....	559
Working with REST Users and Groups.....	559
Working with Process Apps.....	561
Working with REST and heritage process document attachments.....	561
Heritage - Adding a document to a process instance using REST.....	562
Heritage – Getting a list of attached documents.....	562
Working with REST Asset Lists.....	562
Getting Project branchIds and Snapshots.....	563
Getting Business Object definitions.....	564
Working with REST retrieved diagrams.....	565
Process Visual Model.....	565
Executing JavaScript in the context of a process from a REST request.....	570
Calling REST from Java.....	571
REST encoded UUIDs/GUIDs.....	572
REST input and output data types.....	573
REST Security.....	573
Database Integration.....	576
Integration with supplied DB connectors.....	576
Using LiveConnect and JDBC.....	578
Example – Selecting rows from a table.....	579
Security with Database Interaction.....	579
JDBC definitions needed to access databases.....	579
Service to Insert a row.....	580
Service to Delete a row.....	580
Service to Update a row.....	581
Calling stored procedures.....	581
Mapping IBM BPM Data types to DB data types.....	582
Java Integration.....	583
Java source level Debugging.....	587
eMail.....	591
Sending an email.....	591
Receiving an email.....	592
Send an email and receive a response.....	594
Installing hMailServer as a test EMail provider.....	595
WPS and SCA.....	603

Asynchronous Invocation of an SCA Module.....	603
Process Scheduling with Job Scheduler.....	610
The nature of a IBPM Job.....	610
The Job Scheduler Java API.....	610
An illustrative solution.....	610
Debugging.....	626
Debugging with Inspector.....	626
Debugging the environment.....	632
Browser tabs and Process Inspector.....	632
Logging.....	633
Tracing Web Service SOAP traffic.....	634
Working with IBM Defect Support.....	635
Raising defects with IBM.....	635
Development.....	636
Sharing projects with others.....	636
Naming conventions and recommendations.....	636
Documentation.....	637
Operations.....	639
Configuration Files.....	639
Networking.....	640
WAS Server.....	640
Stopping the server.....	640
Windows Services.....	640
Port Numbers.....	640
WAS Admin Console.....	640
JDBC Resources.....	641
WAS Security.....	641
The wsadmin command.....	641
Java programming for admin commands.....	641
Changing Passwords.....	642
File Structures.....	642
Operational Databases.....	642
Process Server Database.....	643
Performance Data Warehouse Database.....	643
Defining custom databases.....	643
SI Bus Resources.....	647
Cleaning/removing completed processes.....	648
Adding and removing Process Servers from Process Center.....	650
Deployment of applications to servers.....	650
Offline Application Deployment.....	650
Un-Deployment of applications from servers.....	654
Modification of Process Instance data.....	655
Performance.....	656
The Event Manager - Tuning attributes.....	656
Monitoring using the Process Admin Console.....	657
Process Admin Console – Monitoring > Instrumentation.....	657
Process Admin Console – Monitoring > Process Monitor.....	658
DB2 Database Performance.....	660
Monitoring DB2.....	660
Useful scripts and tools.....	662

Viewing database table contents.....	662
Recovery options.....	664
JavaScript in IBPM.....	665
Editing JavaScript.....	665
IBPM JavaScript name spaces.....	665
Reusing JavaScript.....	666
BPD Data Types.....	666
Data Type – ConditionalActivity.....	667
Data Type – Map.....	667
Data Type – NameValuePair.....	668
Data Type – ProcessSummary.....	668
Data Type – Record.....	668
Data Type – SQLResult.....	669
Data Type – String.....	670
Data Type – TaskListData.....	670
Data Type – TaskListItem.....	671
Data Type – TaskListInteractionFilter.....	672
Data Type – TaskListProperties.....	672
Data Type – Team.....	673
Data Type – TeamDashboardSupport.....	673
Data Type – TWAdhocStartingPoint.....	673
Data Type – TWDate.....	674
Data Type – TWDocument.....	676
Data Type – TWHolidaySchedule.....	677
Data Type – TWLink.....	678
Data Type – TWManagedFile.....	678
Data Type – TWModelNamespace.....	678
Data Type – TWObject.....	679
Data Type – TWParticipantGroup.....	680
Data Type – TWProcess.....	680
Data Type – TWProcessApp.....	681
Data Type – TWProcessAppSnapshot.....	681
Data Type – TWProcessInstance.....	682
Data Type – TWProcessPerformanceMetric.....	683
Data Type – TWReport.....	684
Data Type – TWRole.....	684
Data Type – TWSearchColumn.....	685
Data Type – TWSearchColumnMetaData.....	687
Data Type – TWSearchCondition.....	687
Data Type – TWSearchOptions.....	688
Data Type – TWSearchResults.....	688
Data Type – TWSearchResultRow.....	688
Data Type – TWService.....	688
Data Type – TWServiceTypes.....	689
Data Type – TWStep.....	689
Data Type – TWTask.....	690
Data Type – TWTeam.....	691
Data Type – TWTimePeriod.....	692
Data Type – TWTimeSchedule.....	692
Data Type – TWTimerInstance.....	694

Data Type – TWUser.....	695
Data Type – XMLDocument.....	695
Data Type – XMLElement.....	696
Data Type – XMLNodeList.....	696
JavaScript Libraries.....	697
JavaScript package - tw.system.*.....	697
Method: executeServiceByName(name, inputValues).....	699
Method: retrieveTaskList().....	699
JavaScript Package - tw.system.step.*.....	700
JavaScript Package - tw.system.org.*.....	700
JavaScript Package - tw.system.model.*.....	701
Getting a list of Process Apps.....	701
JavaScript Package - tw.object.*.....	702
JavaScript Package - tw.system.model.....	702
Creating Business Object instances in JavaScript.....	704
Variables in a service.....	705
The Dojo Toolkit for JavaScript.....	705
Searching for processes and tasks from JavaScript.....	706
Calling Java through LiveScript.....	709
Working with XML in JavaScript.....	709
Working with document attachments in JavaScript.....	710
Working with JSON in IBPM.....	711
JavaScript Fragments.....	712
Starting a new process.....	713
Getting the current process instance.....	713
Getting the current userid.....	713
Starting an external application.....	713
Returning the owner of a task.....	713
Extracting a managed file.....	714
Generating a Random Number or string.....	714
Key Performance Indicators (KPIs).....	715
Custom KPIs.....	715
Associating KPIs with BPD activities.....	717
Service Level Agreements (SLAs).....	717
Creating SLAs.....	718
Dash-boarding.....	721
Architecture.....	721
Tracking data.....	721
Tracking Groups Overview.....	725
Database Structure for a Tracking Group.....	727
Database Structure for TRACKINGGROUPS view.....	728
Database Structure for TRACKINGPOINTS view.....	728
Database Structure for TASKS view.....	728
Database Structure for PROCESSFLOWS view.....	729
Miscellaneous Tracking Data Notes.....	730
Removing tracking data tables.....	730
Minimizing Tracking Group entries.....	731
Timing Intervals.....	731
Performance Data Warehouse (PDW) SQL Snippets.....	734
How many have started but not finished (in progress).....	734

Number of items of different types/day.....	735
Getting the last state of a process.....	736
Breakdown of how many in which step.....	736
Last Autotracked row.....	737
Counts of processes started vs completed over an interval.....	737
General DB/SQL useful functions for reports.....	737
Date formatting.....	737
Mapping BPM date types to DB TIMESTAMP types.....	738
Number of items in the current week.....	738
Counting when a column equals a value.....	738
Selecting records within a date range.....	738
Reporting with Microsoft Excel.....	738
Developing Custom Dash-boards.....	738
A Sample Dashboard.....	739
Generating sample data.....	743
Simulation and Optimization.....	744
Defining simulation values.....	744
Fixed distribution type.....	745
Uniform distribution type.....	745
Normal distribution type.....	746
Gateways and simulation.....	746
Arrival rate of simulation items.....	747
Simulating the cost of execution.....	747
Available staff members for tasks.....	748
Simulation Analysis Scenarios.....	749
Simulation Profiles.....	750
Running Simulations.....	751
Optimizing a Process.....	754
Simulation Tutorials.....	757
Experiment 1 – A basic start.....	757
Experiment 2.....	760
Experiment 3.....	761
Experiment 4 – Path analysis.....	763
IBPM Web API – Web Services API Access.....	765
Starting a Process.....	770
Notes on Object Factory.....	772
Executing a search.....	772
Getting the details of a task.....	773
Web API Data types.....	773
CustomProperty.....	773
ExternalActivity.....	773
ExternalActivityAttachment.....	774
ExternalActivityData.....	774
Parameter.....	774
Process.....	774
Search.....	774
SearchColumn.....	775
SearchColumnMetaData.....	775
SearchCondition.....	776
SearchMetaData.....	776

SearchResults.....	776
SearchResultRow.....	776
Task.....	776
TaskStatus.....	777
User.....	778
Variable.....	778
Variable.....	780
Exposed item types.....	780
Testing Web API services with soapUI.....	780
Building a Java Client.....	783
Building a Flex Client.....	785
System Data Toolkit.....	791
IBM Supplied Business Process Definitions.....	791
Send SLA Violation Email.....	791
IBM Supplied General System Services.....	791
Default BPD Event.....	791
Default System Service.....	791
Extract XML Validation Results.....	791
IBM Supplied Historical Analysis Scenarios.....	791
IBM Supplied Human Services.....	791
Default Human Service.....	791
lsw Conditional Activity Selection Coach.....	792
Fire Default BPD Event.....	793
IBM Supplied Integration Services.....	793
Read Text File.....	794
Send E-mail via SMTP.....	794
SLA Send Alert Email.....	794
SQL*.....	794
Update ALL SLA Statuses.....	794
Update SLA Status.....	794
Write Text File.....	794
IBM Supplied KPIs.....	795
IBM Supplied Layouts.....	795
IBM Supplied Participant Groups.....	795
IBM Supplied Server Files.....	795
Integration.jar.....	795
IBM Supplied Tracking Groups.....	795
IBM Supplied Undercover Agents.....	795
IBM Supplied User Attribute Definitions.....	795
IBM Supplied Variable Types.....	795
Cookbook Scenarios.....	796
Scenario - Asynchronously starting one process from another.....	796
Scenario – Making a process extensible by others.....	797
Scenario – Starting a process from an external browser.....	797
Additional Toolkits.....	800
Kolban JavaScript Toolkit.....	800
Blueworks Live.....	802
Creating a Space.....	803
Creating a Process.....	804
Discovery Map.....	805

The Process Diagram.....	811
Working with Sub processes.....	812
Editing mode vs Viewing mode.....	813
Sharing the solution.....	814
Creating a Process from a Template Library.....	814
Automating a Process.....	817
Accessing processes from Blueworks in Process Designer.....	818
	821
IBM Business Process Manager – Advanced.....	822
Information Sources.....	823
Redbooks.....	823
DeveloperWorks.....	824
Other.....	832
Integration Designer.....	832
Inter-operating between a BPMN process and an SCA module.....	836
Interface Types Exchange.....	841
Data Types Exchange.....	842
BPDs exposed to an AIS.....	843
Deploying for testing from ID.....	844
Returning faults from an AIS.....	847
Manually un-deploying an SCA module.....	847
Service Component Architecture.....	849
SCA Interfaces and References.....	852
The SCA Import Component.....	854
The SCA Export Component.....	855
SCA Interfaces.....	856
SCA Business Objects.....	858
SCA Event Sequencing.....	860
SCA Store and Forward.....	862
SCA Versions.....	862
Installing an SCA Module through scripting.....	862
SCA Java Components.....	864
Calling a Reference.....	864
Java Component Error Handling.....	864
Client Programming with Java Components.....	865
BPEL - Business Process Execution Language.....	867
BPEL Activities.....	868
The notion of a BPEL Process.....	868
Variables in BPEL.....	869
Receive Activity.....	869
Reply Activity.....	870
Invoke Activity.....	871
Assign Activity.....	871
Wait Activity.....	871
Choice Activity.....	872
While Loop Activity.....	872
Repeat Until Loop Activity.....	872
For Each Loop Activity.....	872
Parallel Activities Activity.....	873
Scope Activity.....	873

Generalized Flow Activity.....	873
Terminate Activity.....	873
Throw Activity.....	873
In-line Human Task Activity.....	874
Snippet Activity.....	874
BPEL Transactionality.....	874
BPEL and AIS.....	875
BPMN compensations using BPEL.....	875
Mediations and ESB.....	882
Service Message Object (SMO).....	882
Mediation Primitives.....	885
Input primitive.....	885
Callout primitive.....	885
Business Object Map primitive.....	885
Custom Mediation primitive.....	885
Data Handler primitive.....	886
Database Lookup primitive.....	887
Endpoint Lookup primitive.....	888
Event Emitter primitive.....	888
Fail primitive.....	888
Fan In primitive.....	888
Fan Out primitive.....	888
Flow Order primitive.....	889
Gateway Endpoint Lookup primitive.....	889
HTTP Header Setter primitive.....	890
JMS Header Setter primitive.....	890
Message Element Setter primitive.....	890
Message Filter primitive.....	891
Message Logger primitive.....	891
Message Validator primitive.....	891
MQ Header Setter primitive.....	891
Policy Resolution primitive.....	892
Service Invoke primitive.....	892
Set Message Type primitive.....	893
SLA Check Primitive.....	893
SOAP Header Setter Primitive.....	893
Stop primitive.....	893
Subflow primitive.....	893
Trace primitive.....	893
Type Filter primitive.....	893
UDDI Endpoint Lookup Primitive.....	893
Mapping primitive.....	894
Core transformations.....	894
XSLT and XPath functions.....	896
Calling Java.....	896
The catalog of Data Maps.....	896
The Service Gateway pattern.....	897
Proxy Gateways.....	898
SOAP Message Attachments.....	900
Tracing.....	901

Business Calendars.....	903
JCA Adapters.....	903
Flat File Adapter.....	903
Sample – Reading a Flat File with delimited data.....	903
JDBC Adapter.....	921
Advanced Web Services.....	925
WS-Notification.....	925
Subscribe Request.....	926
Publishing a message.....	929
Topics for WS-Notification.....	932
Simple Topic Expression.....	932
Concrete Topic Expression.....	932
Full Topic Expression.....	932
Building a Notification Subscriber.....	932
A WS-Notification Sample.....	934
Implementation notes.....	945
Bugs and Fixes.....	946
References – WS-Notification.....	946
WS-Addressing.....	946
WS-Security.....	948
Building a secure Web Service.....	953
Testing Web Services security with TCP/Mon.....	954
JAX-WS.....	955
JAX-WS Java SE stand alone Service Provider.....	955
Business Space.....	956
Administering SCA Modules.....	958
Exercises and Case Studies.....	959
Exercise – Education class enrollment.....	959
Looking for home	964
Human Task Manager.....	965
Work Baskets.....	965
Work Basket Roles.....	966
Work Basket Business Space widgets.....	966
Human Task Definition for Work Baskets.....	968
Business Categories.....	969
Parallel Tasks.....	969
Completion Selection.....	970
Work Items.....	971
Authorization and Staff Resolution.....	971
Staff Resolution Criteria (aka Verbs).....	972
Verb Descriptions.....	972
Group Search.....	972
Group Members.....	973
Dynamic user selection.....	974
Staff Resolution plugins.....	974
Custom Human Task staff resolutions.....	974
Staff resolution plugin language – The Staff Query.....	978
User registry queries.....	979
LDAP queries.....	979
XSLT Mapping to Staff Resolution plugin language.....	981

Programming tips.....	986
Caching of staff resolutions.....	986
Tracing Staff resolution.....	986
Worked Examples.....	987
Group and User List intersection.....	987
Group and Group intersection.....	989
StaffQueryResultPostProcessorPlugin.....	990
Replacement Expressions.....	991
Stand-alone Expressions.....	991
Getting data from Tasks.....	994
Human Task User Interfaces.....	994
Human Task API Event Handlers.....	994
Query Tables.....	1006
Query Table Editor.....	1006
Installation of Query Table Editor.....	1006
Query Table programming.....	1009
Using Query Tables with Business Space.....	1010
Business Objects.....	1022
SDO Programming Model v1.0.....	1023
SDO Programming Model v2.0.....	1023
Recording changes.....	1023
Splitting namespaces across projects.....	1023
Business Object Programming.....	1024
Sequenced Objects.....	1024
Nested BOs.....	1024
Business Object Inheritance.....	1026
Application Specific Information (ASI).....	1027
SDO Programming.....	1027
Walking a DataObject.....	1027
Create and Populate a DataObject.....	1028
IBM DataObjectUtils.....	1028
SCA Business Objects.....	1029
Creating an SCA Business Object.....	1029
Converting to/from XML.....	1029
XML documents with no namespace.....	1030
SCA.....	1030
Asynchronous processing.....	1030
SCA Import and Export protocol bindings.....	1030
Web Services.....	1031
SOAP/HTTP.....	1031
SOAP/JMS.....	1031
JMS.....	1035
HTTP.....	1035
MQ.....	1035
MQ Correlation messages.....	1037
Testing MQ bindings.....	1038
MQ messages and XML.....	1038
EJB SCA Bindings.....	1039
Auto-generated WSDL to Java Bridge Component.....	1039
DataHandlers.....	1040

XML Data Handler.....	1040
Default XML Data Handler serialization.....	1043
XML Documents that are different from Business Objects.....	1045
Fixed Width DataHandler.....	1046
Custom DataHandlers.....	1048
DataHandler/DataBinding properties.....	1050
BindingContext.....	1051
Creating a Business Object.....	1052
The Binding Registry.....	1052
Calling a DataHandler from a DataBinding.....	1052
Creating a UI Configuration class.....	1053
HTTP DataBinding.....	1056
References.....	1056
DataObjects.....	1057
Creating an SCA Business Object.....	1057
Converting to/from XML.....	1057
XML documents with no namespace.....	1058
Advanced BPEL.....	1058
Variable initialization.....	1058
Scope.....	1059
BPEL - Java Mapping.....	1059
Variable Properties.....	1059
Endpoint References.....	1060
BPEL Endpoint Reference variable type.....	1060
JavaSnippet access.....	1061
getServiceRefFromPartnerLink.....	1061
Query Properties.....	1061
Implementation.....	1062
Performance.....	1062
Activities.....	1062
Inline Human Tasks.....	1062
Previous task owners.....	1062
Invoke.....	1063
Expiration.....	1063
JavaSnippet.....	1065
Standard Functions.....	1067
Custom Properties.....	1068
References.....	1068
BPEL Process Migration.....	1068
References.....	1068
Versioning.....	1068
User Interfaces.....	1069
Business Space.....	1069
Events	1069
Custom Widgets.....	1069
iWidget Mode.....	1070
iWidget Structure.....	1070
iWidget Modules/Data Structures.....	1072
Misc.....	1079
iWidget editor.....	1079

BSpaceWidgetRegistry.....	1082
BSpaceGeneralHelper.....	1083
Registering Widgets.....	1084
The Catalog File.....	1085
Registry Endpoints XML.....	1088
JavaScript for a new widget.....	1089
Modes.....	1090
HTML rewriting.....	1090
Debugging iWidget JavaScript.....	1091
Event handling.....	1091
Dojo Level Workarounds.....	1093
Debugging and Problem determination.....	1093
Custom Widget Walk through.....	1094
Using RAD 8.0.3 to create an iWidget.....	1105
Mashup Center.....	1107
Multiple Browser instances and BPM.....	1107
Business Space Supplied Widgets.....	1109
Business Configuration.....	1109
Business Rules.....	1109
Human Task Management.....	1109
Escalations List.....	1110
Human Workflow Diagram.....	1110
My Team's Tasks.....	1111
My Work Organizer.....	1111
Process Definitions List.....	1111
Processes List.....	1111
Task Definitions List.....	1111
Task Information.....	1111
Tasks List.....	1111
Problem Determination.....	1112
Solution Administration.....	1112
Solution Operation.....	1112
User Management.....	1112
Viewers.....	1112
Web Viewer.....	1112
Script Adapter.....	1112
Visual Step.....	1112
JSPs.....	1112
WebSphere Portal.....	1112
Liferay.....	1113
Installing Liferay on WAS.....	1113
Using the Liferay Tomcat bundle.....	1114
Developing Liferay Portlets.....	1114
Lotus Forms.....	1115
Dojo.....	1115
Custom Dojo development environment.....	1115
Flex and Business Space.....	1116
Widgets that use Flex.....	1116
Flex Custom Widgets in Business Space.....	1116
Building a new Custom Widget for Flex.....	1117

Environment of the Flex application.....	1120
Widget Instance Configuration.....	1121
var attributes:Object.....	1121
event: endConfigure.....	1121
Making REST requests.....	1122
Handling returned XML.....	1124
Cross domain security.....	1124
Problems with PUT and DELETE requests.....	1125
Widget to Widget Event Handling.....	1125
Sending Events.....	1125
Receiving Events.....	1126
Working with WPS Human Task / Process Data.....	1127
Mapping Data types and controls.....	1129
String.....	1129
Int.....	1129
Date.....	1129
Working with repeating/array data.....	1130
Working with AnyType.....	1130
Performing input validation.....	1130
Single User Workflow.....	1130
Flex related REST Problems.....	1130
Elixir.....	1131
References – Elixir.....	1131
References – Flex Widgets.....	1131
Custom Process Portals.....	1132
Types of Custom Portal.....	1133
Generic Widgets for Custom Portals.....	1134
The Task List Widget.....	1134
The Task Table Widget.....	1136
The CoachViewer widget.....	1137
Determining when the current Coach has completed.....	1137
Using the Custom Portal widgets.....	1138
Writing custom portals using Coaches.....	1140
Apache Flex.....	1140
Flex for Mobile.....	1140
Installing FlashDevelop.....	1141
Google Charts.....	1143
Providing data to Google Charts.....	1143
Google Chart – Gauge.....	1143
Google Web Toolkit – GWT.....	1143
Making REST calls from GWT.....	1144
Working with JSON in GWT.....	1145
Building GWT Apps in Eclipse.....	1145
Notes on GWT Widgets.....	1149
RootPanel.....	1149
DockPanel.....	1150
DialogBox.....	1150
DisclosurePanel.....	1150
Composite.....	1150
ListBox.....	1150

FlexTable.....	1150
Grid.....	1151
HTML.....	1151
PushButton.....	1151
FileUpload.....	1151
GWT Cell processing.....	1151
Cell Selection.....	1153
Adding Data to a Cell widget.....	1153
CellList notes.....	1154
CellTable notes.....	1154
DataGrid Notes.....	1155
Cell Tree Notes.....	1155
GWT Modules.....	1155
Naming conventions and recommendations.....	1156
The Entry-Point class.....	1156
Source Path.....	1156
Public Path.....	1156
Module declaration files.....	1156
Working with resources.....	1157
GWT Layouts and resizing.....	1157
RootLayoutPanel.....	1157
StackLayoutPanel.....	1157
TabLayoutPanel.....	1158
Things that seem to scroll ok.....	1158
UIBinder – XML descriptions of screens.....	1158
Using CSS for changing the appearance of UI.....	1159
The GWT Editor Framework – Mapping data to widgets.....	1160
Working with XML in GWT.....	1160
Integrating with native JavaScript.....	1160
Building custom GWT Widgets.....	1160
Adding Event Handling.....	1160
Using Google Charts with GWT.....	1163
Deploying a GWT application.....	1165
Debugging and Logging GWT applications.....	1170
GWT Libraries – Smart GWT.....	1170
Smart GWT – ListGrid.....	1171
Smart GWT – DataSource.....	1172
Smart GWT – Drawing.....	1174
Integrating GWT with IBPM Coaches.....	1174
Using tables with GWT.....	1175
Kolban's GWT BPM Utils.....	1176
Visual Task Lists.....	1176
REST wrappers and helpers.....	1177
Examples.....	1178
Showing a visual data table.....	1178
Starting a process from GWT.....	1180
Showing a Coach within GWT.....	1181
BM Business Monitor.....	1182
Installation of Business Monitor.....	1184

Creating a profile.....	1189
Augmenting an existing profile with Business Monitor.....	1207
Information Sources for Business Monitor.....	1214
InfoCenter for Business Monitor.....	1214
Redbooks.....	1215
DeveloperWorks for Business Monitor.....	1215
Introducing the Monitor Model.....	1216
Monitoring Context.....	1219
Key Fields.....	1219
Inbound Event.....	1220
Metric.....	1221
Key.....	1223
Counter.....	1223
Stopwatch.....	1223
Trigger.....	1223
Outbound Event.....	1223
Key Performance Indicators.....	1225
Alerts and Actions.....	1228
Action Service templates.....	1228
Notification Templates.....	1228
Monitor SVG Diagrams.....	1236
SVG Action – Set Color.....	1238
SVG Action – Set Text.....	1239
SVG Action – Hide Shapes.....	1239
SVG Action – Set Diagram Link.....	1239
SVG Action – Send Human Task Notification When Clicked.....	1239
SVG Action – Send Notification When Clicked.....	1239
Generating events from a BPMN process.....	1239
The IBM_BPM_EMITTER_SERVICE application.....	1242
Diagnosing problems.....	1242
BPM side definitions.....	1242
Monitor side definitions.....	1243
Configuring CEI on BPM.....	1243
Configuring for remote Monitor ↔ BPM.....	1243
Event data format.....	1246
Common patterns.....	1248
Examining events issued from BPM.....	1248
Capturing additional data from BPM.....	1249
Capturing the Task ID of a task.....	1250
Capturing the Process ID of a process.....	1250
Capturing the Process Name of a process.....	1250
Compiling Monitor Models.....	1250
Versioning Monitor Models.....	1252
Building Monitor models using the Model Editor.....	1253
Monitor Process Start event.....	1253
Correlating BPMN process events.....	1254
Monitor Tracking point data.....	1254
Closing the Monitoring Context.....	1256
Removing monitor 'Problems' view errors and warnings.....	1256

Debugging Monitor Models.....	1257
Recorded Events Management.....	1257
Using the Monitor Model debugger.....	1258
Integrated Test Client.....	1260
Uninstalling Monitor Apps.....	1262
Using Monitor REST APIs.....	1264
The Monitor REST Event catcher.....	1264
The Monitor REST request.....	1264
The Monitor REST response.....	1264
Sending a Monitor REST request from Java.....	1265
Monitor REST Security.....	1265
REST API Components.....	1266
Getting models.....	1266
Getting monitoring contexts.....	1266
Get Metrics.....	1266
Get instances.....	1267
Getting KPI Contexts.....	1268
Getting KPI Lists.....	1269
Getting KPI Value.....	1269
Creating new KPI definitions.....	1272
Time Filters.....	1277
Get Historical KPI data.....	1278
Get Dashboard Alerts.....	1280
Get Alert Subscriptions.....	1280
Create Alert Definition.....	1280
Useful Monitor/REST JavaScript.....	1283
Building a map of Instance data.....	1283
Monitor Security.....	1283
Advanced Model development.....	1285
Expressions.....	1285
Data functions.....	1285
String functions.....	1286
Math functions.....	1286
Time functions.....	1286
IBM Monitor extensions.....	1287
Authoring User Defined XPath functions.....	1288
Two Stage Modeling.....	1291
Monitor and Cognos.....	1291
Cubes, Categories, Dimensions and Levels.....	1292
Cognos Charting.....	1294
Business Space Widgets for Monitor.....	1296
KPIs.....	1296
Report Viewer Business Space Widget.....	1298
Generated Monitor Models for a Process Application.....	1300
Instance Metrics.....	1302
BPD level.....	1302
<BPD> Steps Level.....	1303
KPIs.....	1304
Working with Changed Variables.....	1304
Sample Monitor Solutions.....	1307

Task Escalation alerts.....	1307
Monitor Notes.....	1308
	1308
WebSphere Application Server.....	1309
Profile Management.....	1309
References – Profile Management.....	1309
Command line profile management.....	1309
Switching a server from production to development mode.....	1309
Recovering from a failed adapter install.....	1310
Recovering from XA recovery.....	1310
WAS Security.....	1310
Virtual Member Manager.....	1311
WebSphere Application Server WEB 2.0 Feature Pack.....	1311
AJAX Proxy.....	1314
JAX-RS – REST Services in Java.....	1315
JAX RS in WebSphere Application Server.....	1319
Deploying Java EE applications to WAS.....	1323
WAS Messaging.....	1324
Foreign Connections.....	1324
	1325
Enterprise Content Management Integration – ECM.....	1326
Content Management Interoperability Services – CMIS.....	1326
Configuring IBM BPM for an external ECM.....	1328
The Document List Coach View.....	1330
Document List – Search service.....	1334
The Document Viewer Coach View.....	1336
Content Integration node.....	1337
Description of a Document (ECMDocument).....	1338
Description of a Search Result (ECM SearchResult).....	1338
Description of an ECMID.....	1339
Description of Document Info (ECMDocumentInfo).....	1339
Description of a folder/document property (ECMProperty).....	1339
Description of a folder (ECMFolder).....	1339
Description of a Content Stream (ECMContentStream).....	1340
Description of an object type definition (ECMObjectTypeDefinition).....	1340
Content Integration node - Add document to folder.....	1342
Content Integration node - Cancel check-out document.....	1342
Content Integration node - Check-in document.....	1342
Content Integration node - Check-out document.....	1342
Content Integration node - Copy document.....	1342
Content Integration node - Create document.....	1342
Content Integration node - Create folder.....	1342
Content Integration node - Delete document.....	1343
Content Integration node - Delete folder.....	1343
Content Integration node - Get all document versions.....	1343
Content Integration node - Get documents in folder.....	1343
Content Integration node - Get document.....	1344
Content Integration node - Get document content.....	1344
Content Integration node - Get folder.....	1344
Content Integration node - Get folder by path.....	1344

Content Integration node - Get folder tree.....	1344
Content Integration node - Get type definition.....	1344
Content Integration node - Get type descendants.....	1344
Content Integration node - Move document.....	1344
Content Integration node - Move folder.....	1344
Content Integration node - Remove document from folder.....	1344
Content Integration node - Search.....	1344
Content Integration node - Set document content.....	1349
Content Integration node - Update document properties.....	1349
Examples of Content Integration nodes.....	1350
Finding documents in a specific folder.....	1350
Inbound ECM events.....	1350
ECM event REST requests.....	1351
Debugging ECM calls.....	1351
CMIS Tools.....	1352
CMIS Workbench.....	1352
API Programming for ECM.....	1353
Uploading a new document.....	1353
ECM Providers.....	1353
IBM BPM Document Store.....	1354
FileNet.....	1354
FileNet Information Sources.....	1354
FileNet Installation.....	1355
Alfresco.....	1374
Alfresco Installation.....	1374
Alfresco debugging.....	1383
.....	1383
WebSphere Operational Decision Management - WODM.....	1384
WODM Installation.....	1384
Creating an instance of a Decision Server.....	1389
Creating an instance of Decision Server Events.....	1389
Augmenting an existing App Server with Decision Server Events.....	1390
WODM Documentation.....	1396
IBM Product Documentation.....	1396
Web Page Information Sources.....	1396
DeveloperWorks.....	1397
Redbooks.....	1397
WODM Architecture.....	1397
WebSphere Decision Center.....	1397
WebSphere Decision Server.....	1397
WODM Terms.....	1398
Rule Designer.....	1398
Project Structure.....	1398
Learning WODM.....	1399
Rule Projects.....	1400
Running a test execution.....	1401
Defining a Rule Execution Server configuration.....	1402
Categories.....	1405
The Vocabulary.....	1406
The Business Term.....	1407

Categories.....	1408
Functions.....	1408
Execution Object Model (XOM).....	1408
Business Object Model (BOM).....	1408
BOM Class category.....	1409
BOM Member category.....	1410
BOM Data types.....	1411
BOM implementation files.....	1412
Working with XSDs.....	1412
Rule Sets.....	1416
Rule Set Parameters.....	1416
Rule Flow.....	1417
Business Policies.....	1420
Business Action Language.....	1420
BAL Definitions.....	1421
BAL Conditions.....	1421
Rule Actions.....	1422
Business Rules.....	1423
Decision Tables.....	1424
Decision Trees.....	1425
WODM – Event Processing.....	1429
Event Processing Architecture.....	1429
IBM Decision Server Events – Knowledge.....	1430
InfoCenter.....	1430
Installing the Event Processing Business Space widgets.....	1430
Creating an Event Project.....	1430
The DES console.....	1431
The Rule Execution Server.....	1431
Rule Execution Server Console.....	1431
Hosted Transparent Decision Services – Web Services.....	1433
Rule Application.....	1433
Deploying a RuleApp.....	1436
RuleApp Properties.....	1440
WODM – Decision Center.....	1440
Configuring Decision Center.....	1440
.....	1444
Business Process Management scenarios.....	1444
Scenario: Exception Handling.....	1445
Advanced IBPM Troubleshooting.....	1445
Data Studio.....	1445
DB2 Troubleshooting.....	1453
Eclipse.....	1456
Unix Notes.....	1457
X-Windows.....	1457
Database Installation.....	1457
MS SQL Server 2008 Installation.....	1457
Java Programming.....	1478
Writing XML documents.....	1478
Working with Apache POI.....	1478
Reading an Excel Spreadsheet with POI.....	1478

Writing an Excel Spreadsheet with POI.....	1478
Demo – Book Order Fulfillment.....	1479
Part I – A simple start.....	1480
Part II – Billing problems.....	1485
Part III – Starting process instances.....	1488
Part IV – Check Stock.....	1491
Part V – Advanced UI.....	1493
Part VI – Business Monitoring.....	1493
POC Considerations.....	1494
Choice of server hosting environment.....	1494
Choice of developer environment.....	1494
Software choices.....	1494
Environment Checklist.....	1494
DB.....	1494
Process Center.....	1495
Kolban's Projects.....	1496
Apache Velocity Integration.....	1496
The Velocity Template Language.....	1496
XSLT DataHandler.....	1498
Converting a BO to a plain XML.....	1498
IBM BPM 7.5 Coach Auto-save.....	1499
Process and Task Data Explorer.....	1499
Out of Office support.....	1501
Tutorials.....	1502
Starting a business process on a schedule.....	1502
Building a Monitor Model for BPMN events.....	1506
Glossary.....	1529
Ideas.....	1529
Areas to write about and work to do.....	1529
Quick Links.....	1531
The End?.....	1531

About this book ...

As an IBMer who's job it is to work with IBM Business Process Management products my style of learning is to learn something and then to write down what I have learned. I do this for two primary reasons. The first is that as I get older, I can't remember information for as long as I would like. The second reason is that I am a **firm** believer in writing down information and sharing it with others in the hope that all can benefit from things learned as much as I benefit from reading the words that others have laid down.

Over the last six years, I had been a heavy contributor to IBM's internal Wikis on IBM BPM products. Unfortunately, despite hundreds of new pages and several hundreds of hours invested in that medium, I was forced to abandon that experiment and decided to write my thoughts in the form of a book.

I've written books before and had them published and swore I would never go down that road again. This time I am approaching it with a different technique. There is no publisher for this book and there *may* never be. Instead, I have chosen to make it available inside IBM and to consumers of the product for benefit and value to all IBMers and consumers.

The content within represents innumerable hours of study and tinkering mostly in the evenings and weekends. It is a loosely related set of articles and notes and little time has yet been spent in polishing the content for clarity and quality ... this means it is pretty darn raw. I also don't claim accuracy or best practices on any of the material found within so use and leverage at your own risk.

When this book was first released back in December 2010, it focused exclusively on the WebSphere Lombardi Edition (WLE) product. With the release of IBM Business Process Manager, every page has (hopefully) been re-read and modified to reflect the change to the new product. However, there will undoubtedly be places where the old material was not updated but this will rectified over time.

With these disclaimers in place, I sure hope you find some value within.

The book will **most definitely** continue to be updated over the time ahead and will be refreshed on a monthly basis as new knowledge is covered, mistakes corrected and the product(s) evolve. A permanent web page was created to host access to the book. The link for this page is:

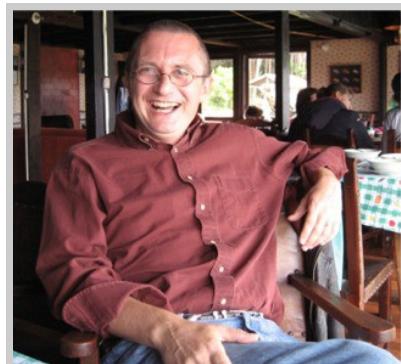
<http://www.neilkolban.com/IBM>

Readers should check back there periodically to validate that they have the latest version available.

Lastly, these notes represent my best understanding of a story or technology, there may be errors or misunderstandings. Beware :-)

Neil Kolban

(kolban@us.ibm.com)



Book Release History

The book is written using Apache Open Office Writer. I had previously used IBM Lotus Symphony 1.3 which I had thought to be awesomely good word processor. In January 2012 IBM announced that there would be no further substantial enhancements to Symphony and all effort would be focused on Apache Open Office. Downloading and working with Open Office, I found that it was (to all intents) identical to Lotus Symphony. Digging further, it seems that Lotus Symphony was itself a branch from an original Open Office. The PDF versions of this book are generated from the in-built PDF generation capabilities found in Open Office.

Screen shots are captured with [Snagit](#) v11 from a software producer called TechSmith.

Custom drawings were built with [Microsoft Photo Draw v2](#) (circa 2000). This is arguably one of the best drawing packages ever created and contains features which I have never subsequently seen in any other drawing package. Unfortunately this product was discontinued by Microsoft a long, long time ago.

Cross references within the book are being added slowly but the desire is for more comprehensive cross referencing as time permits. The cross references are hyper-link-able within the PDF versions.

Below is the history of release and some of the page counts in each release. Some releases have seen the page count fall as old materials are pruned or replaced. Even if the page count falls or maintains the same, each monthly release will be an improvement over the previous one as materials will have been polished or clarified.

Date	Description
02/01/14	Page count 1498.
01/01/14	Page count 1475. Continued to remove redundant information. Removed pre-8.0 BPM Coach information.
12/01/13	Page count 1540. Removed all Dojo programming information into new Worklight book.
11/01/13	Page count 1585
10/01/13	Page count 1606.
09/01/13	Page count 1593.
08/09/13	Page count 1581.
07/01/13	Page count 1550.
06/03/13	Page count 1514.
05/03/13	Page count 1508.
04/01/13	Page count 1504.
03/01/13	Page count 1482.
02/01/13	Page count 1477.
01/02/13	Page count 1482.
12/01/12	Page count 1453.
11/01/12	Page count 1407.
10/01/12	Page count 1392.
09/01/12	Page count 1391.
08/01/12	Page count 1380.
07/01/12	Page count 1331.
06/01/12	Page count 1292. Started migration to 8.0.
05/07/12	Page count 1256.
04/01/12	Page count 1215.
03/01/12	Page count 1163. Bunches more stuff on GWT and a start on JAX-RS thinking.
02/07/12	Page count 1146. Released late.

01/01/12	Page count 1099.
12/01/11	Page count 1033. Started migration to 7.5.1.
11/01/11	Page count ???.
10/01/11	Page count 1006.
09/01/11	Page count 1009.
08/01/11	Page count 992.
07/01/11	Walk through of installs including PMT. Page count 822.
06/03/11	1 st release for IBM BPM v7.5 product. Page count 786.
01/01/11	January 2011 release. Page count: 504. General cleanup; start of rework (where necessary) for 7.2; inclusion of Dojo recipes.
12/03/10	First release of material. Page count: 395.

What is Business Process Management?

Business Process Management or BPM is discussed in great detail in other literature and web sites. The purpose of this book is to describe IBM's Business Process Manager (IBPM) product and not to provide theory on BPM in an abstract sense. This book will cover the usage of IBPM to achieve the goal of BPM.

Overview of IBM Business Process Manager

IBM Business Process Manager (IBPM) provides a platform on which Business Processes can be described, implemented, executed and monitored.

History of IBM Business Process Manager

To truly understand IBM Business Process Manager (IBPM), one should first start with the path that IBM has traveled to get to where they are today. Without an understanding of the market and product evolutions, some aspects of IBPM may seem *odd* or *strange*. Let us start back in 2005 with the release of an IBM product called WebSphere Process Server (WPS). WPS was the preceding Business Process Management product from IBM. It was designed to answer all the needs of a customer from a BPM perspective. Its core design followed a Service Oriented Architecture paradigm. What this meant was that customers would have "business services" and those services could then be aggregated together to build "business solutions". In practical terms, a "business service" would be an application or coarse grained functional component that exposes itself as a reusable service. Commonly it would be exposed as a Web Service but almost any technology to execute such a service could be used. By building out a set of re-usable services, if we could now describe the *rules* that govern the execution of these services, we would have a solution.

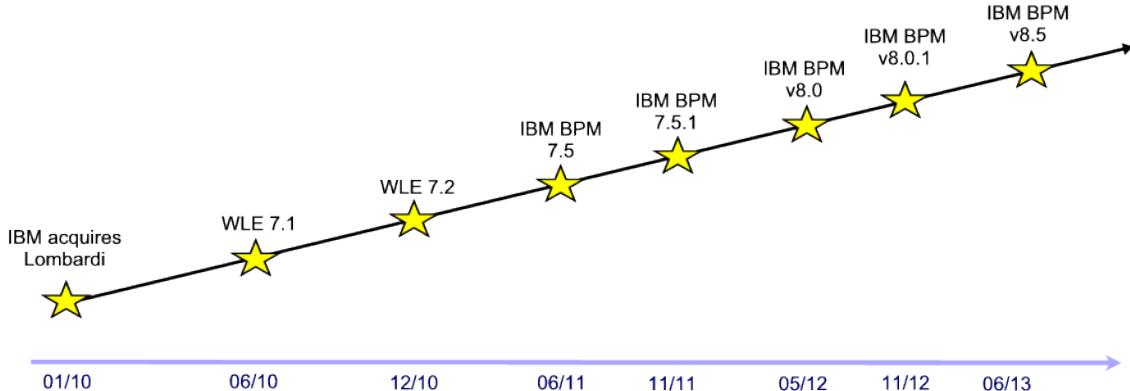
Historically, such rules had been described in application code with the likes of programming languages such as Java or C#. By the time WPS came along, a new player in the story had arrived. This was an open and industry standard called Business Process Execution Language (BPEL). BPEL promised to be the core glue that would solve all Business Choreography needs. Without (yet) going into detail, think of BPEL as a language that described the order in which steps of a process would execute including branching, variable updates, and more. This language could be visualized in a flow chart style diagram. The WPS product provided IBM's implementation of BPEL. Coupling together WPS, its SOA integration capabilities, BPEL and a host of other functions, IBM brought a product to market that appeared to solve the majority of BPM problems.

Unfortunately, despite IBM's best efforts, WPS did not capture 100% of the market. Competitors arrived on the scene and market share became split between IBM and others. What also became apparent was that the expectations of customers was changing. IBM had seen BPM as being extremely closely married to SOA principles and that manifested itself in many aspects of WPS. Competitors were focusing much closer on the Business Users and not as deep on the technology users. WPS was arguably the most powerful SOA engine in the market but its *leanings* were distinctly technical. Business users could not sit down at its development tooling (a product called WebSphere Integration Developer (WID)) and describe/capture their business processes.

To counter this problem, IBM produced additional products such as IBM WebSphere Business Modeler (Modeler) which was aimed at business users. However, modeler, although awesome for describing business processes was distinct from WPS requiring migration work from the model through to WPS for execution. Changes reflected in the process made by technical staff in WID had challenges being brought back into Modeler. Something had to change.

A 3rd party company called Lombardi marketed a product called TeamWorks which was a stiff competitor to IBM's WPS. In 2010, IBM acquired Lombardi and the product previously known as

TeamWorks was re-named as WebSphere Lombardi Edition (WLE) . The first release of this occurred in June 2010 with the 7.1 release. In December 2010, the WLE 7.2 version hit the streets. Although the transition from TeamWorks to WLE was quite transparent, on occasion the original name of the product showed through in a few places. For example, package names and variable names still begin with "tw" (TeamWorks).



WLE continued to be extremely successful for IBM post acquisition and quickly became a staple of IBM's BPM products. However, it didn't take anyone long to realize that IBM had **two** BPM products in the market. These were, of course, WPS and WLE. Since prior to the acquisition, IBM and WLE had been rivals competing against each other for the same consumer projects, something further had to change.

The answer came in June 2011 with the release of the product called IBM Business Process Manager (IBPM). IBPM is a single product that brings together all the features and functions of both WPS and WLE in one environment. It needs to be immediately said that IBPM is **not** the two products known as WPS and WLE dumped into one box ... far from it. IBPM is truly the integration of all the best parts of WPS and the best parts of WLE brought together in a single run-time environment, with identical models of development and deployment.

All this being said, despite the version of the product being labeled 7.5, this is the first release of IBPM to market. The core code built and tested by developers is extremely mature as its heritage are the mature WPS and WLE ancestor products. To maintain backwards compatibility for previous WPS customers and previous WLE customers, IBPM **does** surface many of the concepts that were found in WPS and WLE by themselves. Putting it another way, the nature of its heritage show up on occasion.

IBPM Usage

Business Processes usually exist in the minds and practices of staff a business enterprise. Capturing them for automation in a BPM product is a relatively new endeavor (say within the last ten years). Business Analysts and BPM Analysts could spend extensive amounts of time studying the operation of a business before ever coming close to being able to translate that into automation design or implementation. One of the primary strengths of IBPM is to be able to considerably shorten the duration through a series of interactive *sessions* where the process is directly captured in IBPM as an intuitive diagram. After having captured the "flavor" of the process, the process can be "played back" to parties who understand its nature to validate that what has been captured is correct with respect to how the business operates today or how it should operate in the future. If parts are found to be in error or misunderstood, they can be changed in real-time and the solution played back again. This can be repeated until the process has been correctly captured. This interactive and real-time iterative walk-through capability is significant differentiator for IBPM compared to other

products in the marketplace.

Modeling a process

One of the first parts of building an IBPM solution is to model the Business Process. Modeling the process is the notion of capturing the existing process or the new process that is desired to be created. This task is usually performed by a Business Analyst in conjunction with key individuals involved with the process who may know details about certain areas.

The outcome of this phase of activity is a model of a process.

During this period, we are interested in capturing the logical flow of steps in the process and the coarse grained nature of those steps themselves. What we are **not** interested in capturing (at this time) are the mechanics of how those steps are to be performed ... that is vitally important information, but will come later.

IBPM allows us to capture the model of a process using a powerful drawing technique that is based upon the prevailing Business Process Model and Notation (BPMN) standard. This drawn diagram represents steps in the processes as "boxes" with lines drawn between them to describe the flow of work. Each box represents a step and is labeled with its description in the diagram.

By capturing a diagram of the process, the diagram can be discussed for accuracy with other people. It can be used as the basis of the important question "Is this how we work today?". During these discussions, it is common to catch errors or misconceptions that were made while the diagram was being originally drawn. It is a trivial matter to change the diagram at this stage as needed.

In other BPM products, the step of modeling the business process also occurs. Where IBPM differs is that the model captured here is **exactly** the model that will be refined in the future for implementation. There is **no** concept of exporting the model from one tool and importing into another. That is simply not needed. Because the model is *shared* between design and implementation, it is simply not possible for the two to diverge from each other. This eliminates the "round-trip" problems seen in other environments.

Integrating with IT Systems

Although many business processes involve only routing work from one user to another, the vast majority involve interacting with existing IT applications that may be located on a variety of remote systems. IBPM provides the ability to include these IT systems as part of the process. Integration can be achieved through Web Services or other communication calls.

See also:

- Business Process Definition - BPDs

Architecture

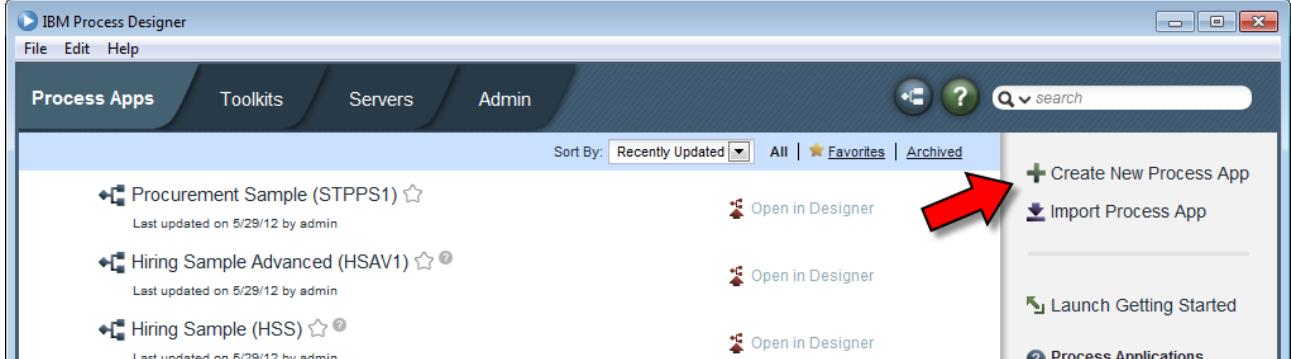
IBPM is composed of a number of functional components. These components and how they interact with each other constitute the architecture of IBPM.

Process Applications

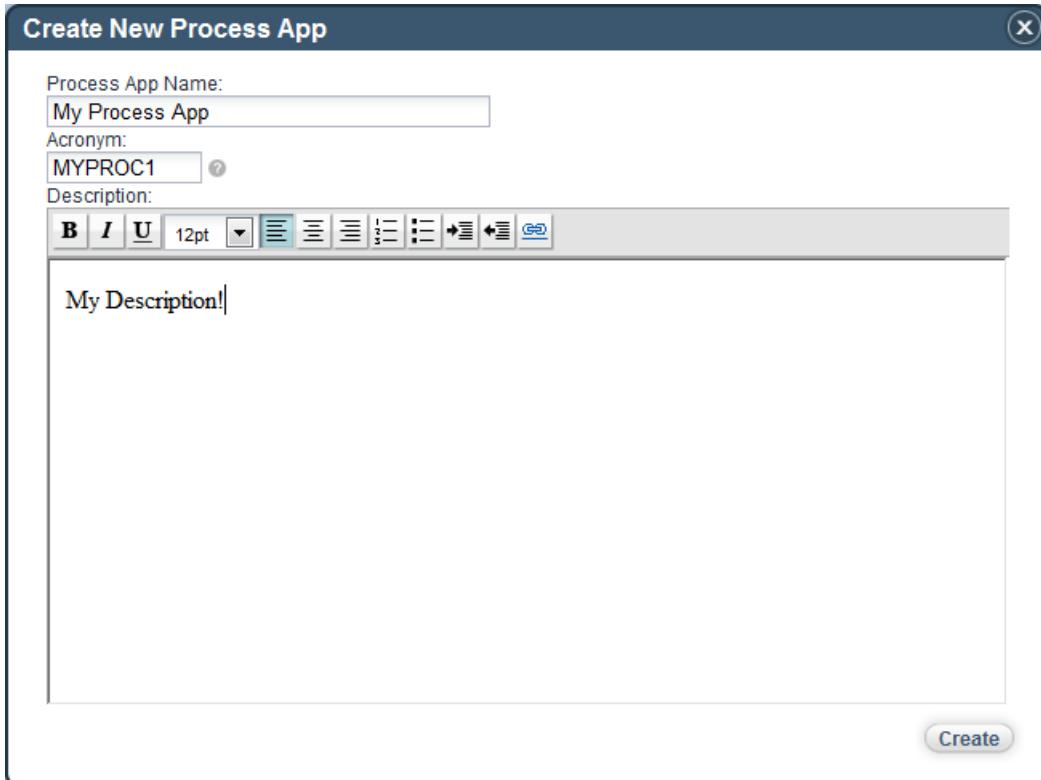
A Process Application is the container for a solution. You can loosely think of it as a project. The Process Application is initially created through the Process Center console. It is given a name and a tag called an *acronym*. The acronym must be unique and can be no more than seven characters in length.

Once the Process Application container has been created, artifacts can then be further created within it using the IBPM Process Designer (PD) tooling. The Process Application and its artifact contents are stored within a repository hosted and managed by a component called the Process Center.

Process Applications can be created from the Process Center console either from its web page interface or from within IBPM PD. The main Process Apps page has a button to create a new Process Application.



The creation of a new Process Application will open a dialog and prompt for a name for the new application as well as its acronym value.



The "state" of a Process Application at a given point in time can be recorded in a "snapshot". This state consists of all the artifacts and their content at the time that the snapshot was taken. As changes are made to the Process Application after the snapshot was made those changes are not reflected in the snapshots that occurred in the past. Additional snapshots can be taken at anytime.

Process Applications commonly contain one or more Business Process Definitions (BPDs). These can be thought of as models of the process that will eventually be executable. In this book I switch between the phrases BPDs and BPMN processes as I see these as synonyms for each other.

See also:

- Business Process Definition - BPDs
- Snapshots

Process Instances

A BPD in a Process Application reflects a *template* of a process as opposed to an *instance* of a running process. When a process is started, a new instance of the process is created from the template. An actual process instance can be thought of having a current state and can only be in one state at a given point in time. The potential states are:

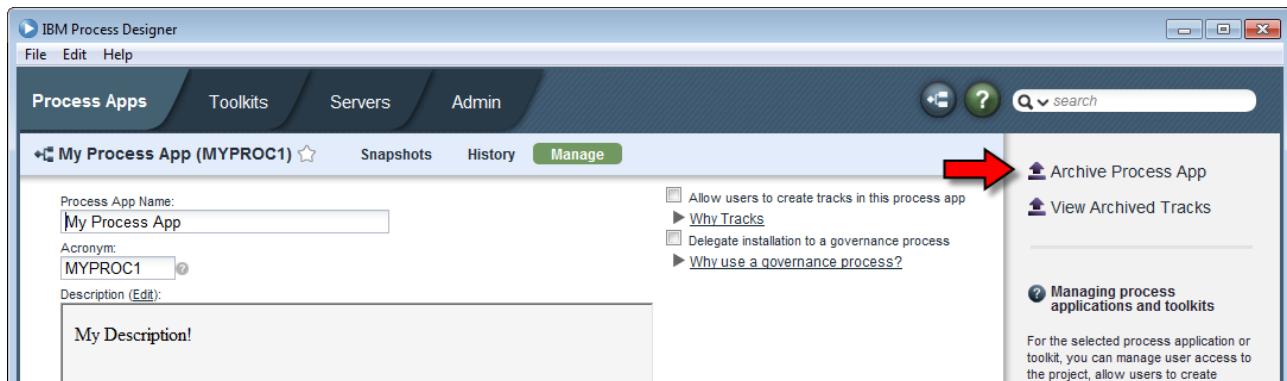
- Active – The process instance is active (running).
- Completed – The process instance has completed.
- DidNotStart – The process instance did not start.
- Failed – The process instance has failed.
- Suspended – The process instance has been suspended and can be resumed. (See Error: Reference source not found).
- Terminated – The process instance was explicitly terminated prior to completion. (See Error: Reference source not found).

In addition to the processes execution state, there is also the concept of the state of any variables that may have been created.

When a process instance is created, it is given a unique integer process id value. This value is unique amongst all the process instances within the environment. Some customers have used this value as a key in 3rd party databases and have had problems when migrating processes to a new version of the product where the instance ids are reset. They remain unique in the environment just not globally unique across time.

Archiving Process Applications

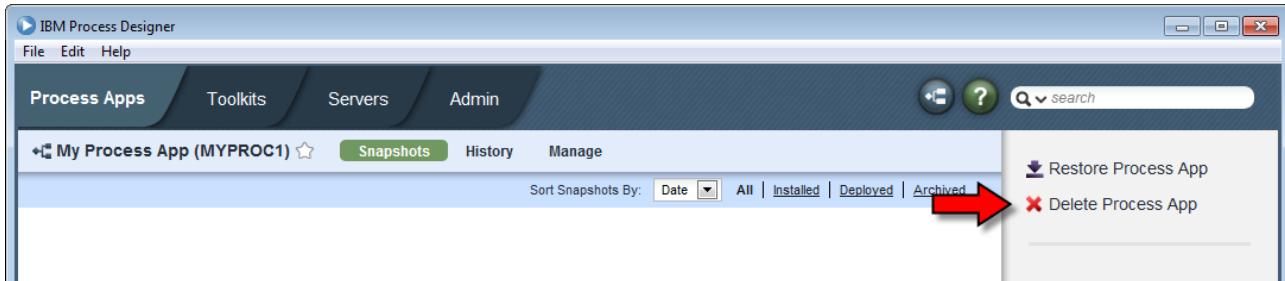
Once created in PD, the definition of a Process Application is *usually* never deleted. This may seem odd at first but if we think about it, the only resource being consumed by a non-deployed Process Application is some disk space within the back-end Repository database which is usually relatively small. In addition, Process Applications are extremely coarse grained concepts and there should be no obvious need to create too many of these. A Process Application can be *hidden* from overt viewing by flagging it as *archived*. Archiving a process application removes it from view from the list.



Even though a Process Application has been archived, it still remains within the Repository. A filter on the Process Center console can be used to view archived applications and restore them to

their visible state.

When a Process App has been archived, it is *eligible* for deletion. Deleting a Process App is not a recoverable command. Once deleted, it is gone forever. In versions of the product prior to 7.5.1, this capability was simply not present and Process Apps and Toolkits remained forever.



See also:

- Video Tutorial: [Deleting a Process App or Toolkit](#) - 2012-03-01

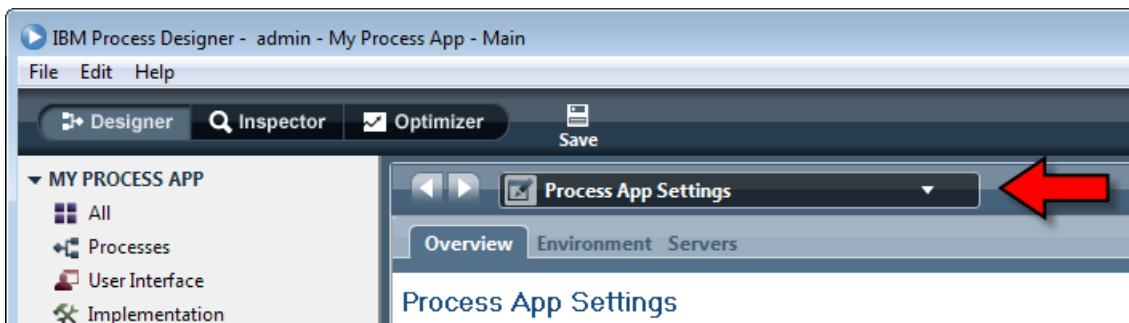
Process Applications state management

A Process Application can be manipulated or worked with as follows:

- Cloned – The process application is duplicated with the new one being given a new name
- Archived – Archiving an application simply means "hiding" it from normal view
- Import – Bring in a previously exported Process Application from a .twx file (Note: The file suffix TWX used to mean "Team Works Export" when the product was historically called TeamWorks).
- Export – Export a Process Application to a .twx file. A .twx file is a file format used to store the content of a Process Application in a computer file. The .twx file can be transferred to other systems running Process Center for subsequent importation.

Changing Process Application settings

Some core settings of a Process Application can be changed within the Process Designer. From the primary menu pull down, an entry for Process App Settings" can be chosen:



After selecting this, an editor is shown in which some key settings can be changed.

These include the name of the Process Application, its Acronym and the textual description. Note that the Process App Acronym can be changed here if needed.

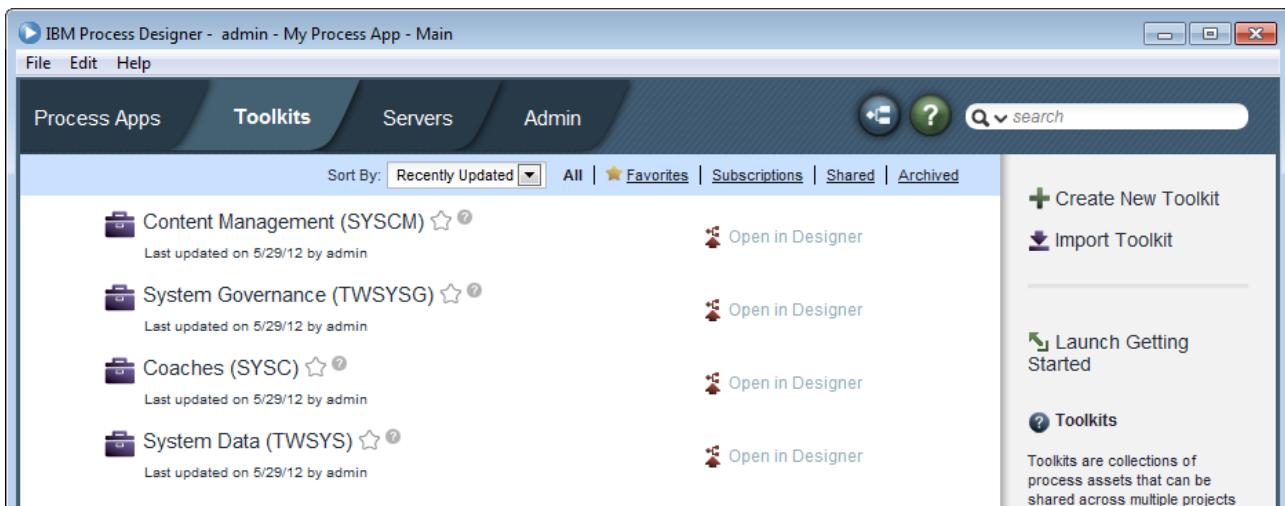
Toolkits

Similar to Process Applications, a Toolkit can also be thought of as a container for artifacts used in solutions. Unlike a Process Application, a Toolkit does not result in a deployable application. Instead, the contents of the Toolkit can be "included" or "used" by one or more Process Applications.

When Process Center is installed and configured, an IBPM supplied Toolkit called "System Data" is automatically imported into the repository. See System Data Toolkit.

This toolkit is marked as read-only and is implicitly a dependent on all other Process Applications and Toolkits. It is the System Data toolkit that contains the core definitions for data structures and other items common across all Process Applications.

Toolkits have their own tabs in the Process Center consoles. From there new Toolkits can be created, Toolkits exported and otherwise managed in a similar fashion to those of the Process Applications.



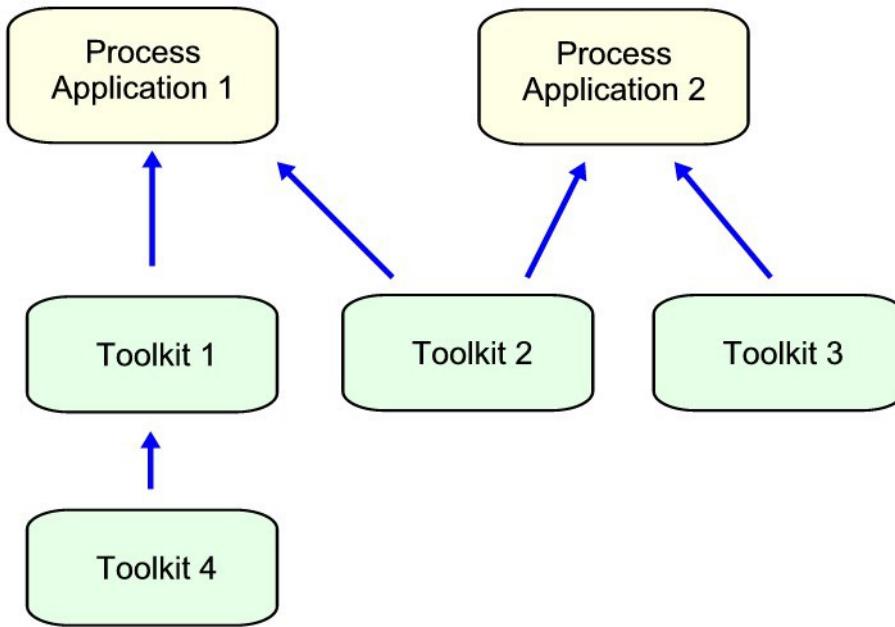
Just like Process Applications, Toolkits can have Snapshots taken off them allowing all the artifacts in a Toolkit to be considered a specific version.

To add a Toolkit as a dependency to a Process Application, the Toolkit must first have a snapshot associated with it. This is because the dependency added to the Process Application is **not** just the Toolkit's name but is instead a specific snapshot of that Toolkit. Once a snapshot of the Toolkit has been taken, a dependency can be added in the Designer view of PD by clicking the **+** icon to the side of the Toolkits entry. A list of potential Toolkits and their associated snapshots is shown for toolkit selection.

Smart filtering is available in this list but is interestingly keyed off snapshot names and not toolkit names.

The following diagram summarizes the story of process apps and their relationship to toolkits. A Process Application can have a dependency (shown by the arrow) on a Toolkit. Multiple Process Applications can have dependencies on the same Toolkit and Toolkits can have dependencies on

each other.



The nesting of toolkits opens up some interesting semantic questions. For example, if Toolkit A has a dependency on Toolkit B then if Process Application C depends on Toolkit A does that also mean that the artifacts defined in Toolkit B are visible to Process Application C? The answer appears to be **no**. If Process Application C wanted to utilize artifacts in Toolkit B it would have to have a dependency on that toolkit to enable access those artifacts directly.

See also:

- System Data Toolkit
- Additional Toolkits

Tasks

A task is a piece of work to be performed by a human being. Typically, this is achieved by the creation of an activity in the BPD which is then associated with a Human Service. When a process reaches a Task, that branch of the process pauses or suspends until the task has been completed. A task has a state associated with it. A task can only be in only one potential state at any given point in time. The states associated with a task are:

- New
- Received
- Replied
- Forwarded
- Sent
- Actioned
- Closed

- Special
- Deleted

Task Priority

Associated with a task is the concept of a priority which indicates how important this task is relative to other available tasks. The Task Priority could be used by user interfaces to determine a display or sort order for tasks and show the tasks with the highest importance first. The choices available for the task priority are:

- Highest
- High
- Normal
- Low
- Lowest

Within a Human Service, the current task can be found from the variable:

```
tw.system.currentTask
```

This object contains a field called `priority` which is the priority of this task. Changing this value results in the task's priority being changed.

See also:

- Data Type – TWTTask

IBPM Components

At a high level, IBPM is comprised of a number of coarse grained *components*. Taken together, these are the IBPM product. Each component serves a unique and distinct purpose and are employed at different stages in the development or operation of an IBPM solution. Breaking IBPM down into these constituent components both aids in the understanding of the product as well as providing a practical differentiation between phases and pieces of operation.

Component – Process Server

Process Servers are the components/engines which run the business processes described by BPDs. During development, the processes run on an instance of a Process Server located at the Process Center. When it comes time to put the applications that have been built in the test and production environments, they will be installed on other Process Server instances.

A Process Server is implemented by a WebSphere Application Server (WAS) with the IBPM product integrated within it. The IBPM run-time consists of IBM written Java product code engineered to conform to and utilize the Java EE framework.

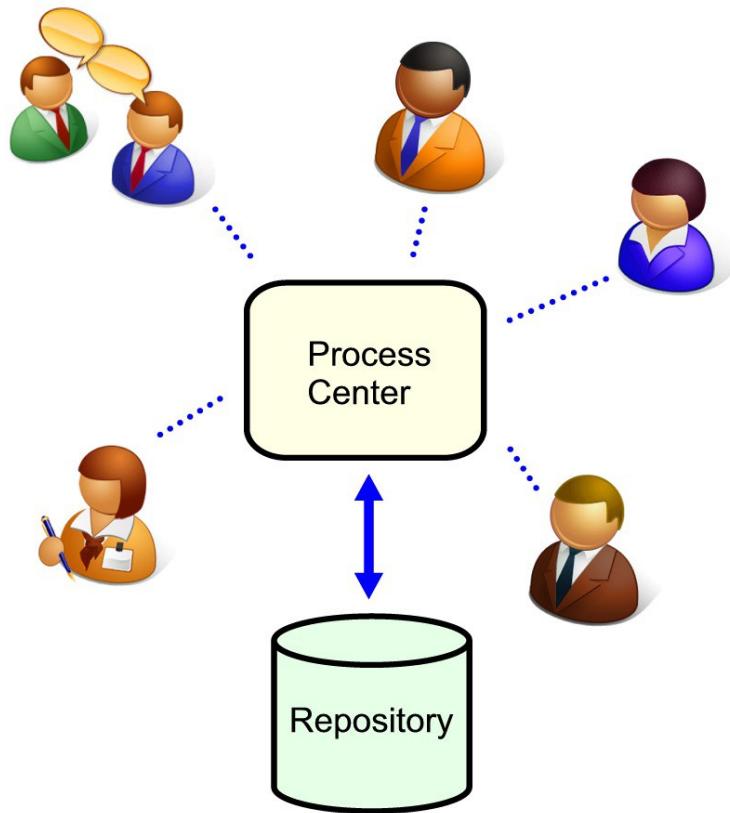
For users of IBPM Advanced, the Process Server also contains a large set of additional functions including Service Component Architecture (SCA), BPEL processes and mediations (to name but a few).

Component – Process Center

One of the core concepts of IBPM is that there is only ever one definition of the solution you are building. This may sound obvious but comparing IBPM to other products, we find that those other products have different "representations" and "copies" of the solution being built depending on what is being done. For example, some products store modeling data in one format/location, process development data in a different format/location and monitoring data in yet another format/location. The result of this is that a plethora of different data structures and sources exist with little interoperability between them. A change in a model, for example, may need to be manually reflected as a change in the implementation of the process. Because the different tools and products don't use the same underlying data and conversions from one product to another must happen, mistakes and misinterpretations can easily occur. The result can be a complex mess.

IBPM on the other hand utilizes a concept that is called the "shared model". In simple terms this means that no matter what is being done within the overall solution, there is only one common repository and a single representation of that solution. Because of this, it is impossible to get two phases of the same solution out of sync with each other.

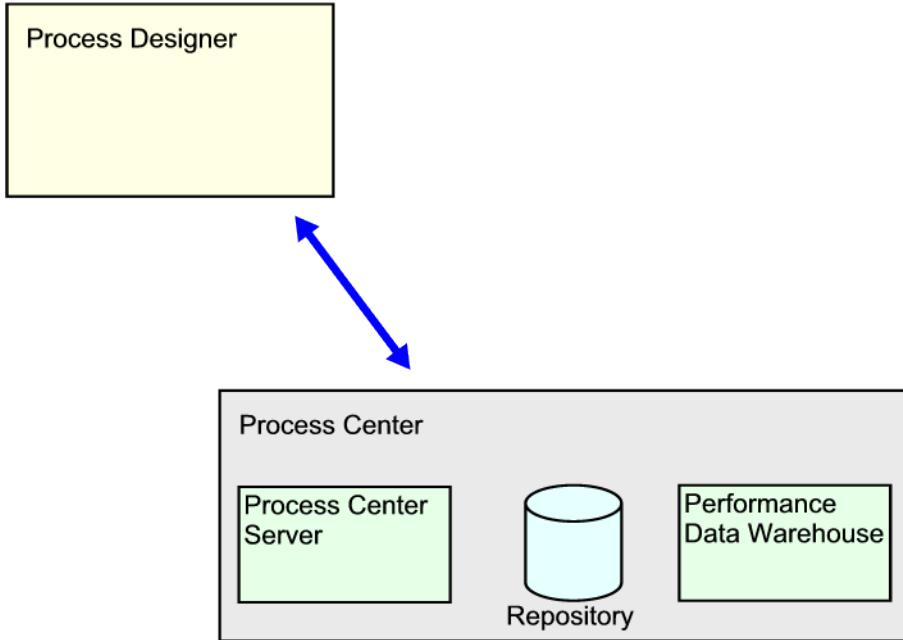
This shared model is realized by an IBPM component called the "Process Center". Part of the process center is a data repository which is simply called the "Repository". Within the repository, there exists the representation of the solution. The IBPM PD tooling connects as a client to the IBPM Process Center to obtain copies of the solution for working upon. When a user makes a change and saves those changes, the results are written back to the repository.



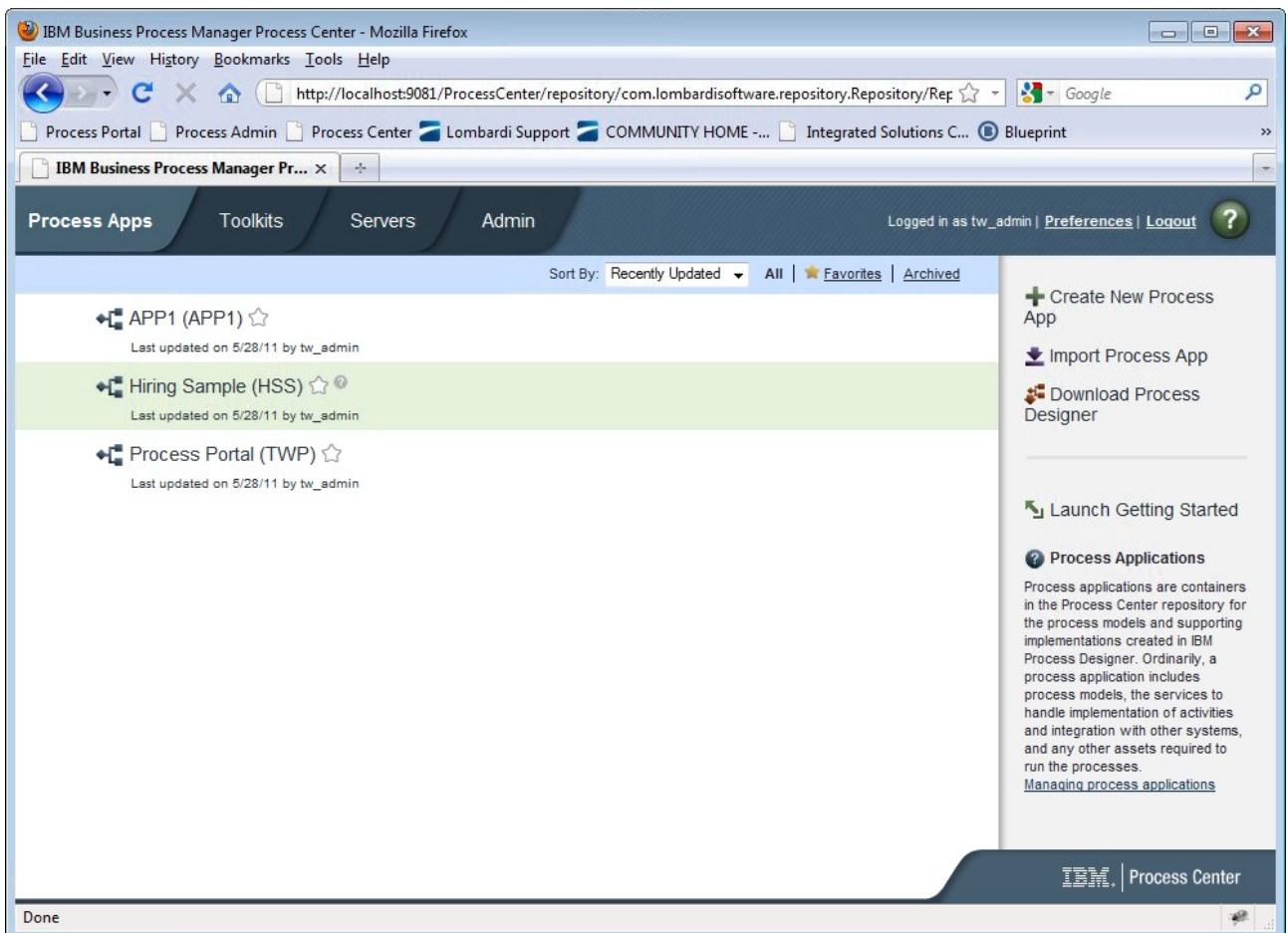
Another way of saying this is that the IBPM PD does **not** maintain the artifacts of a solution on the user's workstation. Instead, they are retrieved from Process Center to IBM PD for editing and when the edit has completed, the changes are saved back to Process Center. Contrast this with other products where when a developer makes a change, the change is made locally on the user's workstation and no-one else sees that change. In order for others to see it, copies of the artifacts are passed around resulting in potential inconsistencies.

The Process Center repository is implemented as tables within a database (commonly DB2). The content of these tables are opaque and access to the repository information is achieved through the tools and web pages. One should never attempt to access these tables directly through database tools.

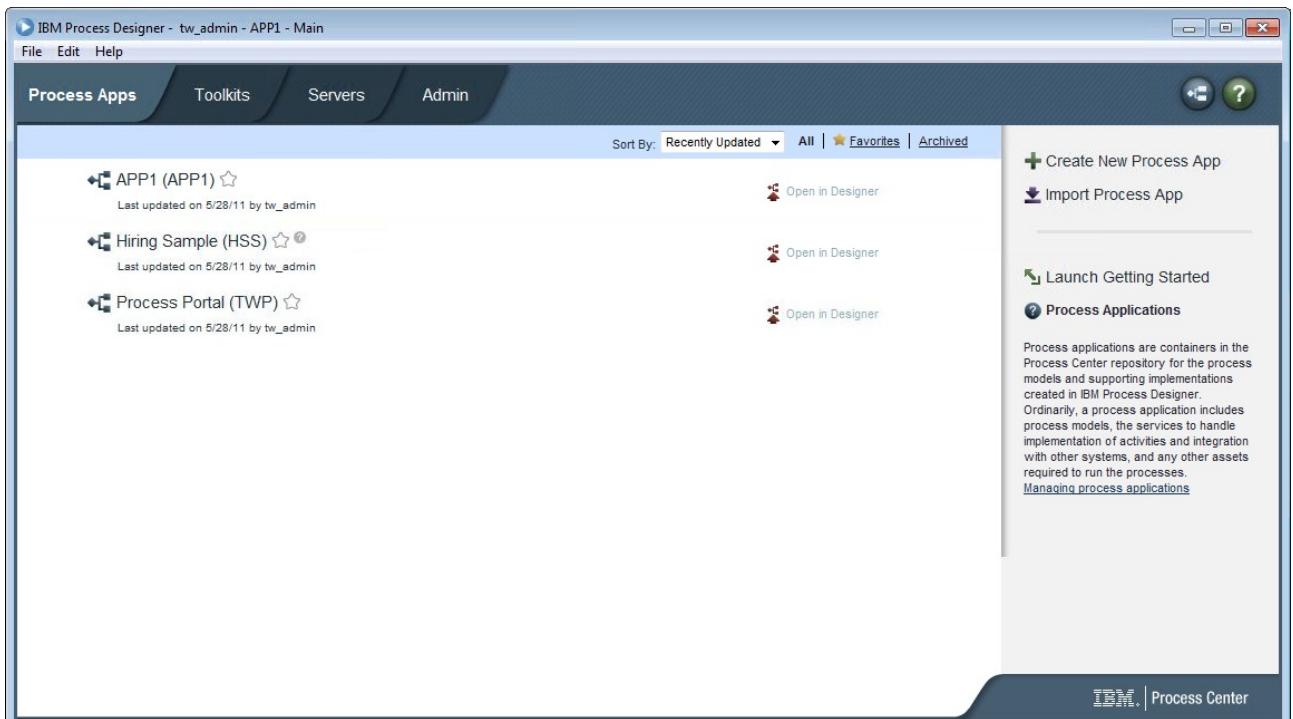
The Process Center is actually comprised of three components. It has the Process Center repository which is responsible for managing the solution's artifacts and it has an instance of a Process Server and a Performance Data Warehouse both used for unit testing.



The Process Center can be accessed either through IBM PD or through a web based interface. Here is a screen shot of the web access:



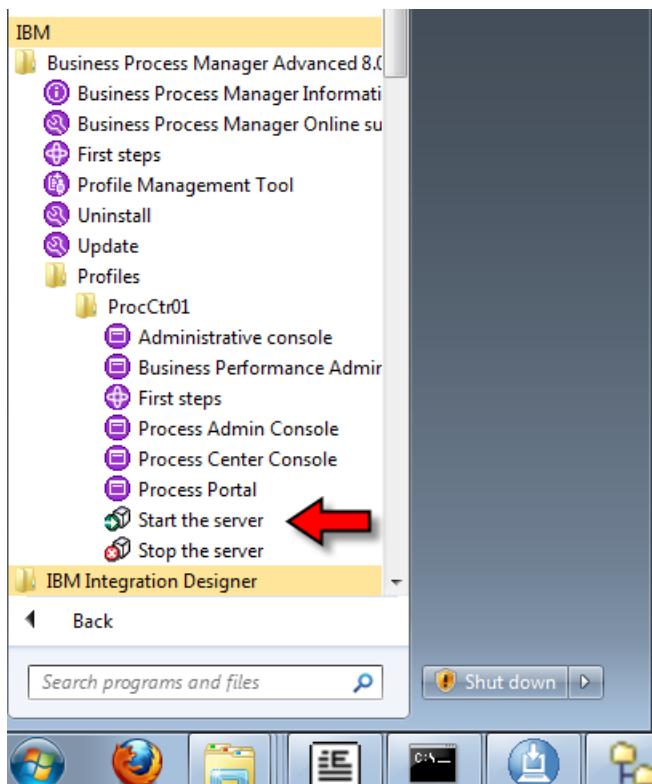
and here is the same interface in the IBPM PD desktop client tooling:



As can be seen, they are virtually identical from an appearance stand point providing a consistent view of the models.

Starting Process Center

The Process Center component runs within a WebSphere Application Server (WAS) server so starting Process Center is actually the task of starting a WAS server which has been configured to host Process Center. During install of IBPM, a WAS server that runs Process Center is registered with the Windows Services systems. An icon is added to the start menus to launch Process Center/WAS:



The Process Center server starts up quietly. A recommendation is to use a tail tool that can be applied to the WAS console log to follow the start-up information until start-up has completed. The console file for this log file can be found at:

```
<IBPM Install>/profiles/ProcCtr01/logs/server1/SystemOut.log
```

This is the default WAS log. The location where the log file is written can be changed through the WAS admin console but it is recommended to leave it as the default unless there is a compelling reason to change.

Component – Process Designer

The IBM Business Process Manager Process Designer (PD) is the development time tooling used to design, model and build processes. It is implemented under the covers using the Open Source technology called Eclipse but has a very non-Eclipse like skin applied over it. Unless you knew otherwise, you would never know that it is Eclipse based. This is both good and bad. It is good in that it builds upon the proven robustness and maturity of Eclipse and leverages a whole host of trusted functions under the covers. The down side is that because it doesn't "feel" like Eclipse, it loses one of the major strengths of an Eclipse hosted application which is the consistency and familiarity of the framework. The decision to *hide* the Eclipse nature of PD was extremely deliberate. It was felt that the visual complexity of Eclipse was too overwhelming for business users. As such, even though it was just mentioned that under the covers PD is Eclipse based, you should now put that out of your mind as it will serve no further purpose.

IBPM PD can be downloaded from the IBPM Process Center and installed on a user's Windows based workstation. It is delivered as a 270 MByte (plus) ZIP file. Once extracted, the program called "eclipse.exe" can be executed to start the tool. Note that IBM charges a per-seat license for installing the Process Designer so just because one *can* simply click a button and install it on a workstation doesn't mean that you are licensed to do so. Discuss this with your IBM sales representative if you have any questions.

IBPM PD connects to a single instance of a Process Center and there is no exposed ability to change which Process Center instance against which it should communicate. This is by design as the intent is that there only ever be one Process Center in any given environment.

At a high level IBPM PD allows us to describe business processes using the BPMN notation. Processes are "drawn" visually on the screen canvas and the technical skills needed to achieve this task are as low as can possibly be made.

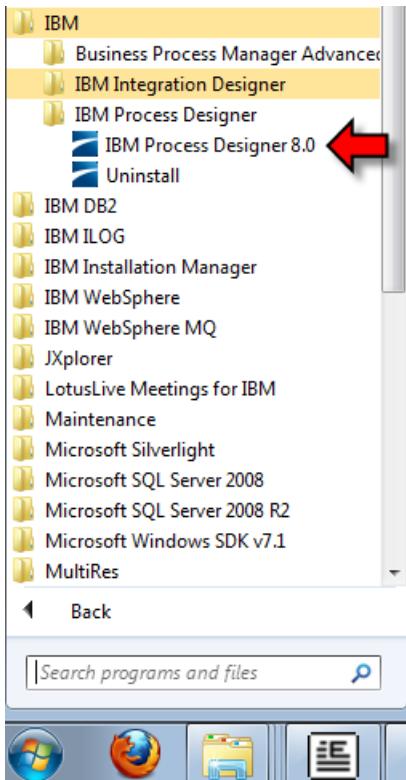
When drawing items on the canvas, holding down the keyboard CTRL key in combination with cursor arrow keys will move any selected elements pixel by pixel. This provides a great way to align the components into a pleasing visual layout.

Play back sessions

One of the key strengths of IBPM is the ability to incrementally build out and demonstrate the business processes being constructed. Rather than having to complete a phase (such as modeling) before seeing how it "feels" during further development, a concept called a "play back" can almost immediately be applied. A play back is the real-time execution of the process without having to explicitly compile and deploy a solution. Think of a play back as the ability to quickly build a "skeleton" of a business process and run it to see what it looks like. The real-time nature of change and play back allows us to enter a step, test it, realize that something is missing, change the process and re-test all without missing a beat or having to "flip" from one development tool or environment to another.

Starting IBPM Process Designer

After installing IBPM, the IBPM Process Designer can be launched from the Windows start menu. Before starting IBPM PD, ensure that the Process Center that it will connect to has been previously started. Remember that IBPM PD is basically a client to the Process Center server. If Process Center is not running, IBPM PD will not function.



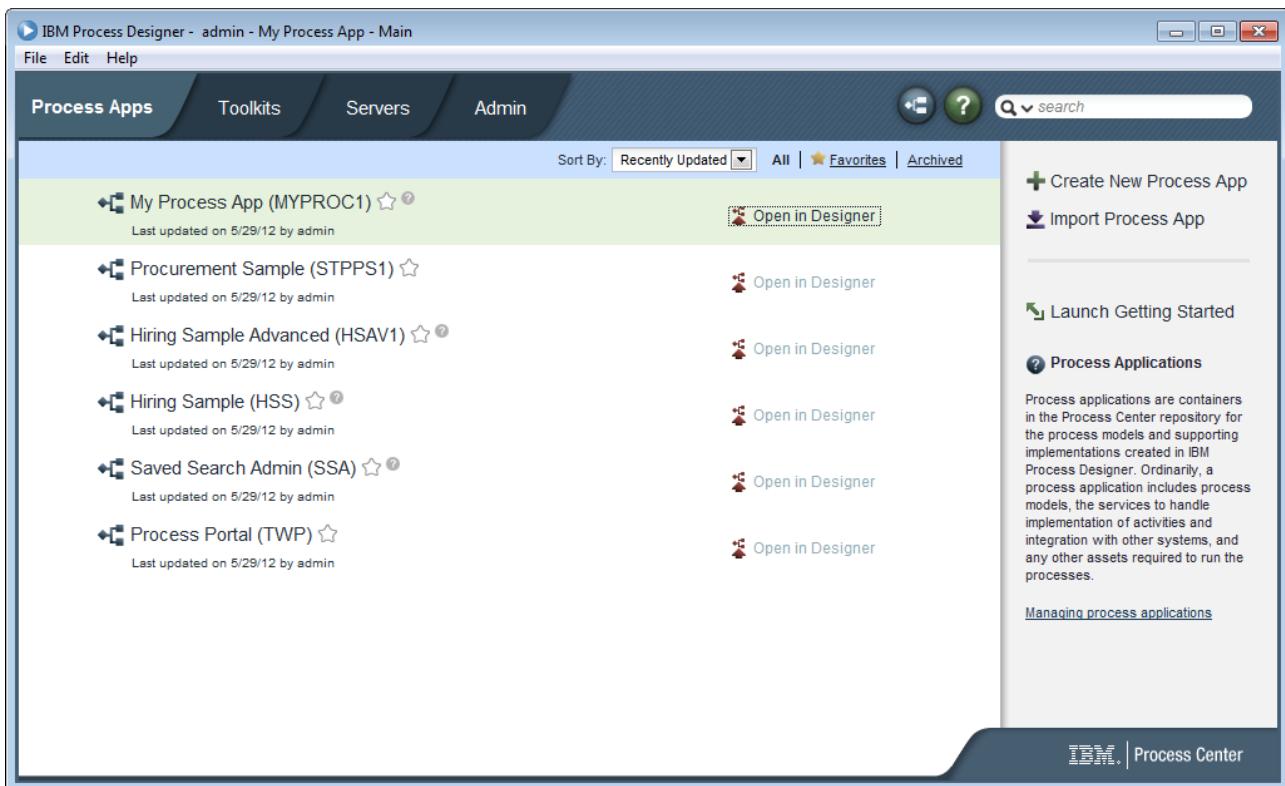
This is actually just a shortcut to Eclipse installed by IBPM. This is usually found at:

<Install>/ProcessDesigner/v8.0/eclipse.exe

Once launched, the IBPM PD will ask for authentication information with a userid/password pair. The default/initial administrator userid and password are commonly:

admin/admin

IBPM PD will then connect to the Process Center to retrieve the information it should display and start showing the appropriate application windows:



Note that the currently logged in user is shown in the title of the window. To be able to login to the IBPM PD, the user must be authorized to access the Process Center repository. See: Securing Access to the repository.

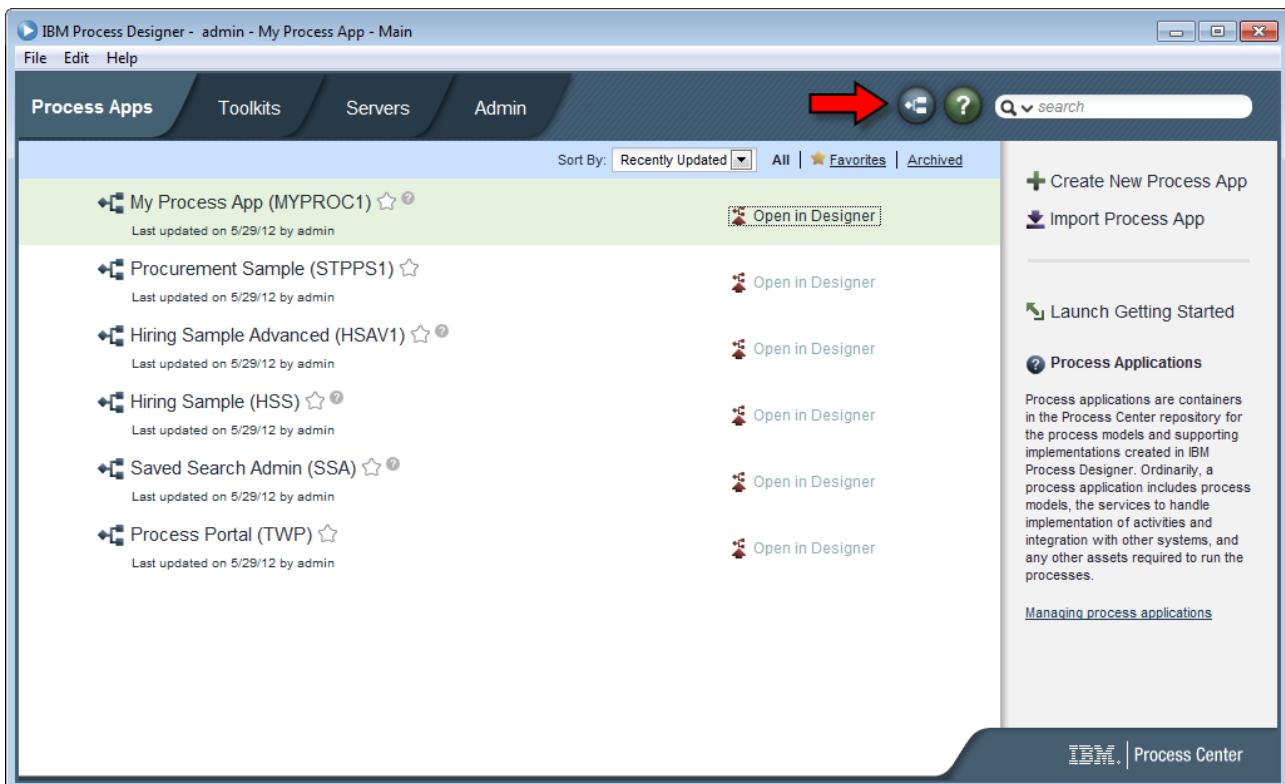
As mentioned, although IBPM PD is actually Eclipse, it looks nothing like a standard Eclipse environment. Gone are the classic Eclipse concepts of perspectives, views and editors.

In the IBPM PD interface there are the following major "views" only one of which is shown at a time:

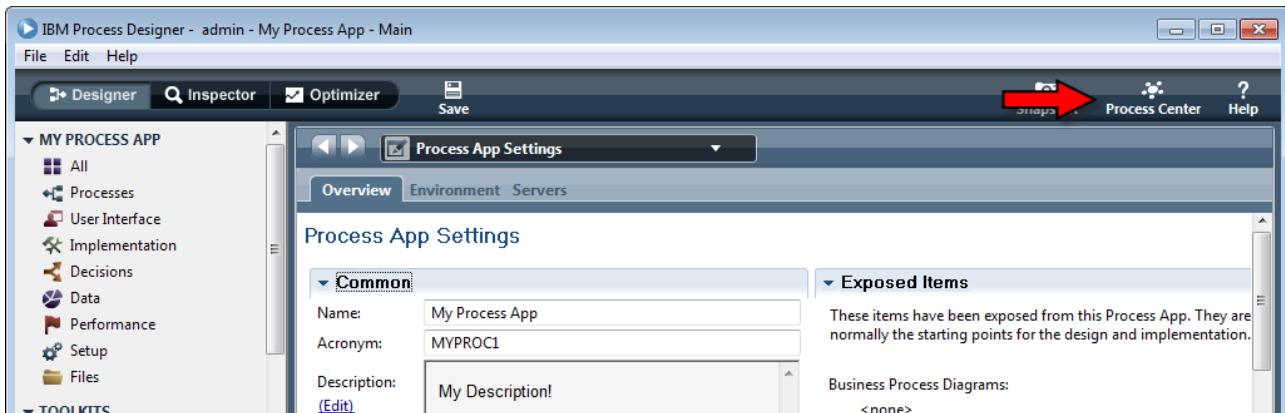
- Process Apps – Selection of available/existing Process Apps
- Toolkits – Selection of available/existing Toolkits
- Servers – List of the Process Server instances
- Admin – Administration tasks
- Designer – Design/construction of a solution
- Inspector – Debugging/monitoring a solution (usually in test)
- Optimizer – Examining performance characteristics

The screen shot previously shown shows the Process Apps view. Access to the views for Toolkits, Servers and Admin is obtained by clicking on the appropriate tab at the top of the screen. Access to the other views is a little more subtle.

An application can be opened in the Designer view by selecting the link to the right of the application name. Alternatively, there is a button that will take you to the Designer view:



From the Designer view, access to the Designer, Inspector and Optimize view can again be easily navigated in a single click using the names at the top of the window.



A button called "Process Center" will return us to the previous view (Process Apps).

At the bottom right of the designer view there is a status icon that shows two things. Firstly, it shows the network host name of the machine running Process Center. This is the instance of Process Center to which IBPM PD is connected. Secondly, it shows the status of the connection between the PD client and the Process Center server. A green icon means the connection is good while a red icon means that the server is probably down. The icon may also be colored yellow or orange to show that connections are possible but slow.

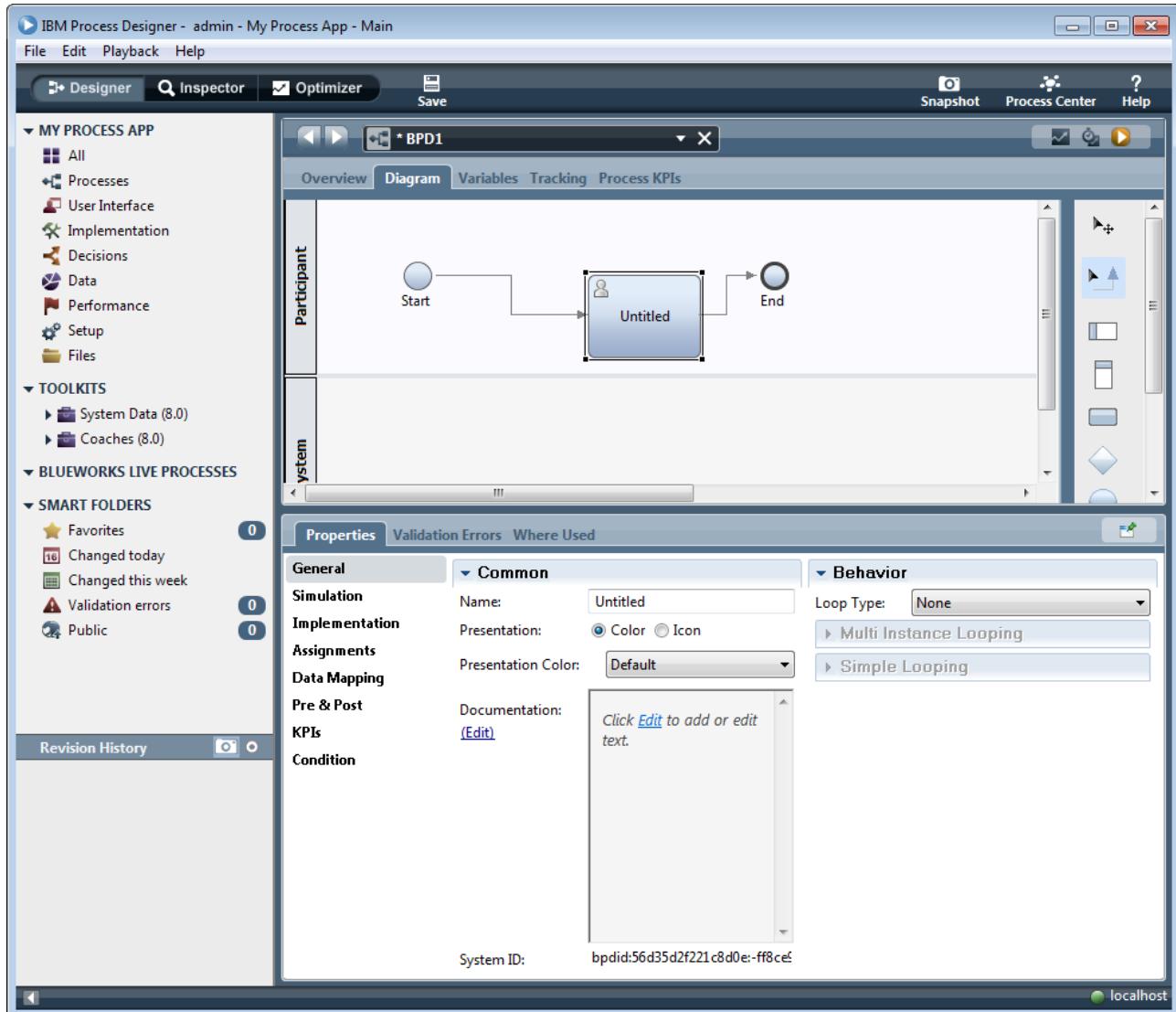
Although not formally exposed or supported, the host name and port number to which PD expects to find the Process Center is contained in the file called `eclipse.ini`. Look for an entry called:

```
-Djava.naming.provider.url=corbaname:iiop:<hostname>:<port>
```

This is where Process Center lives. Formally, one should download a copy of PD from the Process Center to which the PD is to connect even if this means that you have multiple copies of PD on your disk.

Working in the Designer view

The majority of the time spent in IBPM PD will be spent in the Designer view. This is where the bulk of the description of the process is performed. The Designer view is shown in the following screen shot:



The Designer view contains a number of screen "areas" that change based upon what task is being performed. In summary, the view is built from four major areas.

In the top left we have a list of all the artifacts of the solution. In the top right, we have an editor for the current artifact being worked upon. In the bottom left we have a history of the changes and snap-shots available to us. In the bottom right we have some tab sheets the most important of which is the one called Properties.

Tip:

The two left windows can be hidden or shown by using the icon in the bottom left of the window:

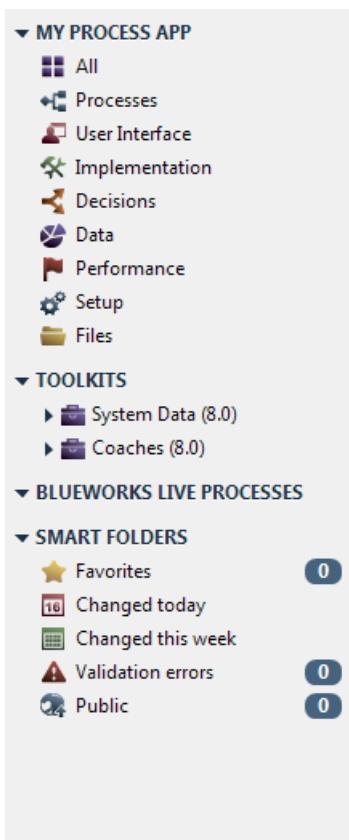


This is a toggle-able button. One click hides and a second click shows. Use this button to give your environment more visual space if the resolution of the screen is low or you wish to see more of your project while editing.

Working with the Library

Within IBPM, you may be working with a large number and a wide variety of artifacts. Trying to manage these artifacts in a way that you can easily locate them can be a challenge. Classic user interface designs have provided folders in a tree structured diagram. IBPM has provided an alternative to this model. Although definitely non-standard it has managed to achieve a high degree of ease of use.

When working in the Designer mode, you can see the area called the "Library". The library is the list/catalog of all artifacts that you are working with or have visibility to.



The major (top level) entries are:

- The current Process Application
- Toolkits
- Bluedworks live defined processes

- Smart Folders

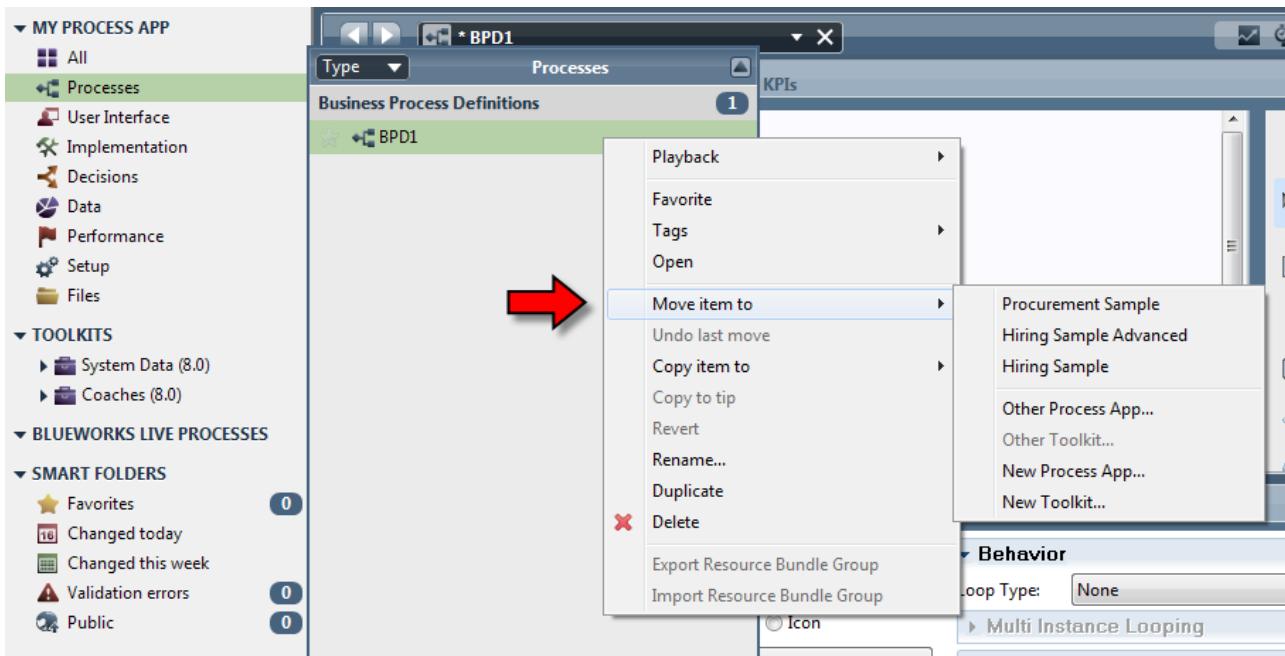
The Library contains high level categories and within each category are the artifacts associated with those categories.

Here is a break-down of the items in the library:

	Processes category
	Business Process Definition – see Business Process Definition - BPDs
	Participant Group
	Historical Analysis Scenario
	Simulation Analysis Scenario
	User Interface category
	Human Service
	Coach View
	Ajax Service
	Localization Resource
	Implementation category
	Integration Service
	General System Service
	External Implementation
	IBM Case Manager Integration Service
	Undercover Agent
	Web Service
	Advanced Integration Service
	Event Subscription
	Decisions category
	Decision Service
	Service Level Agreement
	Data category
	Business Object
	Exposed Process Value
	User Attribute Definition
	Performance category
	Tracking Group
	Timing Interval
	Key Performance Indicator
	Setup category

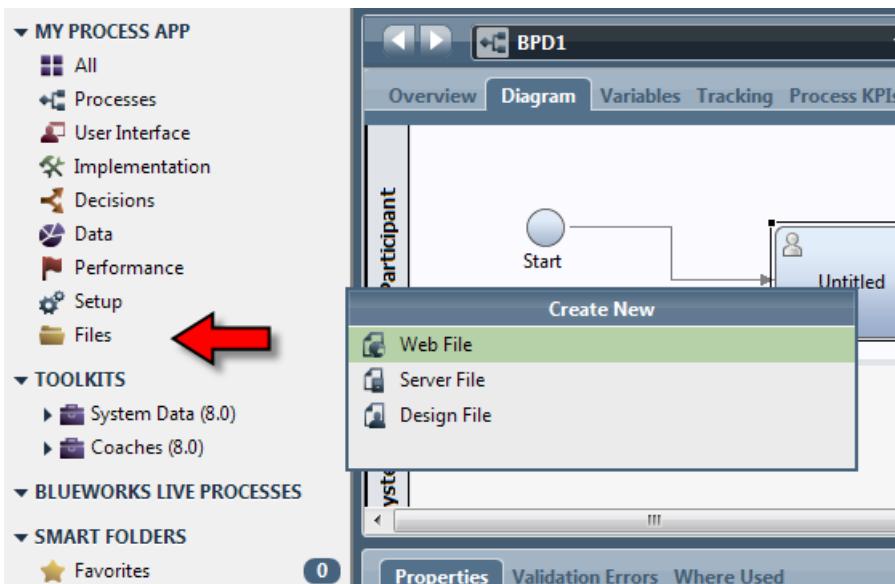
	Localization Resource
	Files category
	Web File
	Server File
	Design File

Items defined in the library are local to just that Process Application. To move or copy items from one Process Application to another, the context menu of the item to be moved or copied can be used. It contains menu entries for both Move and Copy. The target is a different Process App or Toolkit.



Adding managed files

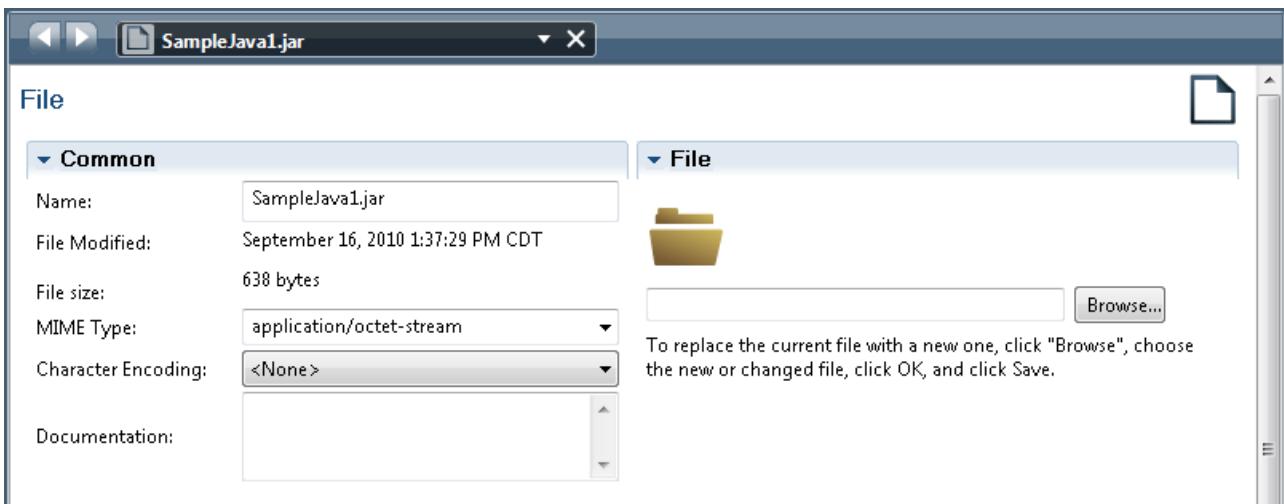
On occasion, files that were originated outside of IBPM PD may need to be included in the solution. IBPM PD provides the ability to include such files for packaging within a Process Application. Examples of files you may wish to package include image files, HTML files and JAR files.



There are three “types” of files that can be defined as server managed ... these are:

- Web File – Web based artifacts such as CSS style sheets, images and other web loaded assets
- Server File – Server files such as Java JAR files and Java Script files
- Design File – Product specific files such as XSLT style sheets used in Coach transformation

After a file has been added, its properties are shown:



These files are deployed with the Process Application. The files can be accessed from JavaScript code. The following fragment retrieves a JavaScript TWManagedFile object:

```
tw.system.model.findManagedFileByPath("<file name>", TWManagedFile.Types.Web);
```

The TWManagedFile has a number of properties including one called "url". This property contains the web URL that can be used to access the file over a network.

If the managed file is a ZIP, appending the name of the file in the ZIP to the URL can be used to retrieve the specific file.

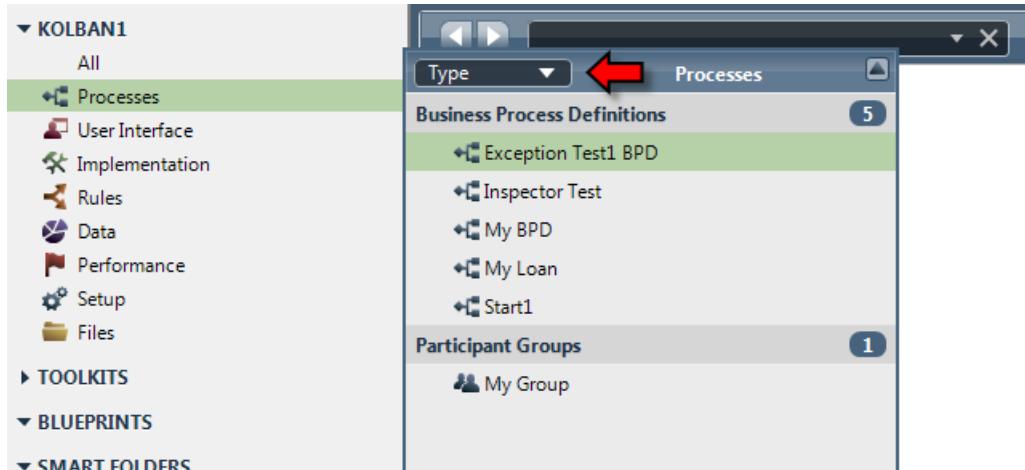
See also:

- Error: Reference source not found

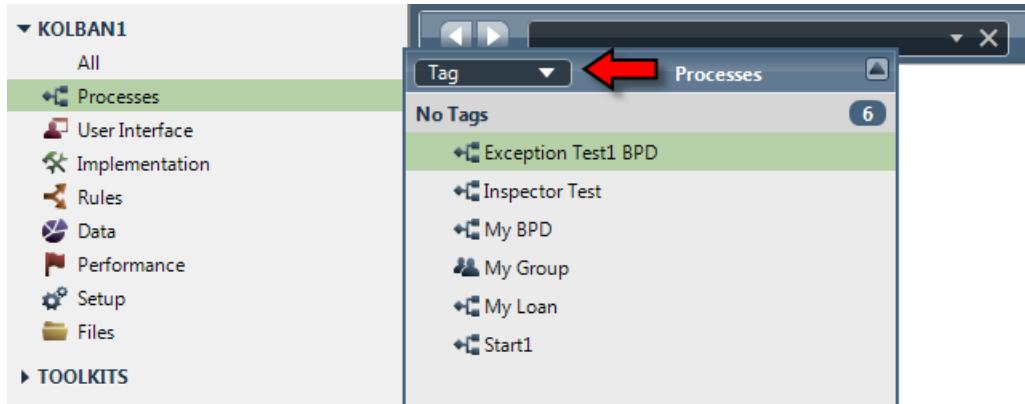
Tagging

When working with artifacts such as BPDs, services, and more, it can sometimes be a challenge to find the parts you are looking for. PD provides an elegant solution to this problem through the technique known as *tagging*. A tag is a keyword that you make up. Think of it as the name of a collection or set. Each artifact that you care about can then be tagged with this keyword. Once tagged, you can then ask IBPM PD to show you the tags of the artifacts or use this attribute in a search or sort filter.

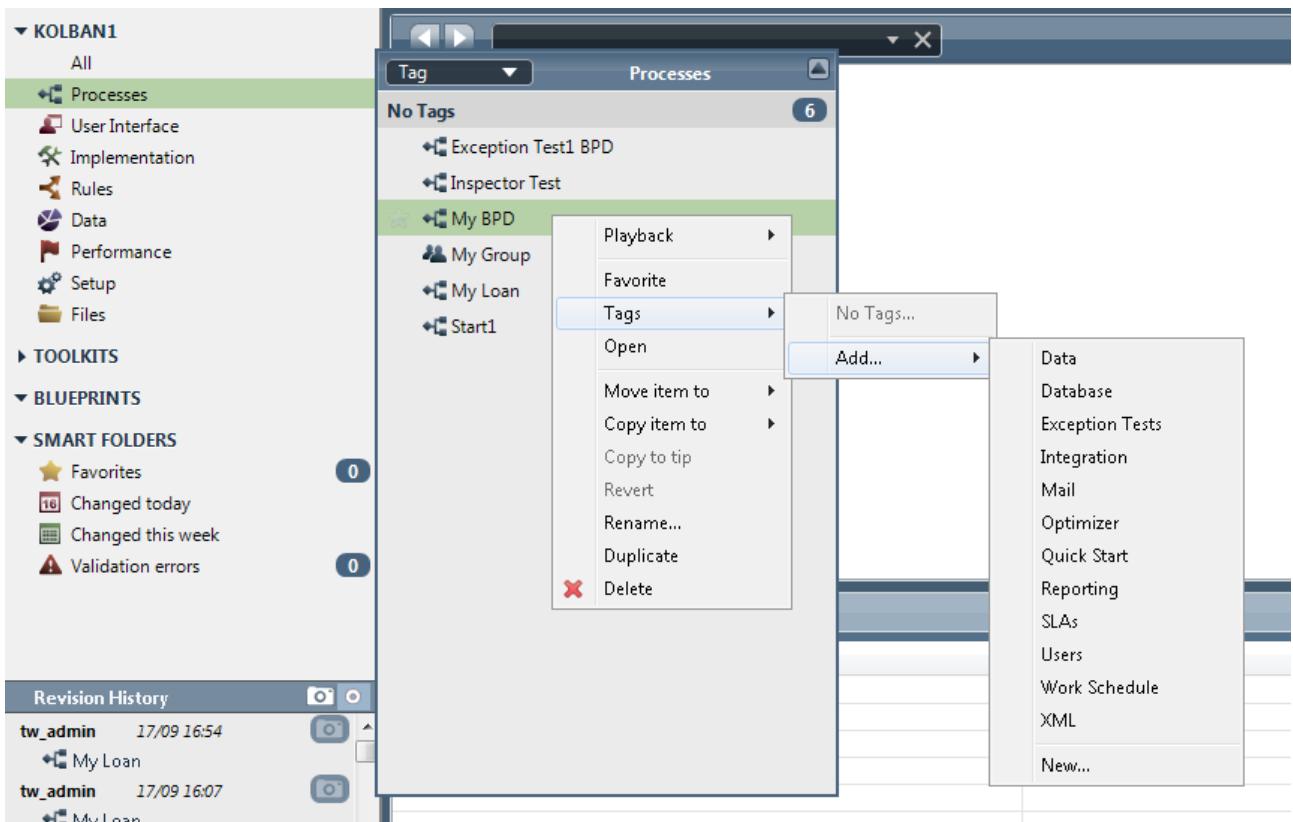
Here is an example. Looking at the BPDs in a Process Application, we may initially see the following list:



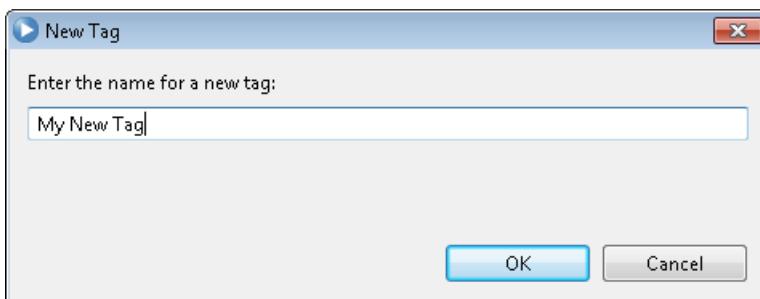
In the top left of the list there is a pull-down that allows us to show how the elements in the list are categorized. By default, they are categorized by type. We can change this to Tag and now the elements are categorized by their tags. Initially, none of the artifacts have any tags associated with them and we see that they belong to the category called "No Tags".



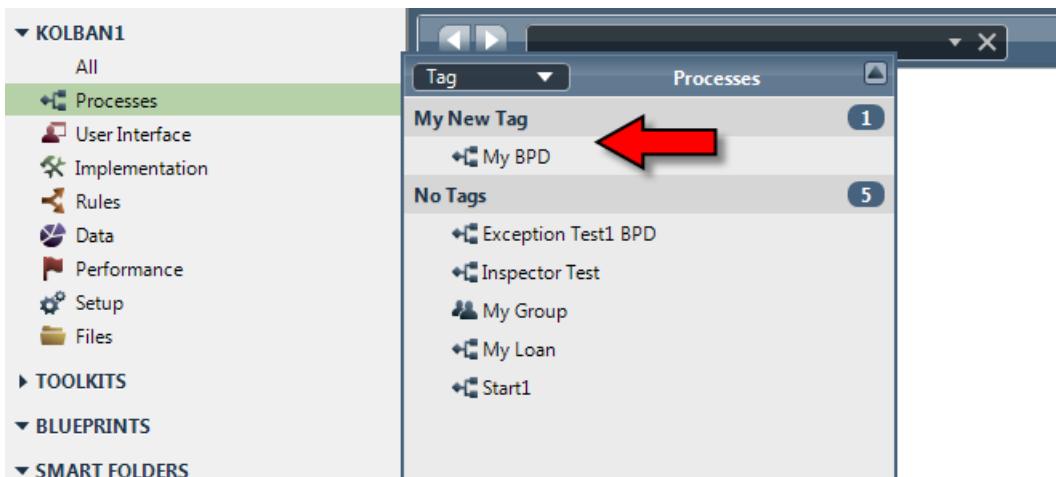
By right-clicking on an entry, we expose its context menu. In that menu there is an entry called Tags. This is where we can associate the entry with one or tag values. If we want to create a new Tag, we can select the New... entry:



Here we see the creation of a new tag:

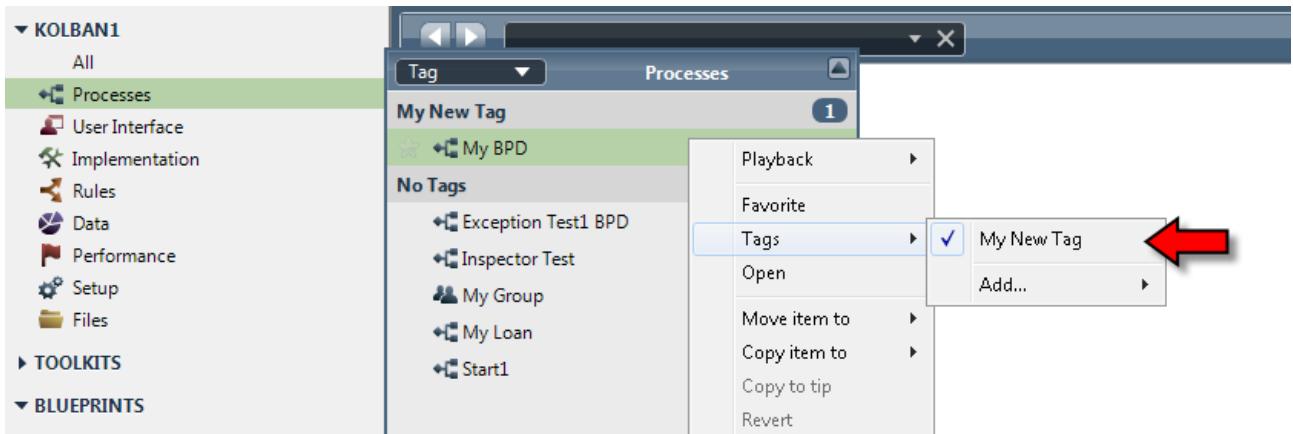


After having tagged an entry, when the lists is shown again, we see that it is part of the grouping for the tag value:

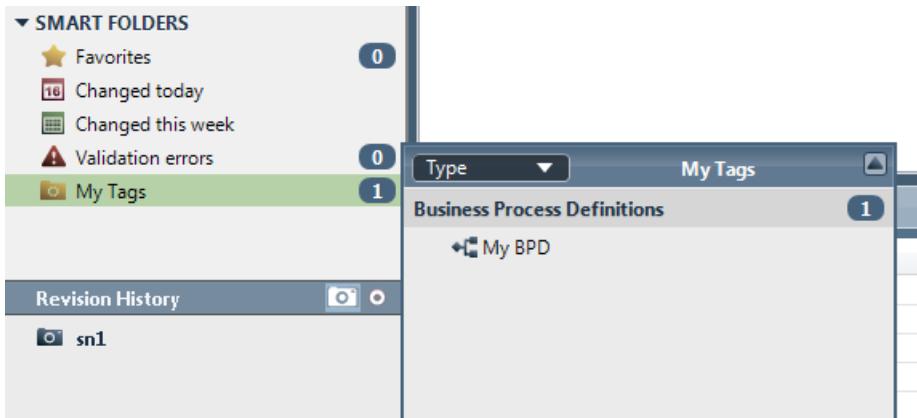


An artifact can be tagged with multiple tags in which case it will appear in the list multiple times, once for each tag.

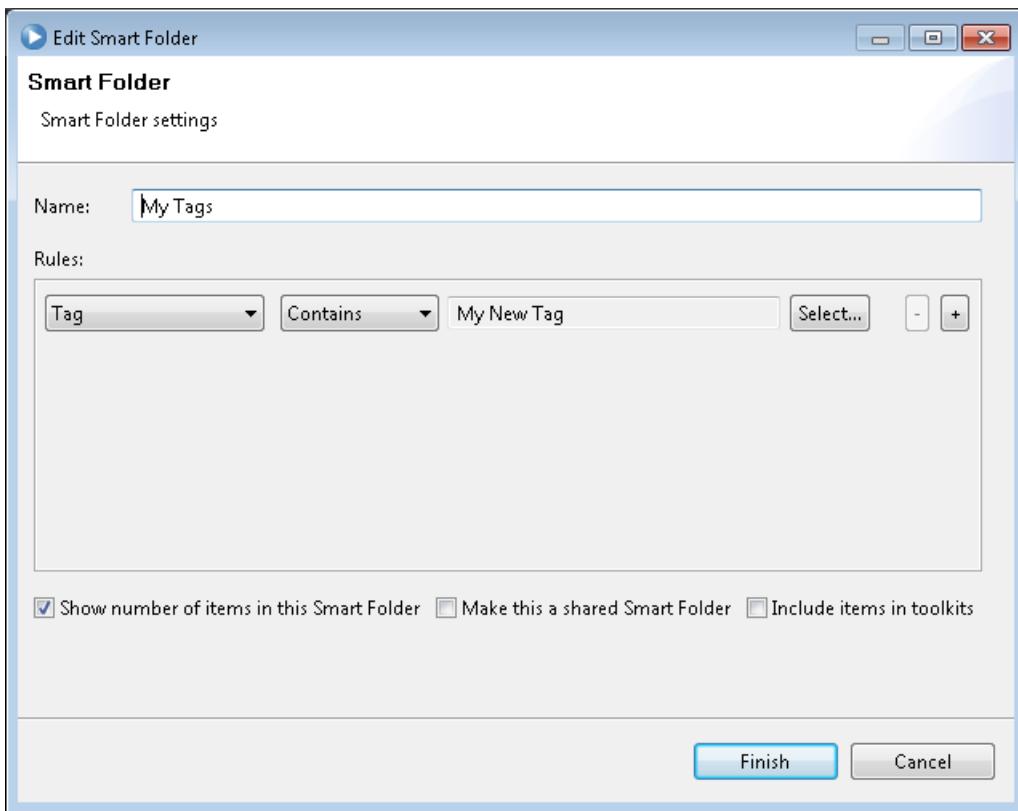
To remove a tag from an artifact, again go to its tag settings where there will be a check mark for each tag associated with the artifact. Un-checking the tag from the artifact will remove it from the tag set:



By combining tags with Smart Folders (see: Smart Folders) we can create a folder that only contains tagged artifacts:



The Smart Folder definition for this may look like:

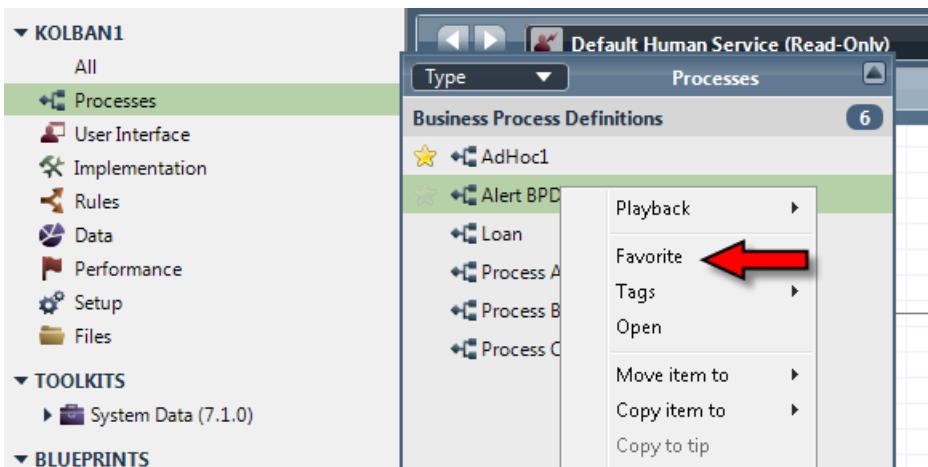


Smart Folders

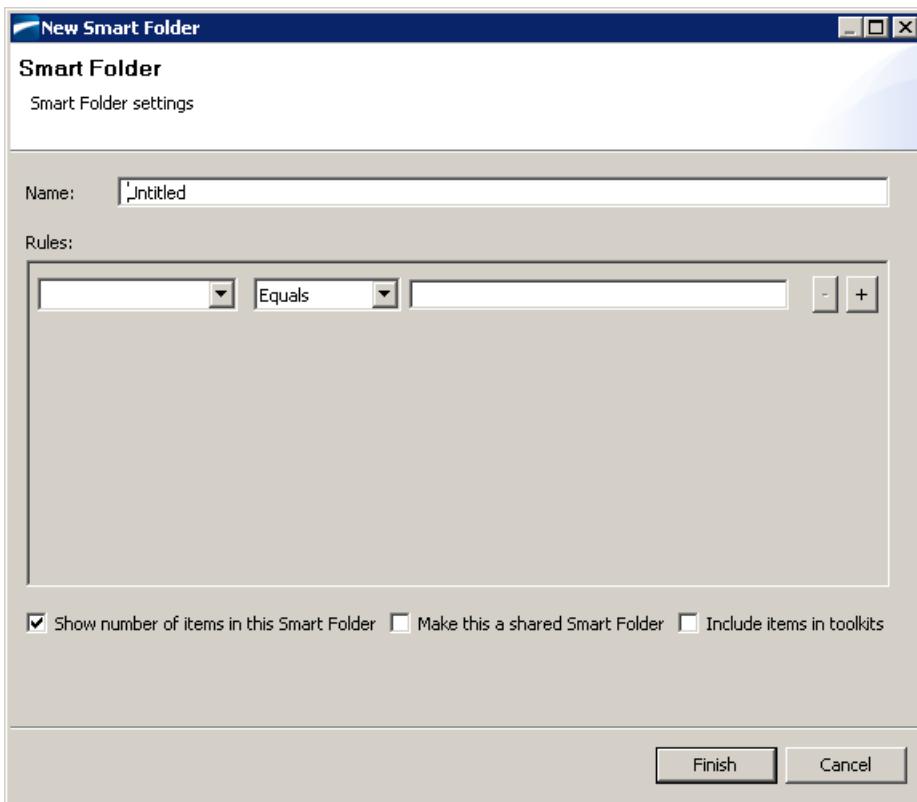
Smart folders are a way of organizing your work and seeing only those artifacts that you need to see at a particular time in a convenient manner. By default, four folders are pre-created for you:

- Favorites – Artifacts tagged as favorite
- Changed today – Artifacts that were changed today
- Changed this week – Artifacts that changed this week
- Validation errors – Artifacts that contain validation errors

To add an artifact as a favorite, bring up the context menu of the artifact that you wish to flag and select Favorite from the menu. A star icon will appear next to the artifacts you have marked as favorites.



In addition to these pre-defined folders, you can create your own custom folders. Clicking the add button to the right of the Smart Folders label, you can create a new folder.

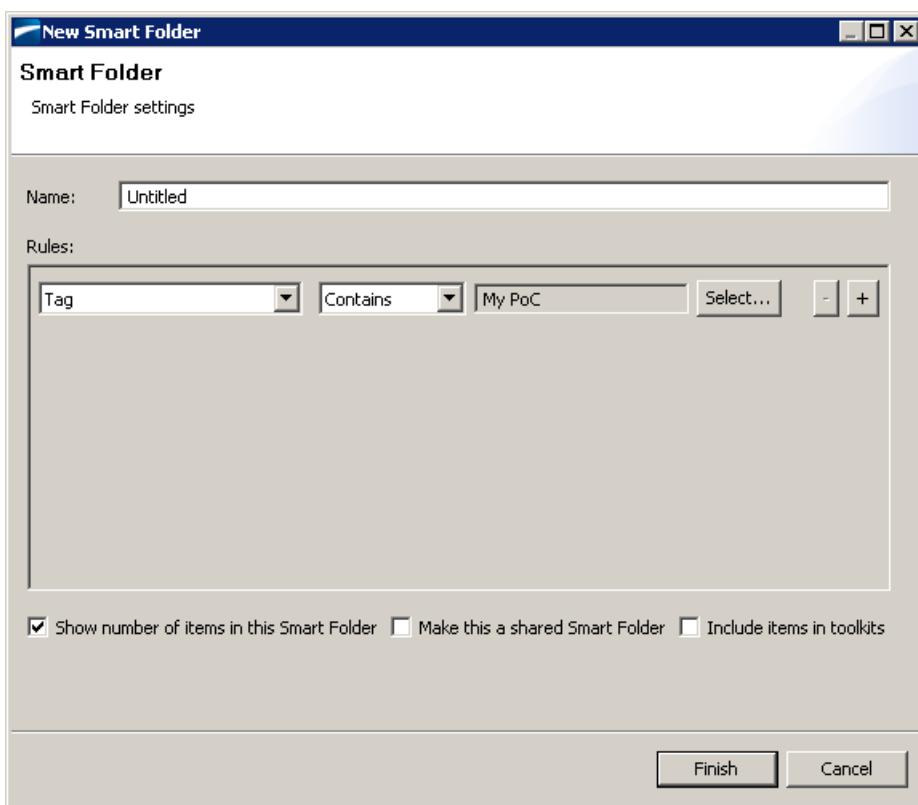


You would give the folder a name and then provide one or more rules that describe the types of artifacts that the folder should contain. These rules contain:

- Category
- Currently Changed
- Currently Changed by other
- Currently Open
- Exposed
- Favorite

- Last Modified By
- Last Modified Date
- Last Modified by me
- Name
- Tag
- Type
- Validation Error
- Validation Warning

For example, to create a folder that shows all the artifacts that are tagged "My PoC", you might use:



One particularly good use of smart folders and tagging is to create a tag called "ToDo" which is used to identify artifacts that still need work before the solution can be deployed. As a solution is built, it is common to build a number of artifacts as place holders for further implementation. Remembering to work on these is the challenge. If a tag called "ToDo" is created and a smart folder defined which shows these items, one can immediately see if there is work outstanding.

See also:

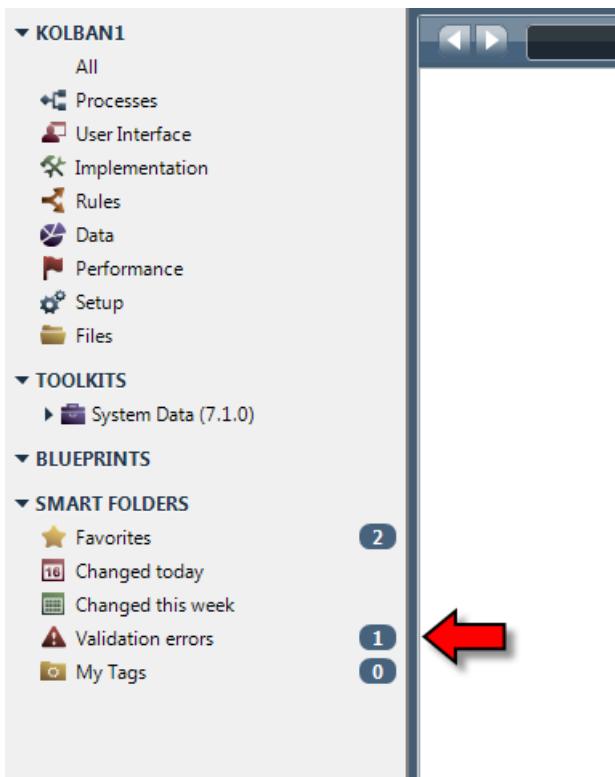
- Validation errors

Validation errors

When ever changes made to a Process Application are saved, the solution as a whole undergoes an automated correctness validation. This will attempt to catch development time errors that may have

been introduced. Validation errors may include such things as missing artifacts, duplicate named service components or no longer supported component types.

The Smart Folders section contains a visual indication of the number (if any) of validation errors detected.



Selecting this folder will show a list of artifacts that are flagged as containing errors. When an artifact is opened that contains an error, the Validation Errors tab will show the details of those errors:

A screenshot of the 'Validation Errors' tab. The tab bar includes 'Properties', 'Validation Errors' (which is selected and highlighted in blue), and 'Where Used'. The main area displays an error message: 'Error Message: The name 'End' is not unique. This can cause problems for debugging and error reporting, but the Service... The Send Task service component is no longer supported.' There is a large, empty text area below the message.

Validation only occurs when artifacts are saved to the Process Center. All validation errors should be corrected before attempting to deploy a Process Application.

See also:

- Smart Folders

Working as part of a team

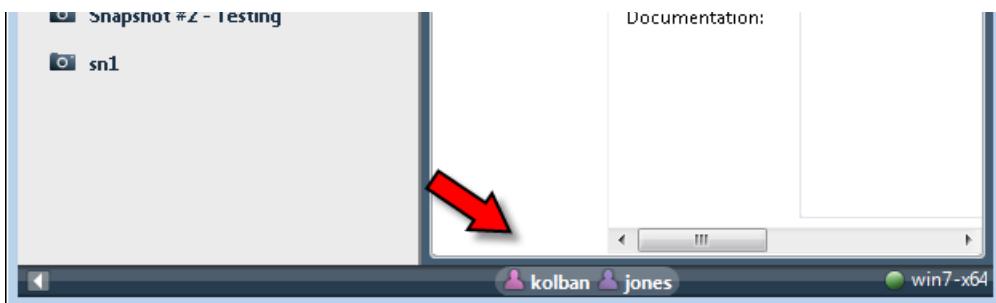
It is extremely common for a team of people to be working on a single Process Application solution

at the same time. Some may be developers, some may be analysts and some may be performing other roles. Since each user of IBPM is using PD to connect to the same Process Center Repository, we need some form of shared notification to ensure that users do not step on each others work and get notified when changes are complete.

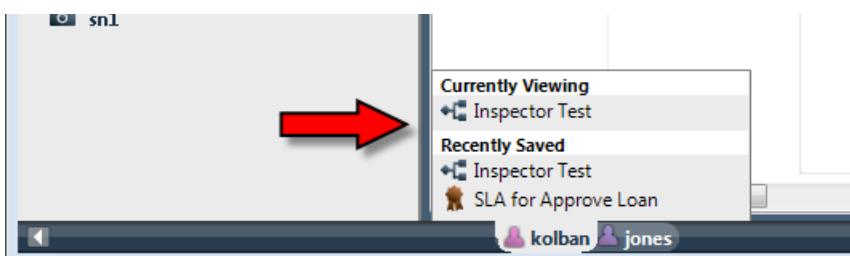
IBPM PD provides a highly elegant solution to the problem.

When multiple users open the same Process Application at the same time, the list of users that have that Process Application open is shown at the bottom of PD.

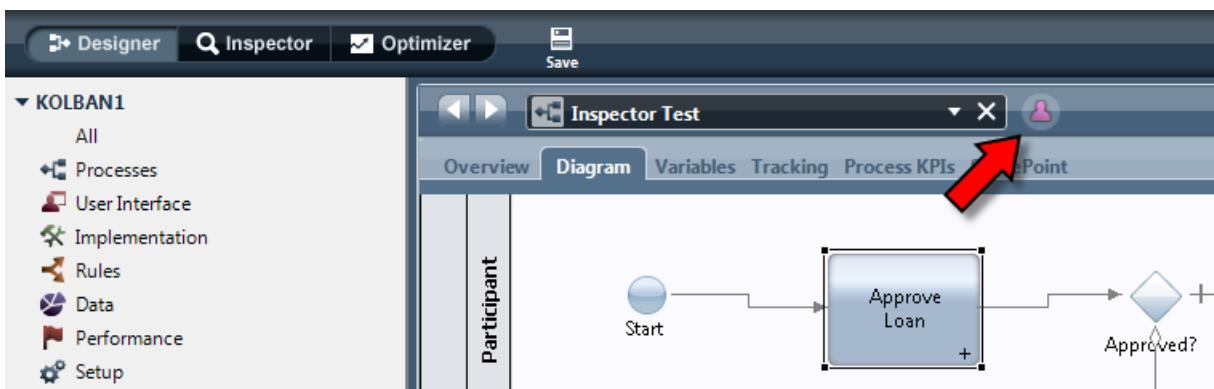
The following illustrates that there are two other users who also have this same Process Application open in their IBPM PD tools.



Clicking on a user brings up details of what that user has open for editing or has recently worked upon:



It is permissible for multiple users to open the same artifact at the same time, in which case a notification that another user has it open will be shown:



If another user makes a change to the artifact, then it will be made read-only for everyone else until the changes are completed or discarded. The indication that it is being edited by another is immediate:



When the changes are completed, the artifact immediately becomes unlocked for the others **and** changes made by others are also reflected in the open editors of all the other users.

Component – Performance Data Warehouse

The Performance Data Warehouse is a database responsible for collecting and managing data originated by Process Server instances. Think of this as the repository for information about the history of the system as is used for reporting of the outcome of processes. The sourcing of data for reporting comes from the single Performance Data Warehouse database which may be associated with multiple Process Server instances. This architecture allows aggregation of information from multiple servers to be achieved. Since generation of reporting knowledge can be computationally expensive, the separation of the Process Servers from the Performance Data Warehouse also allows for reports to be generated without impacting the operation of running processes.

Component – Process Center Console

The Process Center Console provides a web based interface for managing the Process Center maintained projects. This capability is also available within the IBPM PD thick client tool.

Component – Process Portal

The IBPM Process Portal provides the primary end user interface for users to start process instances and see work tasks awaiting their attention.

Component – Process Admin Console

The Process Admin Console is a web based interface for system administrators. It provides a wealth of functions for system operations.

See also:

- Process Admin Console

IBPM Knowledge

There are a variety of sources of knowledge and information related to IBPM. This section captures some of the more important ones. When a question or puzzle on using the product comes to light, the chances are extremely high that the answer can be found in the existing documentation, forums, TechNotes, RedBooks and articles. The real challenge is searching for the answer in the mass of information available.

Product Documentation

The documentation for much of IBPM can be found in the v8.5 [InfoCenter](#).

The InfoCenter can be executed off-line using the IEHS package (IBM Eclipse Help System). To achieve that, download and un-compress the IEHS. The download for the documentation for the 8.5.0 release can be found here:

<http://www-01.ibm.com/software/integration/business-process-manager/library/documentation/>

See also:

- TechNote: [4031355](#) - IEHS is not in the downloadable image and DVD for IBM Business Process Manager (7.5.1)

Web Sites

Some commonly visited web sites on the Internet related to IBPM are:

- [IBM BPM Home Page](#) - The IBM Internet web page related to the BPM product
- [IBM BPM Wiki](#) - A fantastic resource of articles on all versions (past and present) on the IBPM product
- [IBM Education Assistant](#) – The free IBM Education Assistant website provides articles on various IBM products including the IBM BPM suite.

Public Forums

IBM's public Internet accessible developerWorks website has a forum dedicated to IBM Business Process Manager. This is a very active forum and is used by most IBPM consumers and is watched on a best effort basis by most specialists within IBM itself. Support through the forum is provided on a best effort basis and should not be confused with formal PMRs (support requests) through IBM product support.

<https://www.ibm.com/developerworks/community/forums/html/forum?id=11111111-0000-0000-0000-000000002382>

IBM RedBooks

RedBooks have been a staple of knowledge on IBM products for decades. These are free-of-charge down-loadable books that are commonly written by IBM staff outside of their normal work jobs. In addition, RedBook authoring is also open to customers who wish to work closely with the product and other IBM teams. These books are written by practitioners who use the product on very frequent basis and, as such, the books represent real-world experiences and practices. Here is a list of IBPM related RedBooks:

- [Leveraging the IBM BPM Coach Framework in Your Organization](#) – SG24-8210-00 - 2013-02-11
- [Discovering the Decisions within Your Business Processes using IBM Bluworks Live](#) – REDP-4993-00 - 2014-01-30
- [Business Process Management Deployment Guide Using IBM Business Process Manager V8.5](#) – SG24-8175-00 - 2015-01-15
- [Empowering your Ad Hoc Business with IBM Business Process Manager](#) - REDP-4995-00
- [Creating a BPM Center of Excellence \(COE\)](#) – REDP-4898-00 - 2012-09-13
- [Implementing an Advanced Application using Processes, Rules, Events, and Reports](#) – SG24-8065-00 - 2012-09-11
- [IBM Business Process Manager Security: Concepts and Recommendations](#) – SG24-8027-00 - 2012-09-04
- [IBM Business Process Manager \(BPM\) 7.5 Performance Tuning and Best Practices](#) – REDP-4784-00 - 2012-01-27
- [IBM Business Process Manager V7.5 Production Topologies](#) – SG24-7976-00 - 2011-10-20
- [Scaling BPM Adoption from Project to Program with IBM Business Process Manager](#) – SG24-7973-00 – 2011-09-27
- [IBM BPM for Dummies](#) – Although not a redbook, a good overview of BPM and because it is a book, added to this list

IBM Community

Within internal IBM, there is a community that includes a forum and a Wiki for discussing the IBPM product. This can be found at the following URL. This is only available to IBM employees.

[http://w3.ibm.com/connections/wikis/home?
lang=en_US#/wiki/W09eab198c214_436e_a456_02c99b7b475f](http://w3.ibm.com/connections/wikis/home?lang=en_US#/wiki/W09eab198c214_436e_a456_02c99b7b475f)

or at the much more intuitive/readable URL of:

<http://www.ibmbpmcommunity.com>

developerWorks on IBM BPM

IBM's DeveloperWorks is a public Internet based web portal for technical discussions. A large number of papers have been written on IBPM usage. These papers are commonly written by IBM staff when they hear common questions and queries from product users. These papers are of particularly high quality. The current link for developerWorks papers on IBM BPM is:

<http://www.ibm.com/developerworks/bpm/>

- [Teams in Business Process Manager V8.5, Part 1: Modeling teams with IBM Process Designer](#) - 2014-02-19
- [Integrating RESTful web services into a business application in IBM Business Process Manager](#) - 2013-12-11
- [Enable WS-Security and Transport Layer Security in IBM Business Process Manager Standard](#) - 2013-08-28
- [BPM Voices: BPM and Lean -- a powerful combination for process improvement](#) - 2013-08-28
- [Configuring cross data center BPM deployment environment to achieve faster disaster recovery](#) – 2013-08-28
- [Create a mobile BPM application by integrating IBM Worklight and IBM Business Process Manager](#) - 2013-08-28
- [Using the new features of IBM BPM 8.5 dashboards for better views of your business processes](#) - 2013-06-19
- [Deploying IBM Business Process Manager packages to offline process servers](#) - 2013-03-27
- [Designing and implementing a custom inbox and My Team Performance task list with IBM Business Process Manager V8](#) - 2013-03-27
- [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 4: Execution cases](#) - 2013-04-03
- [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 3: Implementing the Advanced Integration Service](#) - 2013-03-27
- [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 2: Defining the business process](#) – 2013-03-13
- [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 1: Introduction and setting up the databases](#) – 2013-03-06
- [Using WebSphere Adapters with IBM Business Process Manager to communicate with external systems](#) – 2013-03-06

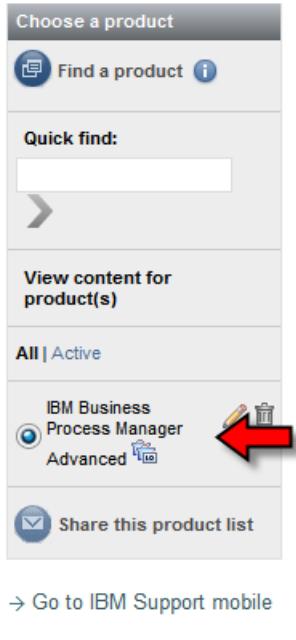
- [BPM Voices: Insight at the edge](#) – 2013-02-13
- [BPM Voices: Evaluating BPM applications: BPM design reviews and Rubik's Cubes](#) – 2013-02-13
- [Configuring IBM Business Process Management Human Task Management widgets for use in WebSphere Portal](#) – 2013-02-13
- [Implementing a negotiation process in IBM Business Process Manager using web services](#) – 2013-02-13
- [Deploying a BPM pattern on IBM PureApplication System](#) – 2012-12-12
- [Care process management: Using BPM tools and methodology in the healthcare environment](#) – 2012-12-12
- [Deploying process applications in IBM Business Process Manager V8](#) – 2012-12-12
- [Automating your business process application test cases using Java in IBM Business Process Manager](#) - 2012-12-05
- [Integrating business applications with IBM Process Center using WebSphere Adapters](#) - 2012-12-05
- [Configuring error handling for Advanced Integration Services in IBM Business Process Manager Advanced V8](#) - 2012-10-19
- [Data4BPM, Part 3: Modeling processes with business entities using extended BPMN](#) - 2012-10-17
- [Creating custom widgets for Business Space using REST services](#) - 2012-09-26
- [Sharing toolkits across Process Centers in IBM Business Process Manager V8](#) - 2012-09-26
- [Leveraging business data attributes in IBM Business Process Manager V7.5 and V8](#) - 2012-09-26
- [Realizing the value of IBM Business Process Manager in the WebSphere Portal environment, Part 3: Agile development of custom user interfaces using Web Experience Factory](#) - 2012-09-26
- [Linking business processes and enterprise services together using IBM Business Process Manager Advanced](#) - 2012-09-26
- [BPM Voices: Where does BPM end and SOA begin?](#) - 2012-09-26
- [Invoking a process deployed in IBM Business Process Manager V7.5 using WebSphere MQ](#) - 2012-07-18
- [A practical approach to integrating JMS and IBM Business Process Manager](#) - 2012-07-10
- [Designing event-driven business processes in IBM Business Process Manager](#) - 2012-06-27
- [Implementing Java integration components for IBM Business Process Manager V7.5.1](#) - 2012-06-13
- [What's new in IBM Business Process Manager V8](#) - 2012-06-13
- [Using Dojo to extend business processes to the mobile space](#) - 2012-03-14
- [Best practices and patterns for customizing human task forms in IBM Business Process Manager V7.5](#) - 2012-02-15
- [Integrating a host application with an IBM Business Process Manager business process](#) - 2012-02-15
- [Integrating business process applications with CICS in IBM Business Process Manager Advanced V7.5](#) - 2012-02-15
- [Developing an application that integrates business process management and case management](#) - 2012-02-15
- [Mobile business process management, Part 1: Extending BPM processes to mobile workers](#) - 2012-02-15
- [Integrating a business process application in IBM Business Process Manager V7.5.1 with an external system using the REST API](#) - 2012-02-08
- [Associating WebSphere Lombardi V7.2 attributes with users defined in OpenLDAP and routing tasks](#) - 2012-01-18
- [Modeling your business processes with IBM WebSphere Lombardi Edition, Part 5: Customize the user experience with Coaches](#) - 2012-01-11
- [Configuring the IBM Business Process Manager V7.5 environment for a typical installation](#) – 2011-12-14
- [Implementing the facade pattern using IBM Business Process Manager Advanced V7.5](#) – 2011-12-14
- [Monitoring business processes with IBM Business Process Manager and IBM Business Monitor](#) – 2011-12-14
- [BPM Voices: What is BPMN 2.0 and why does it matter?](#) - 2011-12-14
- [Capturing and analyzing interface characteristics, Part 1: Capturing integration complexity for BPM and SOA solutions](#) – 2011-12-07
- [Enabling event flow in BPM solutions and best practices for monitor model deployment, Part 1: Configuring and verifying event flow between a clustered Process Server and IBM Business Monitor server](#) - 2011-11-16
- [Creating and sending alerts with IBM Business Process Manager V7.5](#) - 2011-10-26
- [IBM BPM V7.5 orchestration scenarios, Part 2: Straight through processing using IBM Integration Designer](#) - 2011-11-02
- [Integrate an LDAP user registry into a WebSphere Lombardi Edition business process](#) - 2011-10-31
- [Using the Undercover Agent with a message event in WebSphere Lombardi Edition V.7.2](#) - 2011-10-26
- [Using the SQL integration service with WebSphere Lombardi Edition V7.2 and WebSphere Application Server V7](#) - 2011-10-12

- [BPM Voices: The evolution of business process management](#) - 2011-09-29
- [Integrate external services with IBM Business Process Manager Standard or Express applications](#) - 2011-09-29
- [Realizing the value of IBM Business Process Manager in the WebSphere Portal environment](#) - 2011-09-29
- [IBM BPM V7.5 orchestration scenarios, Part 1: Top-down design using IBM Process Designer and IBM Integration Designer](#) - 2011-09-14
- [Setting up a clustered topology for IBM Business Process Manager V7.5, Part 1: A step-by-step guide for BPM Standard](#) - 2011-09-07
- [Using the REST APIs in IBM Business Process Manager V7.5](#) - 2011-08-31
- [Systems integration with WebSphere Lombardi Edition V7.2, Part 1: Architecture and solution overview](#) - 2011-07-13
- [Common business process modeling situations in WebSphere Lombardi Edition V7.2, Part 1](#) – 2011-06-29
- [Common business process modeling situations in WebSphere Lombardi Edition V7.2, Part 2](#) – 2011-07-13
- [Common business process modeling situations in WebSphere Lombardi Edition V7.2, Part 3: Executing an activity multiple times in parallel in a business process](#) - 2011-08-03
- [Secure integration of an LDAP user registry with WebSphere Lombardi Edition](#) - 2011-06-29
- [Business spaces for human-centric BPM, Part 1: Introduction and concepts](#) – 2011-06-29
- [Business spaces for human-centric BPM, Part 2: Interacting with BPD processes and Human Services](#) – 2011-06-29
- [Business spaces for human-centric BPM, Part 3: Interacting with federated processes and human tasks](#) – 2011-06-29
- [Business spaces for human-centric BPM, Part 4: Using custom form renderer for human tasks](#) - 2011-06-29
- [IBM Business Process Manager V7.5 development topology recommendations](#) - 2011-06-29
- [Best practices when using IBM Integration Designer and IBM Process Designer together](#) - 2011-06-29
- [Introducing IBM Business Process Manager V7.5](#) - 2011-06-29
- [WebSphere Lombardi exception handling and logging](#) - 2011-05-25
- [WebSphere Lombardi Edition V7.1: Integrating with the JMS queue, Part 1: JMS integration with the queue hosted on a local Service Integration Bus](#) – 2010-12-08
- [WebSphere Lombardi Edition V7.1: Integrating with the JMS queue, Part 2: JMS integration with the queue hosted on a remote Service Integration Bus](#) - 2010-12-09
- [Configuring a data source in WebSphere Lombardi Edition V7.1](#) - 2010-10-13
- [Integrating WebSphere Process Server V7 and Lombardi Teamworks](#) – 2010-06-30

TechNotes

TechNotes are IBM hints, tips and warnings about IBM products published as web updates at IBM's support web site. To find the latest list of TechNotes, do the following:

1. Visit the IBM support web site at: <http://www.ibm.com/support>
2. Pick IBM BPM from the list of products



3. Click Search Support at the location shown below. "Enter BPM" as the search keyword.

Search support and downloads

BPM

Within my selected proc

4. You now see a list of all things related to Support for the product
5. Select TechNotes as a filter here:

Content Type

Product news
 Product release notes
 Readme file
 Support technical exchange webcast
 Technotes
 Troubleshooting FAQ
 Update

And now you get the list of TechNotes that you were looking for

Search results 

1 - 20 of 103 results [Next ➔](#)

Sort by: [Newest first](#) 

A "message for key: [xxx] is not found" error shows on the Process Portal Console...
After you upgrade the database from IBM Business Process Manager Version 7.5.x to 7.5.1, you might find some messages that are similar to "messge for key: [scoreboard.MyPerf.name] is not found" in the left panel of Process Portal Console under the"My..."
Last modified date: 10 Jan 2012

The test connection to the process center might fail when installing process server...
When you install a stand-alone process server for IBM Business Process Manager using the typical installation mode, the "test connection to process center" step might fail.
Last modified date: 3 Jan 2012

There are TechNotes available for IBPM. The following table lists the important IBPM TechNotes for v7.1, v7.5 and beyond.

ID	Description	Published
1633278	Snapshot deployment for IBM Business Process Manager (BPM) fails, but returns a success message	10/04/13
1651507	IBM Business Process Manager (BPM) notification emails are not being sent	09/30/13
1651217	The Event Manager for IBM Business Process Manager (BPM) does not respond or process any tasks	09/26/13
1638298	A ServiceRuntimeException is thrown by IBM Business Process Manager (BPM) Advanced and WebSphere Process Server (WPS)	09/23/13
1648774	The migration process for IBM Business Process Manager (BPM) fails with a TWU-0038 message while updating the LSW_OPTIMIZER_DATA.STRING_VALUE table	09/10/13
1646787	Applications in cluster members for IBM Business Process Manager (BPM) might fail to start	08/28/13
1647104	A group manager for IBM Business Process Manager (BPM) cannot assign or re-assign tasks to users from external security provider	08/19/13
1644435	Understanding UTLS0002E error messages emitted during application installation	08/15/13
1646097	IBM Business Process Manager (BPM) does not resolve nested group LDAP members with Tivoli Directory Server	08/15/13
1620729	Start up problems occur after a fresh installation of IBM Business Process Manager (BPM) V8.0	08/14/13
1641188	A WorkflowProcessItemException occurs when updating the properties of a FileNet document from IBM Business Process Manager (BPM)	08/09/13
1645818	Tasks assigned to default lane participant group in the Process Portal for IBM Business Process Manager (BPM) can be re-assigned to any users on the system	08/08/13
1645543	Setting the task complete or due date in the Process Portal to a locale other than US English in IBM Business Process Manager (BPM) Version 8.0 and 8.0.1	08/05/13
1644109	For IBM Business Process Manager (BPM), the Process Portal URL changes to the cluster	08/05/13

	member host and port when accessing the Process Portal through the OnDemand router secured port	
1634645	The "You have been automatically logged out for security reasons" error is seen with IBM Business Process Manager (BPM)	08/02/13
1634857	IBM Process Designer fails to connect to the Process Center for IBM Business Process Manager (BPM)	07/31/13
1635594	Multiple selection items load in the wrong order on the coach for IBM Business Process Manager (BPM)	07/30/13
1643703	The IBM Business Process Manager (BPM) BPMMigrateCluster command fails to run with the Oracle RAC database	07/30/13
1628037	Duplicated tables show on the coach when dropping a variable of type List into a Vertical Section in IBM Business Process Manager (BPM)	07/25/13
1619620	A SqlIntegrityConstraintViolationException occurs during the group replication process for IBM Business Process Manager	01/07/13
1615408	A "javax.xml.ws.soap.SOAPFaultException: Error when executing the ruleset" error occurs when integrating IBM Business Process Manager (BPM) and IBM Operational Decision Manager V8.x	01/04/13
1619258	Changing the tw_admin password in IBM Business Process Manager (BPM) after installation	12/06/12
1619260	Application fails to deploy because of a difference in the tw_author password between Process Server and Process Center in IBM Business Process Manager	12/04/12
1618486	A CWLLG2229E message and a NumberFormatException exception occurs when a web service invokes a business process for IBM Business Process Manager	11/30/12
1618958	Deployment environment generation process fails with a NullPointerException in IBM Business Process Manager (BPM) Advanced V8.0	11/30/12
1617817	A NullPointerException occurs when working with complex business objects (BO) while calling Ajax services from Coach views in IBM Business Process Manager	11/29/12
1614992	Submitting a SOAP Request to IBM Business Process Manager (BPM) or WebSphere Lombardi Edition using the SOAPUI results in a no deserializer error	10/24/12
1614220	The override endpoint address might not seem to work during web services invocation for IBM Business Process Manager (BPM)	10/18/12
7036415	Webcast: IBM Business Process Manager (BPM) Security	10/17/12
1613989	Reading and decoding instrumentation files for Teamworks, WebSphere Lombardi Edition (WLE), and IBM Business Process Manager (BPM) products	10/16/12
1613695	What must I do if I cannot log in to the Process Center perspective in IBM Integration Designer?	10/11/12
1610883	IBM Business Process Manager (BPM) runtime migration support	10/09/12
1611013	Disabling tracking data generation for a Process Server or Process Center in IBM Business Process Manager (BPM)	10/04/12
1612503	Announcing: IBM Business Process Manager, IBM Business Monitor, and IBM Integration Designer Version 8.0.1 products	10/02/12
1608265	Using Business Action Language (BAL) rules in a decision service when the input or output parameters are lists in IBM Business Process Manager (BPM) Advanced	10/02/12
1612772	The server start up process for IBM Business Process Manager (BPM) cannot find LSW_PRI_KEY, LSW_USR, or LSW_USR_XREF tables	10/01/12
1612759	Viewing tasks in Process Inspector for IBM Business Process Manager (BPM) results in date error	10/01/12
1610113	Querying LDAP databases using IBM Business Process Manager (BPM) V8.0	09/17/12
1605686	An administrator is unable to reassign tasks through the Process Portal for IBM Business	09/14/12

	Process Manager (BPM)	
1609973	Making the role binding effective immediately for IBM Business Process Manager (BPM) V8.0	09/04/12
1589168	WSDL discovery fails for the IBM Business Process Manager (BPM) Process Designer when the Process Center and Process Designer are on different machines	08/03/12
1595333	IBM Business Process Manager (BPM) Advanced installation fails when installing on an existing WebSphere Application Server (WAS) Network Deployment V8 installation	07/17/12
1601072	No email is received even if email notification has been enabled in IBM Business Process Manager (BPM)	07/12/12
1601069	A java.lang.RuntimeException is thrown when saving the participant group rules definition for IBM Business Process Manager (BPM)	07/12/12
1600149	Federation reports invalid attribute after removing it from a backend system	07/02/12
1595098	Launchpad does not open on AIX or Linux operating systems for the IBM Business Process Manager (BPM) 7.5.x and IBM Business Monitor V7.5.x products	06/29/12
1600168	A UOWManager NoClassDefFoundError is displayed in Process Designer for the IBM Business Process Manager (BPM) products	06/29/12
1599828	The StartService REST API fails to start a service with the IBM Business Process Manager (BPM) V8.0 products	06/27/12
1599349	NullPointerException occurs when testing the Advanced Integration Service In IBM Business Process Manager (BPM) Advanced 7.5.1	06/27/12
1593461	Collect troubleshooting data for web services problems in IBM Business Process Manager	06/26/12
1597901	IBM Business Process Manager (BPM) Advanced cannot connect the Process Inspector to the Process Server	06/08/12
1507414	A performance delay occurs when creating an adhoc human task on a WebSphere Process Server (WPS) or IBM Business Process Manager Advanced clustered system	06/08/12
1597552	"TypeError: 'handleBidi' undefined" error is seen in Process Portal for IBM Business Process Manager (BPM)	06/05/12
1597258	The BPMInstallOfflinePackage Admin Task fails for IBM Business Process Manager (BPM)	06/04/12
1590115	Configuring a routing server for IBM Business Process Manager (BPM) Process Portal in a three or four cluster topology	06/04/12
1595949	Modifying the full name of an internal user does not take effect for IBM Business Process Manager (BPM)	05/24/12
0	Configuring a routing server for IBM Business Process Manager (BPM) Process Portal in a three or four cluster topology	06/04/12
1591751	A CWLLG2101W message indicates that new transactions are not started for IBM Business Process Manager (BPM)	05/21/12
1593299	IBM Business Process Manager (BPM) Advanced and IBM Business Process Manager Standard internal custom repository user name fails with a CWLLG2015E error message	05/18/12
1593277	A typical installation for IBM Business Process Manager (BPM) does not work	05/18/12
1593423	Error messages about database unique constraints can occur when starting an IBM Business Process Manager (BPM) clustered server	05/18/12
0	IBM Business Process Manager (BPM) installation of IBM Process Designer fails with "Access is denied" message	05/18/12
0	A task assigned to a rule-based participant group cannot be run from the Process Designer Inspector after being upgraded by certain users in IBM Business Process Manager Advanced (BPM)	05/18/12
0	Test connection problem occurs for the Business Process Choreographer messaging engine data source for IBM Business Process Manager (BPM) Advanced	05/18/12

1595333	IBM Business Process Manager (BPM) Advanced installation fails when installing on an existing WebSphere Application Server (WAS) Network Deployment V8 installation	05/18/12
1593555	Start menu items for starting, stopping, and administering profiles are lost after upgrading IBM Business Process Manager (BPM)	05/18/12
1593313	Testing a Human Service in IBM Process Portal	05/18/12
1595478	Some Business Space page management features do not work after restarting the server if the Java Virtual Machine (JVM) is stopped or abruptly terminated in IBM Business Process Manager (BPM) Advanced or in IBM Business Monitor	05/18/12
1594799	The IBM Business Process Manager (BPM) configureNode command fails to create a cluster if a password contains prohibited special characters	05/18/12
0	IBM Business Process Manager (BPM) installation path on Windows must be kept short to create a network deployment environment	05/18/12
1593114	A SRVE0068E error occurs when an LDAP user is added to the tw_admins group when using IBM Business Process Manager (BPM) Advanced and IBM Business Process Manager Standard	05/18/12
1594811	Slow performance when importing or using Process Applications or Toolkits in IBM Integration Designer (IID)	05/18/12
0	Profile creation fails in Business Space when the database password has a combination of ' (single quote) and " (double quote) in IBM Business Process Manager (BPM) Advanced and IBM Business Monitor	05/18/12
0	Trust store problem when connecting from Process Designer to Process Center in IBM Business Process Manager (BPM)	05/18/12
0	IBM Business Process Manager (BPM) Advanced users who use an IP address to access Process Portal might need to log in a second time when opening a task	05/18/12
1592969	"Type reference is unresolved" error after bringing a Process Application or Toolkit into an IBM Integration Designer (IID) workspace	05/18/12
1594923	The default values for an Oracle database do not work when configuring Business Process Choreographer using the bpeconfig.jacl command in IBM Business Process Manager (BPM) Advanced	05/18/12
0	When using IBM Business Process Manager (BPM) with Microsoft SQLServer, deadlocks can occur if process applications, toolkits, and tracks are created concurrently	05/18/12
1594756	In an environment that includes IBM Business Process Manager (BPM), creating a profile without IBM Business Process Manager can cause Business Space to be replaced by Process Portal	05/18/12
0	Installing the required fixes for the IBM Business Process Manager V8.0 products and IBM Business Monitor V8.0	05/17/12
1594714	Server startup problems occur when using IBM Business Process Manager (BPM) with Lightweight Directory Access Protocol (LDAP) for a large number of groups	05/17/12
1585928	Timeout Error or Thread Hang occurs with WebSphere Process Server (WPS) and IBM Business Process Manager (BPM) Advanced during an unused staff query cleanup	05/16/12
1592494	Password limitations exist for IBM Business Process Manager (BPM) when running specific batch commands during installation of process applications to an offline Process Server	05/09/12
1580089	A "CWLLG0095W: The repository contact failed with a status of: 302" error occurs with IBM Business Process Manager (BPM)	05/09/12
1592149	The "Read E-Mail via IMAP" integration service for the IBM Business Process Manager (BPM) products does not support SSL	05/09/12
1591697	Changing ownership of an IBM Business Process Manager (BPM) Process Center or Process Server	04/25/12
1591625	Problem enabling communication between Process Designer and Process Center for IBM	04/24/12

	Business Process Manager (BPM)	
1432275	Database transaction log space full error when deleting a process instance	04/20/12
1591154	Bootstrap for the IBM Business Process Manager (BPM) products fail with SQLCODE=-964, SQLSTATE=57011	04/19/12
0	Imported BPMN 2.0 file fails with a NullPointerException in the SystemOut.log file with the IBM Business Process Manager (BPM) products	04/18/12
1589723	WebSphere Application Server V7.0 import function in Installation Manager V1.5.x is only supported with graphical mode and silent response file	04/17/12
1511784	Cannot log into Business Space	04/17/12
0	Migration of Business Space from Version 6.x to Version 7.x may affect user/group access to pages	04/16/12
0	The createDatabase.sql file is missing for an Oracle Database with the IBM Business Process Manager (BPM) products	04/16/12
0	IBM Business Process Manager (BPM): Process Server and Performance Data Warehouse configuration links are missing in the administrative console	04/11/12
1588515	Launchpad hangs at 41% during a typical installation for the IBM Business Process Server (BPM) products	04/06/12
1589793	A "./configCommonDB.sh: line 30: temp.sql: command not found" error occurs with the IBM Business Process Manager (BPM) products	04/05/12
0	Cannot connect to Business Space WebDAV	04/05/12
0	"DSRA1300E: Feature is not implemented" error is seen from the Microsoft SQL JDBC driver with the IBM Business Process Manager (BPM) products	04/04/12
1439814	How to write services that consume as little heap as possible	03/29/12
1587668	The Process Portal for the IBM Business Process Manager (BPM) products does not render properly when using Load Balancer	03/26/12
1588225	Bootstrap failed and Connection refused errors occur with the bootstrapProcessServerData.sh command for the IBM Business Process Manager (BPM) products	03/20/12
0	Prerequisites for using an existing DB2 database with a typical installation	03/16/12
0	Console View shows garbled Japanese character	03/16/12
1587867	Using Business Action Language (BAL) services in IBM Business Process Manager (BPM) products might result in an exception	03/16/12
1587229	Swimlane disappears after re-import BPMN 2.0 file into Process Designer for IBM Business Process Manager (BPM)	03/12/12
1571924	Configuring SSL for IBM Business Process Manager V7.5.x	03/09/12
0	The upgrade_7x.sh/bat database migration script fails with the "Database design file can't be found" message for IBM Business Process Manager (BPM)	03/09/12
1586586	Error: You cannot maintain internal users because the application server is not configured to use the Business Process Manager (BPM) Internal Security Provider.	03/02/12
0	The Install Launchpad hangs or has no content on start up for WebSphere Enterprise Service Bus V7.5.1	03/02/12
1571624	Service Deploy reports error CWZMU0062E when a Data Handler primitive configuration file is contained within a library project	03/02/12
1586266	The "Send E-mail via SMTP" integration service fails to handle Double Byte Character Set (DBCS) in the message body for IBM Business Process Manager (BPM)	03/02/12
1585971	The button to add a new offline process server is missing and the Admin tab does not display for IBM Business Process Manager (BPM) Advanced	03/02/12
1575124	An EngineRequestRejectedException is seen when sending a message to a BPEL process for	02/28/12

	WebSphere Process Server (WPS) and IBM Business Process Manager (BPM) Advanced	
0	Cannot use the domain user ID to create a new database in IBM DB2 Express with IBM Business Process Manager	02/28/12
1569693	Using the Update Installer utility (UPDI) to apply interim fixes and updates can cause inconsistencies with IBM Installation Manager	02/27/12
1571625	Deployment Environment generation fails to create Business Space tables with an Oracle Common Database	02/23/12
1584411	A CWMCO0908E error message is seen when the upgraded cluster does not match the node level for IBM Business Process Manager (BPM) Advanced	02/14/12
1503268	IBM Business Process Management V7.5 installation fails during an IBM DB2 express installation	02/13/12
1571924	Configuring SSL for IBM Business Process Manager V7.5.x	02/07/12
1579275	The upgrade7x command fails to clean up the SIBus tables during database upgrade to the IBM Business Process Manager (BPM) V7.5.1 products	02/06/12
1515691	IBM Business Process Manager 7.5 Business Space: Folder creation fails using WebDAV	02/03/12
1575491	IBM Business Process Manager (BPM) update fails due to a mkdir error	01/24/12
1577551	A "message for key: [xxx] is not found" error shows on the Process Portal Console after upgrading to the IBM Business Process Manager (BPM) 7.5.1 products	01/10/12
1572465	The test connection to the process center might fail when installing process server for IBM Business Process Manager (BPM)	01/03/12
1576147	Multiple system toolkits cause java.lang.NullPointerException in IBM Business Process Manager (BPM)	12/21/11
1573584	Welcome Space does not display after a Business Space upgrade	12/14/11
1570464	Troubleshooting IBM Business Process Manager: "403 BMWPX0006E: The URL you tried to access through the proxy is not allowed"	12/12/11
1573113	English and non-English language text is mixed in the administrative console for IBM Business Process Manager (BPM)	12/06/11
1572521	Business Space does not start on IPv6 system because of a DB2 SQL error -302	12/05/11
1570097	On Linux platform, IBM Business Process Manager install failed on the DB2 installation	12/04/11
1570089	Responding to warnings encountered while installing IBM DB2 Express	12/04/11
1574106	A "CWLCB0020E: The Common Base Event browser is unavailable" displays when you log into the Common Base Event Browser for IBM Business Process Manager (BPM)	12/04/11
1574113	Performance tuning considerations when IBM Business Process Manager (BPM) is running in a virtual machine	12/02/11
1573982	Default Maximum Connections is too small in IBM Business Process Manager (BPM) 7.5	12/02/11
1500774	A Typical install of the IBM Process Server results in a failure of the installation process for IBM Business Process Manager (BPM)	11/30/11
1573217	Process or service fails with "Error Starting Process" or "Runtime Error Information" message	11/28/11
1572472	How do you distinguish between human and system activity using the REST API for IBM Business Process Manager (BPM)?	11/28/11
1572465	The test connection to the process center might fail when installing process server for IBM Business Process Manager (BPM)	11/28/11
1571843	Importing a Teamworks file that references content from LDAP into IBM Business Process Manager (BPM) can fail	11/18/11
7023411	Recommended fixes for IBM Business Process Manager 7.5.1, IBM Business Monitor 7.5.1, and IBM WebSphere Enterprise Service Bus 7.5.1	11/18/11

4031355	IEHS is not in the downloadable image and DVD for IBM Business Process Manager	11/18/11
4031386	WebSphere Application Server V7.0.0.19 Fix Pack for business process management	11/18/11
1571627	Using default password when installing WebSphere Enterprise Service Bus with DB2 Express fails due to invalid password.	11/18/11
1501295	Error message in IBM Business Process Manager V7.5 indicates that a property has not been declared	11/18/11
7023410	Update Instructions for V7.5.1 Refresh Pack for the IBM Business Process Manager products	11/18/11
1569450	Deploying a snapshot with monitoring for IBM Business Monitor to a network deployment environment fails	11/18/11
1572347	IBM Business Process Manager V7.5.1 is Available	11/18/11
1569343	In a complex flow condition, if all instances return a false value, the flow stops unexpectedly for IBM Business Process Manager (BPM)	10/25/11
1568782	You might not be able to install the Process Designer with IBM Business Process Manager (BPM) Version 7.5	10/20/11
1567195	Microsoft Windows shortcuts do not use the host name specified during the installation of IBM Business Process Manager V7.5	10/07/11
1514927	A typical installation of IBM Business Process Manager (BPM) Version 7.5 on Microsoft Windows 7 fails during the profile creation steps	09/30/11
1516235	IBM Business Process Manager (BPM) V7.5 with a Microsoft SQL database throws a lock request time out period exceeded error	09/20/11
1506937	Collect troubleshooting data for installation or upgrade problems with IBM Business Process Manager	09/07/11
1512262	How do you resolve duplicated scoreboards problem in Process Portal for IBM Business Process Manager Advanced?	09/07/11
1514706	With IBM Business Process Manager products, an Oracle database can display errors when under heavy process load	09/06/11
1509210	A CWTBG0013E error occurs when service data is retrieved using REST Service API for IBM Business Process Manager Advanced	08/23/11
7021829	Frequently asked questions (FAQ) about IBM Business Process Manager Advanced (last updated August 17, 2011)	08/20/11
1508659	A CWLLG1162E error occurs with IBM Business Process Manager	08/17/11
1508496	Increased the session timeout values for Business Space components	08/17/11
1508494	Unable to load custom Business Space widgets	08/15/11
7022332	IBM Business Process Manager Advanced Edition Version 7.5 — Configuring a Clustered Process Server	08/10/11
1508665	The CWLLG2015E: Error: You are not authorized to enter the repository view of a non-repository server occurs when you attempt to access the ProcessCenter console For IBM Business Process Manager	08/11/11
1506813	The "align" attribute of button group is a Japanese character even after importing the process to IBM Business Process Manager 7.5	08/09/11
7022298	Installation Instructions of V7.5.0 Fix Pack 1 (V7.5.0.1) for IBM Process Designer	08/09/11
7022243	Known Issues: IBM Business Process Manager products V7.5.0 Fix Pack 1 (V7.5.0.1)	08/09/11
4030569	Version 7.5.0 Fix Pack 1 for the IBM Business Process Manager products	08/09/11
1508116	IBM Business Monitor plugins are not updated by IBM Business Process Manager 7.5.0 Fix Pack 1	08/09/11
1508135	Configure Business Space super user failed when running createSuperUser.py with stand-alone LDAP security mode.	08/08/11

1504755	Using the XSL connector for WebSphere Lombardi Edition and IBM Business Process Manager	07/07/11
1452291	Enabling HTTPS for WebSphere Lombardi Edition Portal	06/24/11
1460358	WebSphere Lombardi Edition 7.1 / 7.2 Cluster Configuration	06/23/11
1503211	Error CWTBG0019E: Unexpected Exception occurs when starting the process using the REST interface with IBM Business Process Manager	06/22/11
1502592	JMS Failover Configuration in WebSphere Lombardi Edition 7.1 and 7.2	06/16/11
1502889	Custom XML overrides are not picked up in WebSphere Lombardi Edition	06/16/11
1501938	Changing the database password for a WebSphere Lombardi Edition Version 7.1 or 7.2 full installation	06/14/11
1498998	Libraries that might be required as pre-requisites for installing Websphere Lombardi Edition on Linux operating systems	05/12/11
1500897	With IBM Business Process Manager products, IBM DB2 can display errors when under heavy process load	06/05/11
1496717	Some XML schema constructs are not supported in IBM Business Process Manager Advanced	06/03/11
1499592	AIS does not refresh automatically in the Inspector view in IBM Business Process Manager Advanced	06/03/11
1501794	Interim Fixes are required for IBM Business Process Manager Version 7.5 products	06/03/11
1500741	Advanced integration services (AIS) do not participate in the same transaction as business processes	06/03/11
1500698	You might experience process recovery issues in certain situations in IBM Business Process Manager V7.5	06/03/11
1499538	How to resolve memory issues for IBM Business Process Manager V7.5 servers	06/02/11
1499983	Performance considerations for Human Task Management list widgets with IBM Integration Designer	06/02/11
1500111	Installing IBM Integration Designer and the test environment with an existing IBM Rational Application Developer 8.0.x installation might result in unexpected options	06/02/11
1499353	You cannot see long descriptions in dialog boxes and wizards in IBM Integration Designer	06/02/11
1499474	Duplicate models, duplicate batches, or broken links might be created in IBM Process Designer and IBM Integration Designer	06/02/11
1501042	Validation of mediation flows does not occur when changes are made to components in the Assembly Diagram	06/02/11
1500842	You cannot install IBM Integration Designer and its test environment directly from the DVD	06/02/11
1499944	Errors occur when uninstalling multiple server and tooling products	06/02/11
1499978	Process Designer name changes not handled by synchronization	06/02/11
1501037	XSLTransformation failures might occur in mediation flows that use WSDL in RPC style when executing in business objects (BO) Lazy Parsing mode	06/02/11
1496717	Some XML schema constructs are not supported in IBM Business Process Manager Advanced	06/02/11
1501039	XSL Transformation primitive in Business Objects (BO) Lazy Parsing mode producing incorrectly typed correlation, transient, and/or shared context elements	06/02/11
1498998	Libraries that might be required as pre-requisites for installing WebSphere Lombardi Edition on Linux operating systems	05/12/11
1459956	Un-installing a WebSphere Lombardi Edition Express Install (Simple Configuration) on Microsoft Windows	04/28/11

1497156	Change the font type in a WebSphere Lombardi Edition report	04/27/11
1497392	Using twint to recreate databases in Teamworks 7.0.x and WebSphere Lombardi Edition	04/22/11
1497231	WebSphere Lombardi Edition Access Control Settings Overview	04/20/11
1497085	startBpdWithName API fails if its run inside an undercover agent (UCA) for WebSphere Lombardi Edition	04/20/11
1497096	When installing WebSphere Lombardi Edition, some fonts do not display correctly	04/19/11
1446648	How to tune your WebSphere Lombardi Edition Servers for performance	04/08/11
1448625	Exposed Process Values are not global in WebSphere Lombardi Edition / Teamworks Version 7.x	03/29/11
1462333	The coach layout for column widths in IBM WebSphere Lombardi Edition is not correct	03/24/11
1469796	Exporting a snapshot does not complete or is corrupt	03/04/11
1462314	Addendum: Oracle pre-requisite patches for 11gR2 with WebSphere Lombardi Edition	03/02/11
1469410	Tasks failing with "Error converting data type decimal to decimal"	02/28/11
1439787	Task notification e-mails coming from "tw_admin"	02/16/11
1439779	How do I set the security access for editing the process help wiki?	02/16/11
1439703	How do you assign a Calendar to a User or Role in the TeamWorks Admin Console?	02/15/11
1462333	Regression: Coach layout for column widths broken	02/10/11
1461468	WebSphere Lombardi Upgrade Readiness Check Download and Usage	02/10/11
1447860	Start Message Event in a Toolkit	02/08/11
1461918	Logging SQL and its parameters to diagnose performance and data integrity issues	02/07/11
1447788	How to change the database password on WebSphere Lombardi Edition (WLE) 7.1 Simple Install	02/04/11
1459782	DB2 failure during WLE 7.2 Express Installation and Configuration (Simple Installation)	02/02/11
1439686	How to Report Lombardi Product Defects or Enhancements	02/02/11
1461206	Websphere Lombardi Edition 'Custom' type installation is supported on Windows XP SP3 for demonstration and sandbox purposes	01/31/11
1460358	WebSphere Lombardi Edition 7.1 / 7.2 Cluster Configuration	01/21/11
1459956	Uninstalling a WebSphere Lombardi Edition Express Install (Simple Configuration) on Windows	01/21/11
1460193	WebSphere Lombardi Edition 7.x: Configuring secondary schema in Oracle	01/21/11
1460187	WebSphere Lombardi Edition 7.x: Configuring secondary schema in Microsoft SQL Server	01/21/11
1439813	Diagnosing Performance problems	01/05/11
1439814	How to write services that consume as little heap as possible	01/05/11
1440086	MustGather: Read First for Problems with WebSphere Lombardi Edition	12/30/10
1456896	org.eclipse.equinox.launcher errors when logging into WebSphere Lombardi Edition (WLE) Authoring Environment	12/30/10
1458731	configure_cluster_member.cmd output	12/29/10
1457259	Cannot import a TeamWorks/WebSphere Lombardi Edition export if it contains multiple toolkit snapshots	12/12/10
1457363	IBM WebSphere Lombardi Edition 7.1.0 and 7.2.0 support uplevel fix pack versions of IBM WebSphere Application Server Version 7.0	12/10/10
7019834	Known problems and limitations in WebSphere Lombardi Edition V7.2	12/10/10
7019837	Resolved issues in WebSphere Lombardi Edition V7.2	12/10/10

7019760	IBM WebSphere Lombardi Edition, Version 7.2.0, hardware and software requirements - Windows on x86	12/08/10
1452197	Instructions for changing the heartbeat user password on runtime server and process center after system has registered	12/01/10
1439664	Report is not displayed in TeamWorks and Java process consumes CPU	11/30/10
1439821	Services run from the Favorites menu leave tasks in the Teamworks Portal Inbox	11/30/10
1455298	Hot Fix 10100171 Available for WebSphere Lombardi Edition V7.1	11/24/10
1452668	Excessive DB connections opened by WebSphere Lombardi Edition under load	11/23/10
1452663	Improving performance of processing LOB data in WebSphere Lombardi Edition with DB2 database	11/22/10
1445620	Detailed steps for moving a runtime server from online to offline and back to online	11/22/10
1454579	Process Admin Console's Role Bindings on runtime cluster nodes may be out-of-sync until E@GroupMembers cache reset	11/16/10
1448203	How to setup Single Sign-On (Kerberos) on WebSphere Lombardi Edition 7.1 on Windows	11/11/10
1452637	Required DB2 settings to use with WebSphere Lombardi Edition 7.1.0	11/08/10
1452583	Performance issue in WebSphere Lombardi Edition V7.1 with Oracle in UserRegistry	11/04/10
1452575	Null Pointer Exception (NPE) may occur using TWParticipantGroup.addUsers(TWUser[] String[]) if one of the users is null or contains empty string	11/04/10
1452340	Check Box Control onChange and onBlur events do not work as expected in Internet Explorer (IE)	11/03/10
1452323	DB2 Health Monitor can cause Log Full Errors: SQLCODE: -964, SQLSTATE: 57011	11/02/10
1444180	Deploying and redeploying applications in WebSphere Lombardi Edition 7.1	10/26/10
1439852	Error The transaction log for database is full. To find out why space in the log cannot be reused	10/20/10
1439541	Error at login screen: Cannot obtain Locale/Timezone information for your client. Assuming GMT	10/20/10
1439723	Integration Definitions cache WSDL and are not updated when the underlying WSDL changes in the WS Connector	10/20/10
1439836	Improve performance by collapsing system lane activities	10/20/10
1439803	JSP Page that displays a JVM Thread Dump	10/20/10
1439709	The Process Server needs to be restarted after a failed import	10/20/10
1439557	Changes to filter definitions are not reflected in the report	10/20/10
1448203	How do you setup Single Sign-On (Kerberos) on WebSphere Lombardi Edition 7.1 on Windows?	10/04/10
1448216	How to Change the Default User Passwords (including tw_admin) on WebSphere Lombardi Edition	09/30/10
1446263	CWSIA0241E and PersistenceException- Known Problems with Database username in WebSphere Lombardi Edition 7.1.0	09/30/10
1448203	How do you setup Single Sign-On (Kerberos) on WebSphere Lombardi Edition 7.1 on Windows?	10/04/10
1446648	How to tune your WebSphere Lombardi Edition Runtime Server(s) for performance	09/29/10
1448251	WebSphere Lombardi Edition 7.1 Port information	09/29/10
1445844	How do you remove the server from the list of servers when you do not need it anymore?	09/28/10
1448045	Outdated favorites are not deleted correctly when you import a snapshot that does not have favorites	09/27/10

1447593	Offline deployment will fail if you change tw_admin password	09/27/10
1445620	Detailed steps for moving a runtime server from online to offline and back to online	09/02/10
1439657	What's the difference between the functional_task_id, task_id, system_task_id and system_functional_task_id?	08/30/10
1445237	DB2 database credentials following Simple Install of WebSphere Lombardi Edition 7.1.0	08/29/10
1441741	Imported Toolkits are immutable	08/21/10
1444016	WebSphere Lombardi Edition v7.1 – Configuring a Data Source	08/21/10
1444296	How to use SOAPUI to test WebAPI operations in Teamworks and WebSphere Lombardi Edition	08/18/10
1444180	Deploying and redeploying applications in WebSphere Lombardi Edition 7.1	08/18/10
1443840	DB2 licensing with WebSphere Lombardi Edition 7.1	08/16/10
1439614	Teamworks Config File Overview and Explanation	08/09/10
1439654	How is the "at Risk" status of a task is calculated	07/23/10
1439872	Change namespace in Teamworks services exposed as Web Services	07/20/10
1439791	What is the sort order for Saved Searches - can I change them?	07/08/10
1440235	How do you change the tw_admin password in Teamworks 7.0.x?	06/01/10
1439858	How to determine if Performance Server definitions loaded correctly	05/03/10

Requirements and Enhancements

From time to time while using IBM BPM, you may find some feature that doesn't quite work the way you would like it to or some feature that you think is missing. IBM is extremely interested in hearing about this. An IBM website has been created where "Request for Feature Enhancements" can be logged, tracked and monitored. The web site is:

<http://www.ibm.com/developerworks/rfe/>

and provides easy to follow instructions for recording your enhancement requests and viewing those submitted by others.



developerWorks > RFE Community >

IBM RFE Community



Welcome RFE Community users! Here you have an opportunity to collaborate directly with the IBM product development teams and other product users.

- [Search for RFEs](#) (view, comment, vote, and watch)
- [Submit RFEs](#)
- [Track your RFEs](#) (My RFEs)

RFE activities for all brands

→ [Search RFEs](#)

Before you submit a request, search the existing database to minimize the number of duplicate entries. You can add your own comments to existing requests and share information across the community.

→ [Submit RFEs](#)

Submit your own idea for a new RFE. Our development team will review your request and provide status updates throughout its lifecycle.

[Browse RFEs](#)

Spotlight

- [Announcements](#)
- [Give us your feedback](#)

Brands

- [All brands](#)
- [Information Management](#)
- [Rational](#)
- [Tivoli](#)
- [WebSphere](#)

Product Installation and Configuration

Installation of IBPM is very straightforward. An installer application is provided and very few questions are asked. This is known as standard installation. However, despite what I just wrote, experience is showing that Standard installation can be very problematic unless the environment on which the installation is to be performed is about as basic/virgin as it can possibly be. Instead of executing the Standard installation, it is recommended to perform the Advanced installation. In addition to following the instructions for an Advanced install, it is recommended to create a detailed log of the steps followed. This means creating a word processor document (eg. Apache Open Office or Microsoft Word) and logging all the steps performed. Ideally, screen shots of each of the items should also be captured. Any inputs asked for should also have their value recorded. If all goes well, it can be argued that the creation of this document was a waste of time, however, in the event that something goes “south”, this document can become extremely valuable in diagnosing the problems. In addition, if a later review or question comes up over what was entered or selected during installation or configuration, the document will again prove to be a useful guide.

When installing on Windows 7, it is vital to perform the install as an administrator user.

See also:

- DeveloperWorks - [Configuring the IBM Business Process Manager V7.5 environment for a typical installation](#) - 2011-12-14

Part Numbers

The part number for the current products as seen on IBM Passport Advantage are:

Business Process Manager

Windows	
IBM Business Process Manager Advanced 8.5 Windows eAssembly	CRMM2ML
IBM Business Process Manager Advanced 8.5 Windows 1 of 3	CIL8YML
IBM Business Process Manager Advanced 8.5 Windows 2 of 3	CIL8ZML
IBM Business Process Manager Advanced 8.5 Windows 3 of 3	CIL90ML
Linux	
IBM Business Process Manager Advanced 8.5 Linux eAssembly	CRMM5ML
IBM Business Process Manager Advanced 8.5 Linux 1 of 3	CIL96ML
IBM Business Process Manager Advanced 8.5 Linux 2 of 3	CIL97ML
IBM Business Process Manager Advanced 8.5 Linux 3 of 3	CIL98ML

Integration Designer

IBM Integration Designer 8.0 Multiplatform eAssembly	CRIG4ML
IBM Integration Designer Version 8.0 Multiplatform Quick Start Guide	CI824ML
IBM Integration Designer Version 8.0 Windows and Linux 1 of 3	CI825ML
IBM Integration Designer Version 8.0 Windows and Linux 2 of 3	CI826ML
IBM Integration Designer Version 8.0 Windows and Linux 3 of 3	CI827ML
IBM Integration Designer Version 8.0 test environment Windows 1 of 5	CI828ML
IBM Integration Designer Version 8.0 test environment Windows 2 of 5	CI829ML
IBM Integration Designer Version 8.0 test environment Windows 3 of 5	CI82AML

IBM Integration Designer Version 8.0 test environment Windows 4 of 5	CI82BML
IBM Integration Designer Version 8.0 test environment Windows 5 of 5	CI82CML

Process Server

Windows	
IBM Process Server Advanced 7.5 for Windows eAssembly	CREX6ML
IBM Process Server Advanced 7.5 for Windows 1 of 3	CIOBKML
IBM Process Server Advanced 7.5 for Windows 2 of 3	CIOBLML
IBM Process Server Advanced 7.5 for Windows 3 of 3	CIOBMML
Linux	
IBM Process Server Advanced 7.5 for Windows eAssembly	CREX9ML
IBM Process Server Advanced 7.5 for Linux 1 of 3	CI0BSML
IBM Process Server Advanced 7.5 for Linux 2 of 3	CI0BTML
IBM Process Server Advanced 7.5 for Linux 3 of 3	CI0BUML

If you are installing a full DB2, the part numbers for this are:

IBM DB2 9.7 Enterprise Server Edition 32 Bit	CR8NVML
IBM DB2 9.7 Enterprise Server Edition 64 Bit	CR8NWML

See also:

- [Download IBM Business Process Manager Advanced Version 8.5 \(Windows\)](#) – 8.5 – 2013-06-14
- [Download IBM Business Process Manager Standard Version 8.5 \(Windows\)](#) – 8.5 - 2013-06-14

Prerequisites

The official prerequisites for IBPM 7.5 can be found here:

[Detailed hardware and software requirements for IBM Business Process Manager Standard V7.5](#)

[Detailed hardware and software requirements for IBM Business Process Manager Advanced V7.5](#)

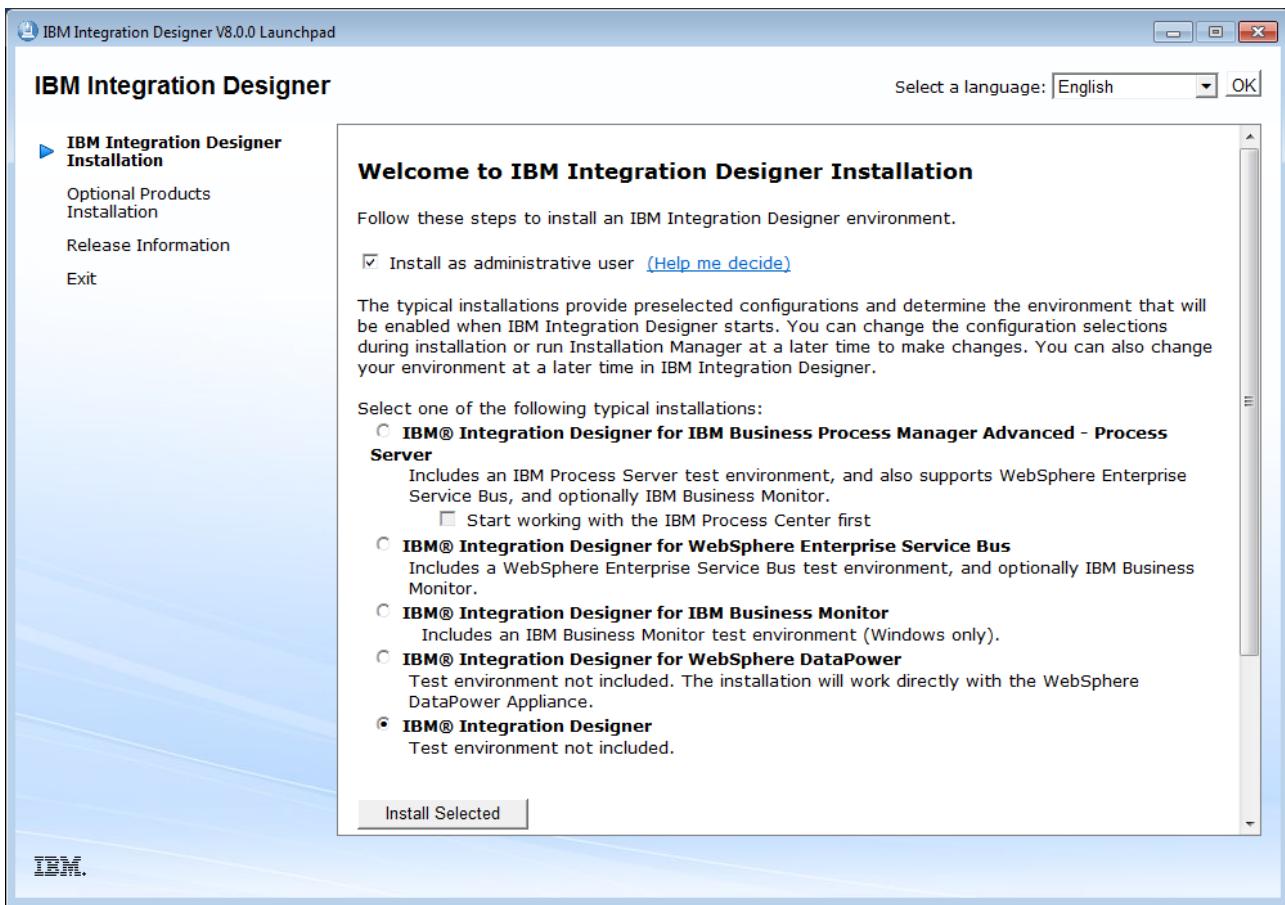
Installing Integration Designer

Integration Designer is the development tooling for building SCA, BPEL and related components. It is an Eclipse based set of tools as well as a local copy of a Process Server for performing stand-alone based unit testing.

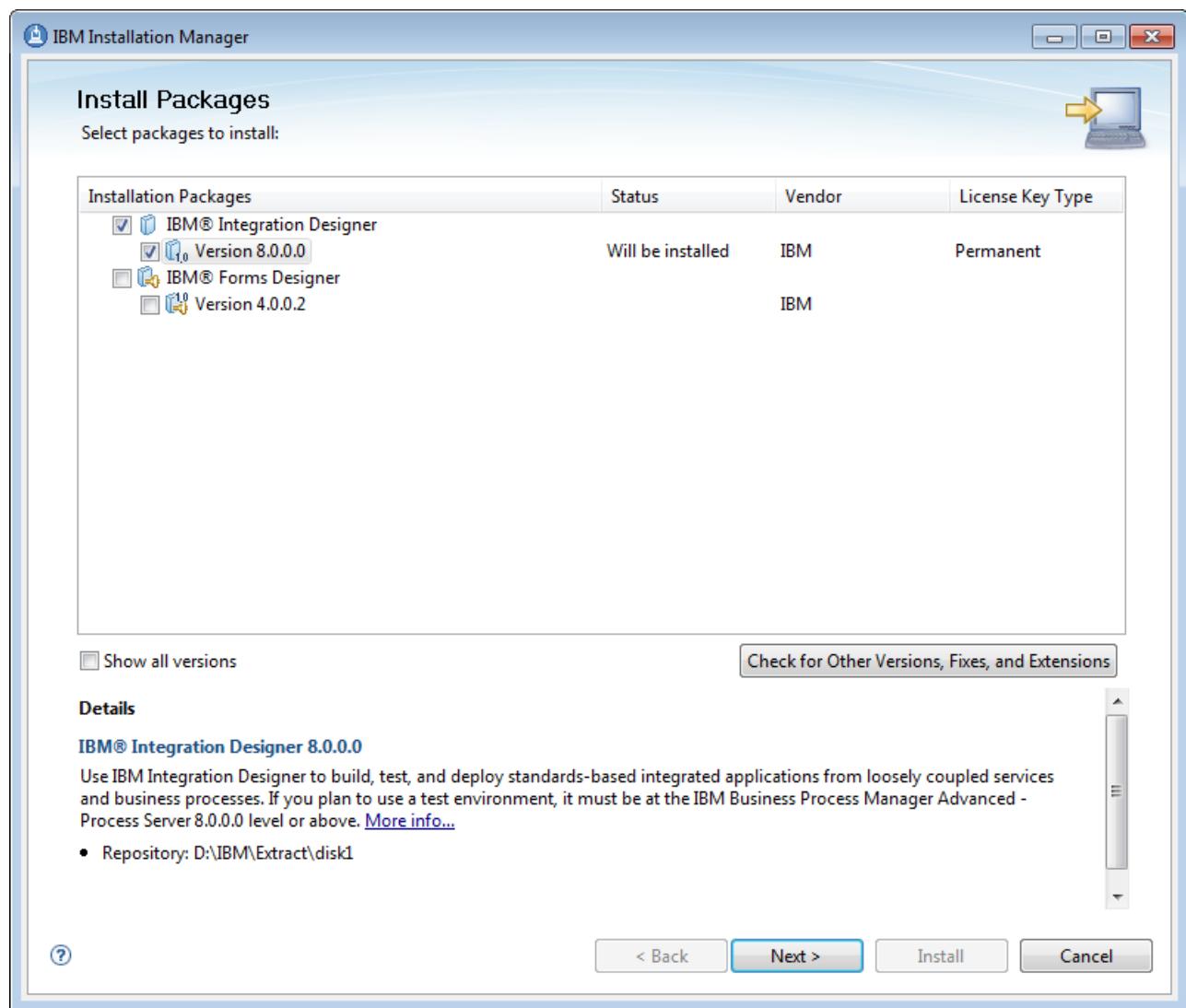
Experience has shown that there are problems installing ID from a Windows share.

To install Integration Designer and related components, follow these instructions:

Run `launchpad64`. This brings up an initial wizard.



This then launches and installs the IBM Installation Manager software.



IBM Installation Manager

Install Packages

Read the following license agreements carefully.

International Program License Agreement

Part 1 - General Terms

BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, CLICKING ON AN "ACCEPT" BUTTON, OR OTHERWISE USING THE PROGRAM, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF LICENSEE, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND LICENSEE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,

* DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, CLICK ON AN "ACCEPT" BUTTON, OR USE THE PROGRAM; AND

* PROMPTLY RETURN THE UNUSED MEDIA, DOCUMENTATION, AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM IT WAS OBTAINED FOR A REFUND OF THE AMOUNT PAID. IF THE PROGRAM WAS DOWNLOADED, DESTROY ALL COPIES OF THE PROGRAM.

1. Definitions

"Authorized Use" - the specified level at which Licensee is authorized to execute or run the Program. That level may be measured by number of users, millions of service units ("MSUs"), Processor Value Units ("PVUs"), or other level of use specified by IBM.

"IBM" - International Business Machines Corporation or one of its subsidiaries.

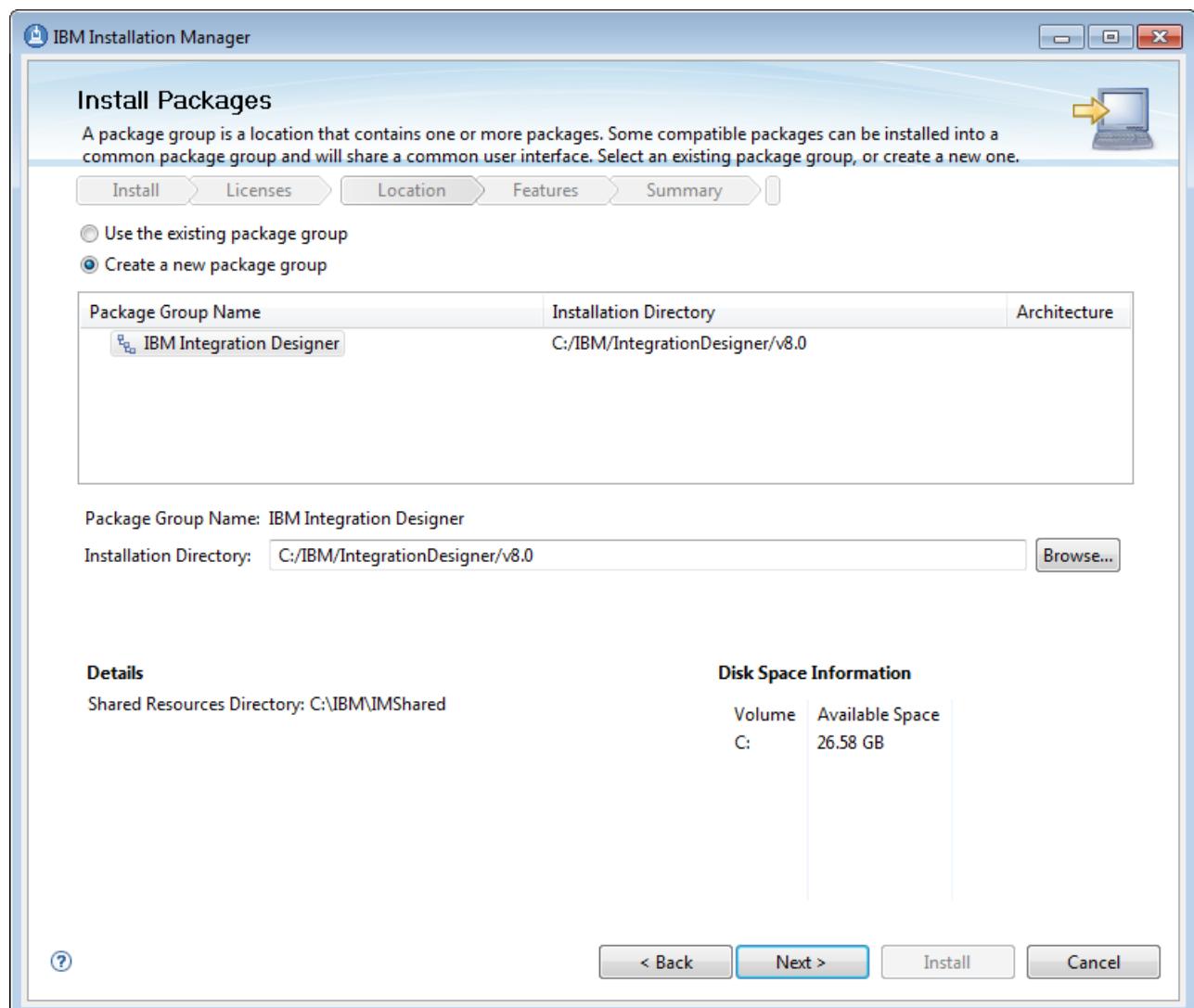
"License Information" ("LI") - a document that provides information and any additional terms specific to a Program. The Program's LI is available at www.ibm.com/software/sla. The LI can also be found in the Program's directory, by the use of a system command, or as a booklet included with the Program.

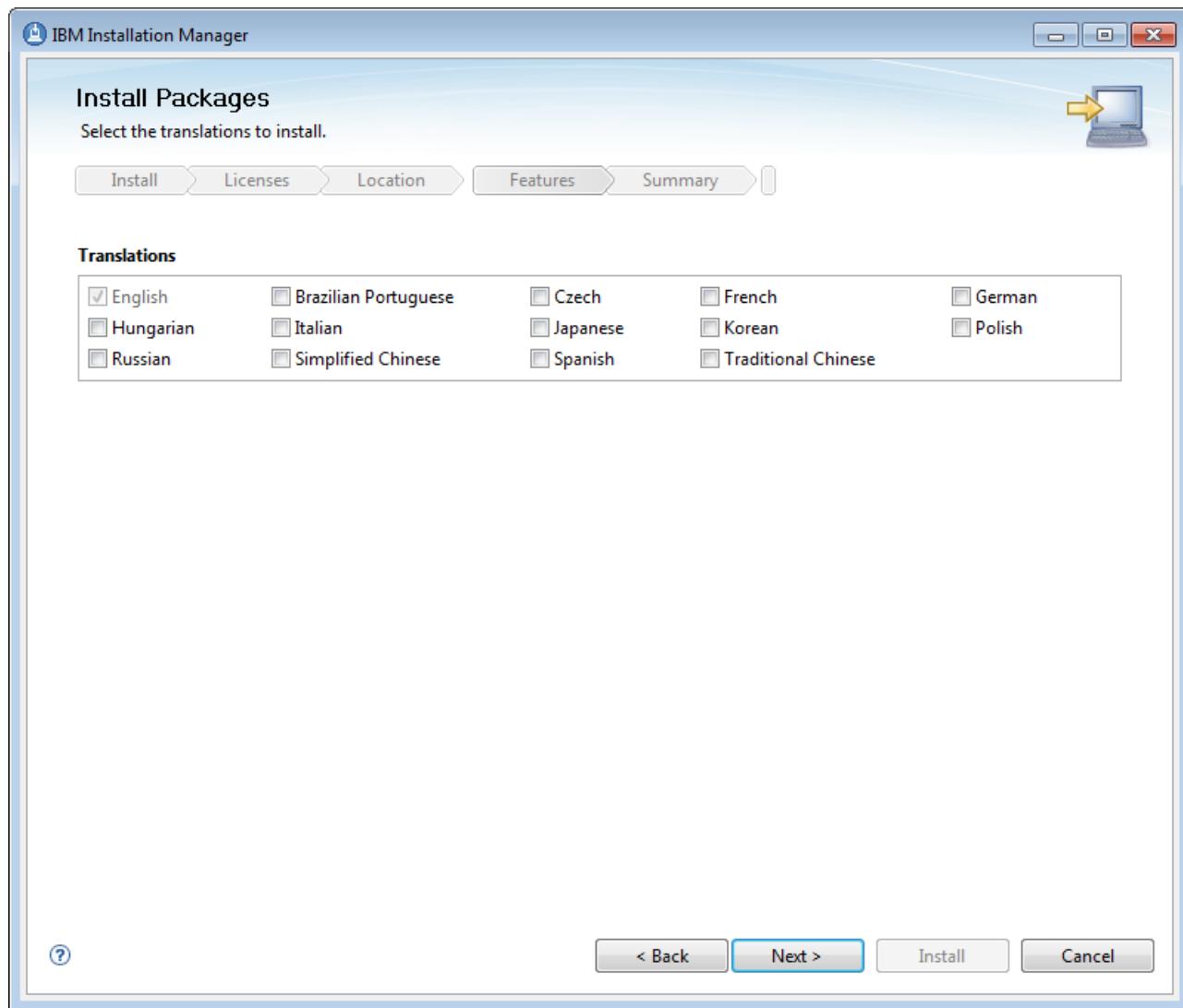
"Program" - the following, including the original and all whole or partial copies: 1) machine-readable instructions and data, 2)

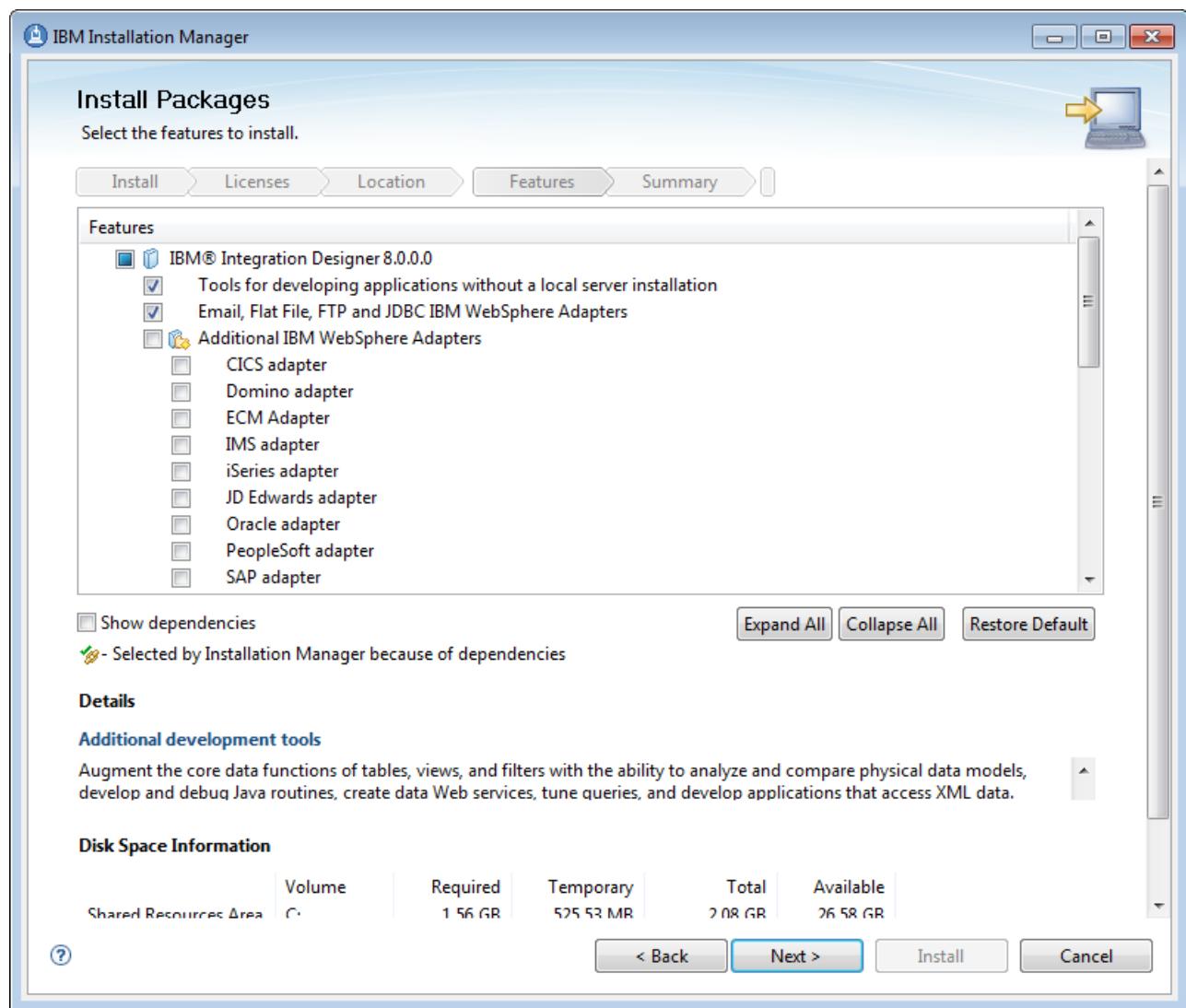
I accept the terms in the license agreement I do not accept the terms in the license agreement

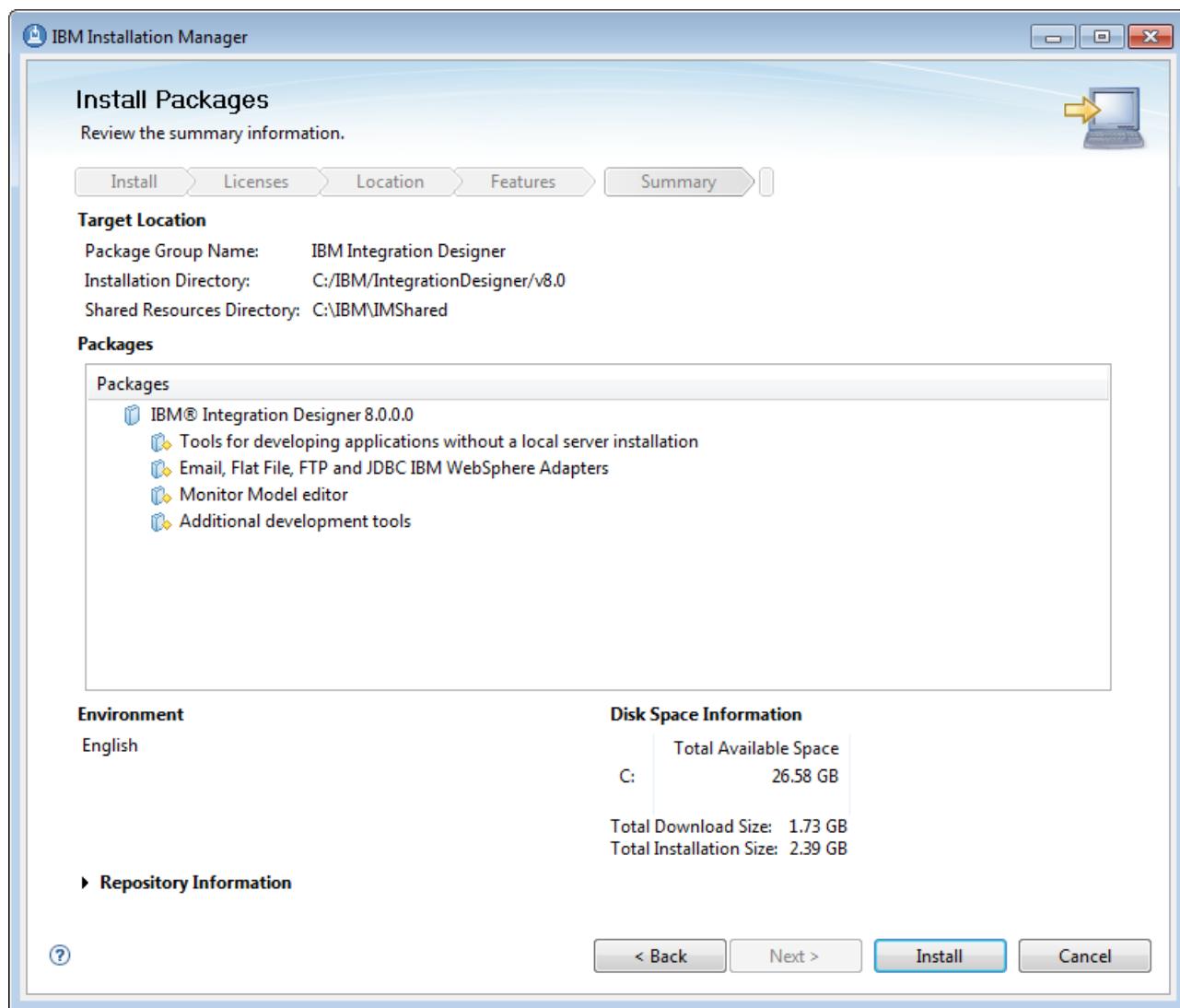
[Print All...](#)

< Back [Next >](#) [Install](#) [Cancel](#)





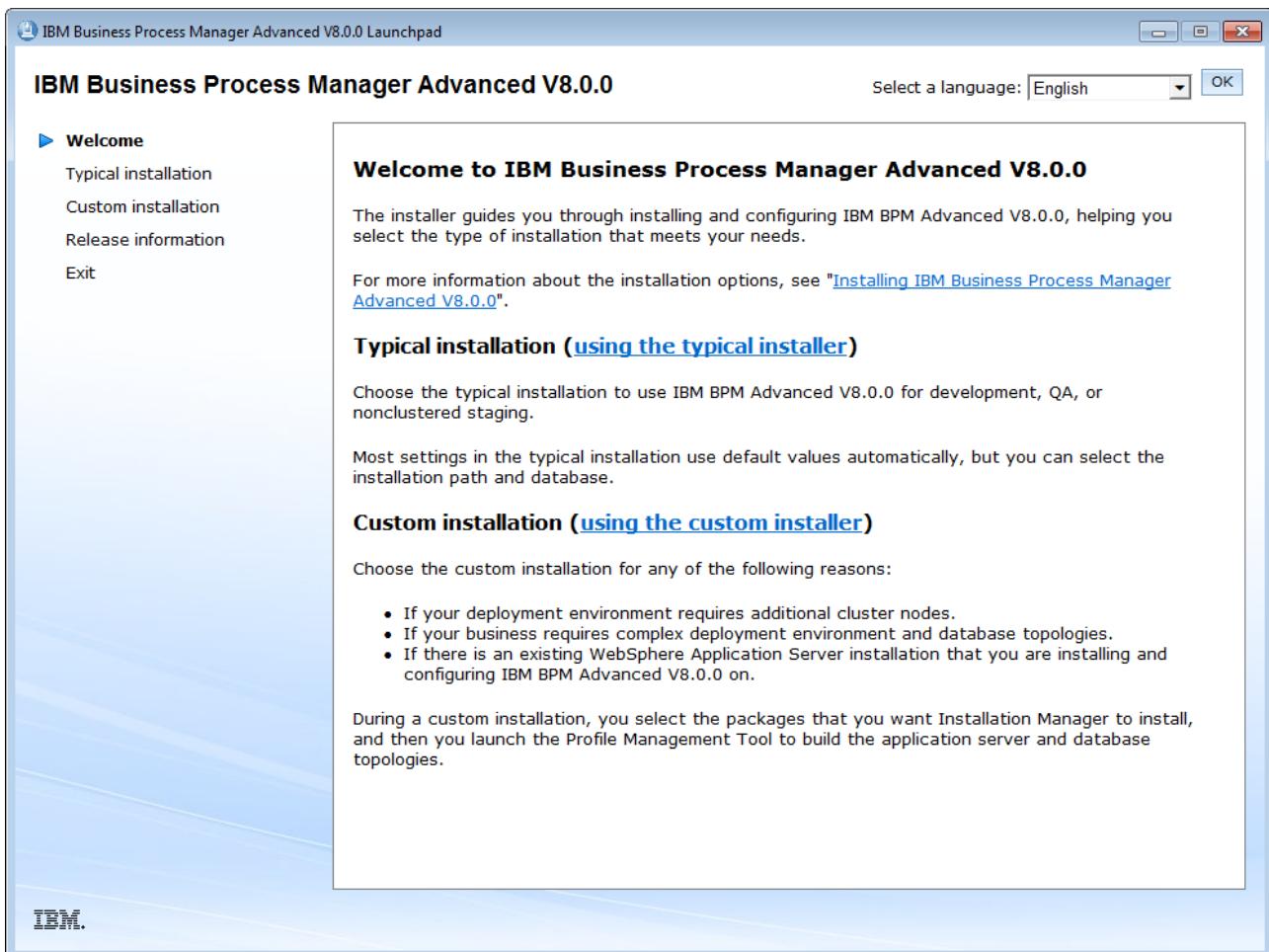


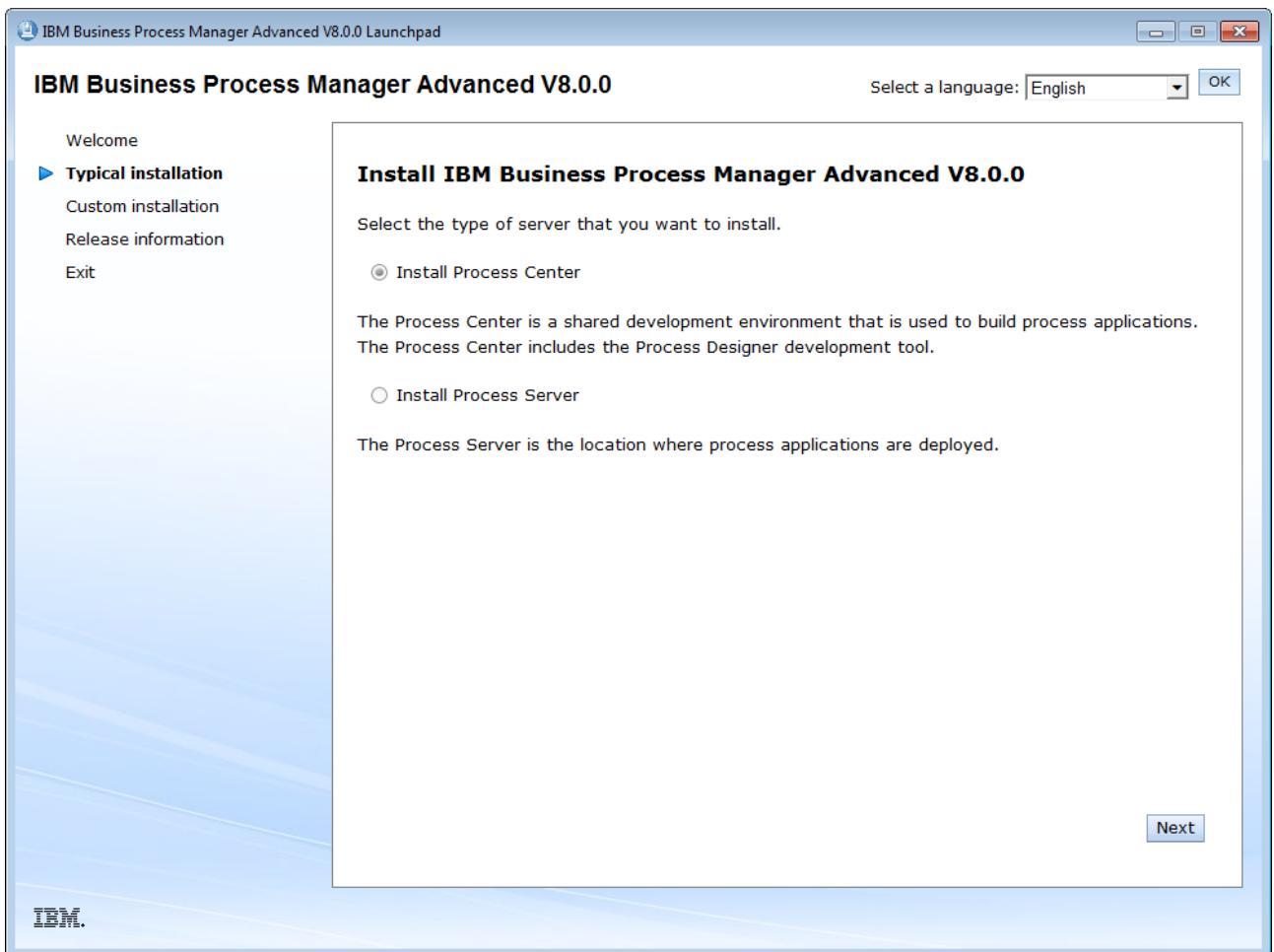


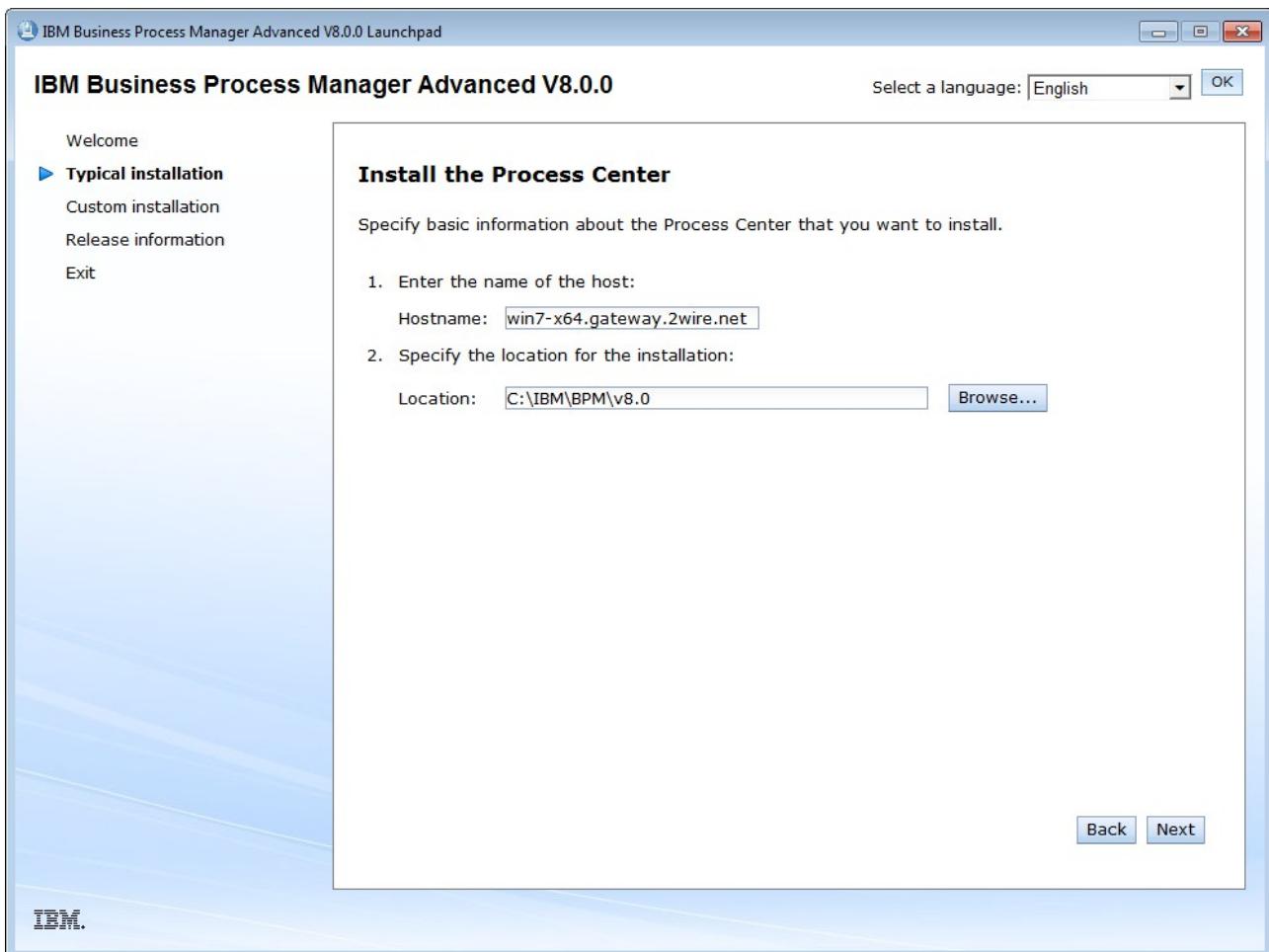
Installing Business Process Manager Advanced in typical mode

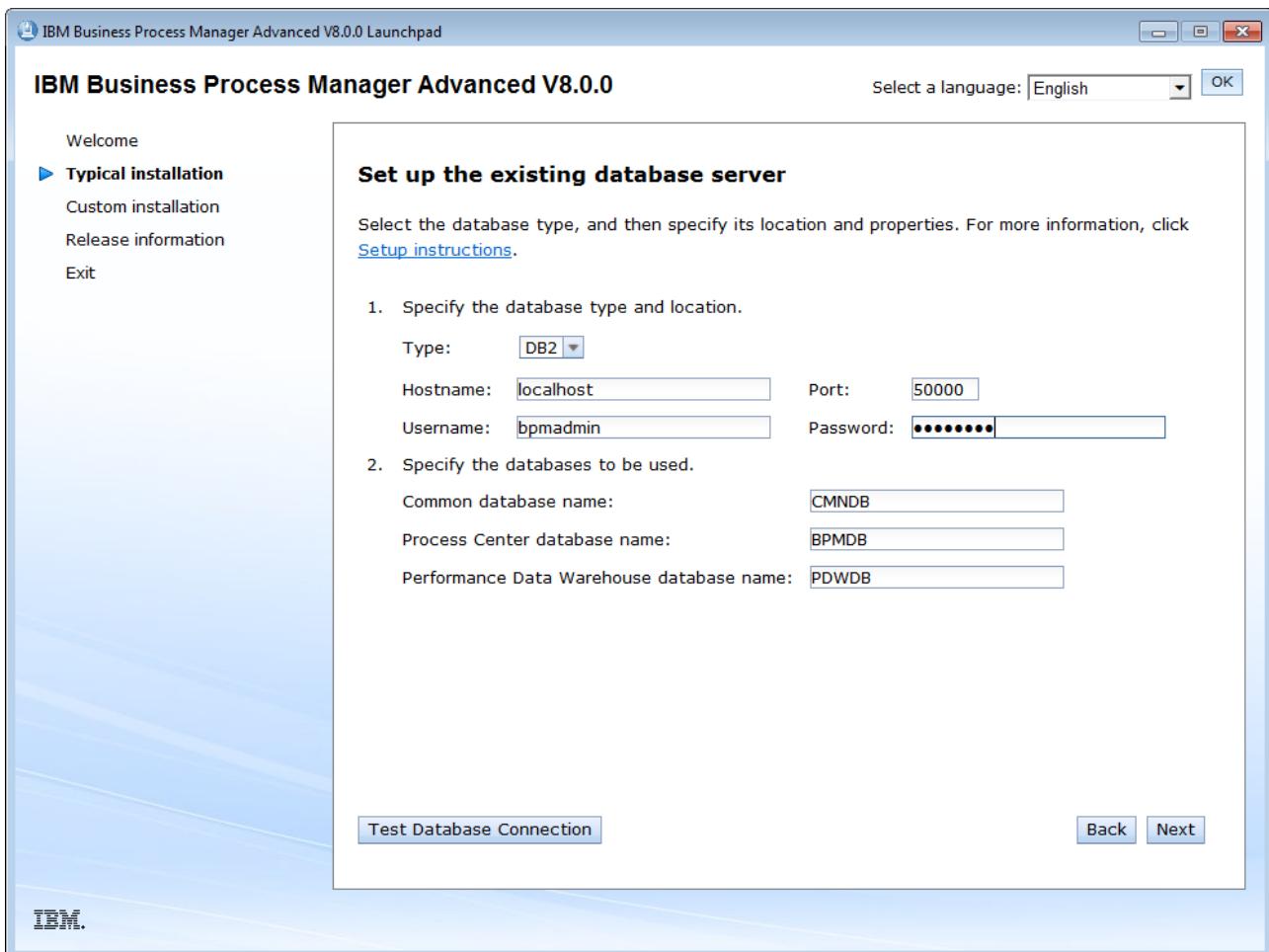
The *typical* install of Business Process Manager installs a WAS server, feature packs and then the Business Process Manager itself. After the mechanical installation has completed, a profile is created that is configured with Process Center. Personally, I prefer to run the custom install as it gives me much more control over the installation process and profile creation. It also allows me to detect problem areas that would otherwise leave a complete installation broken.

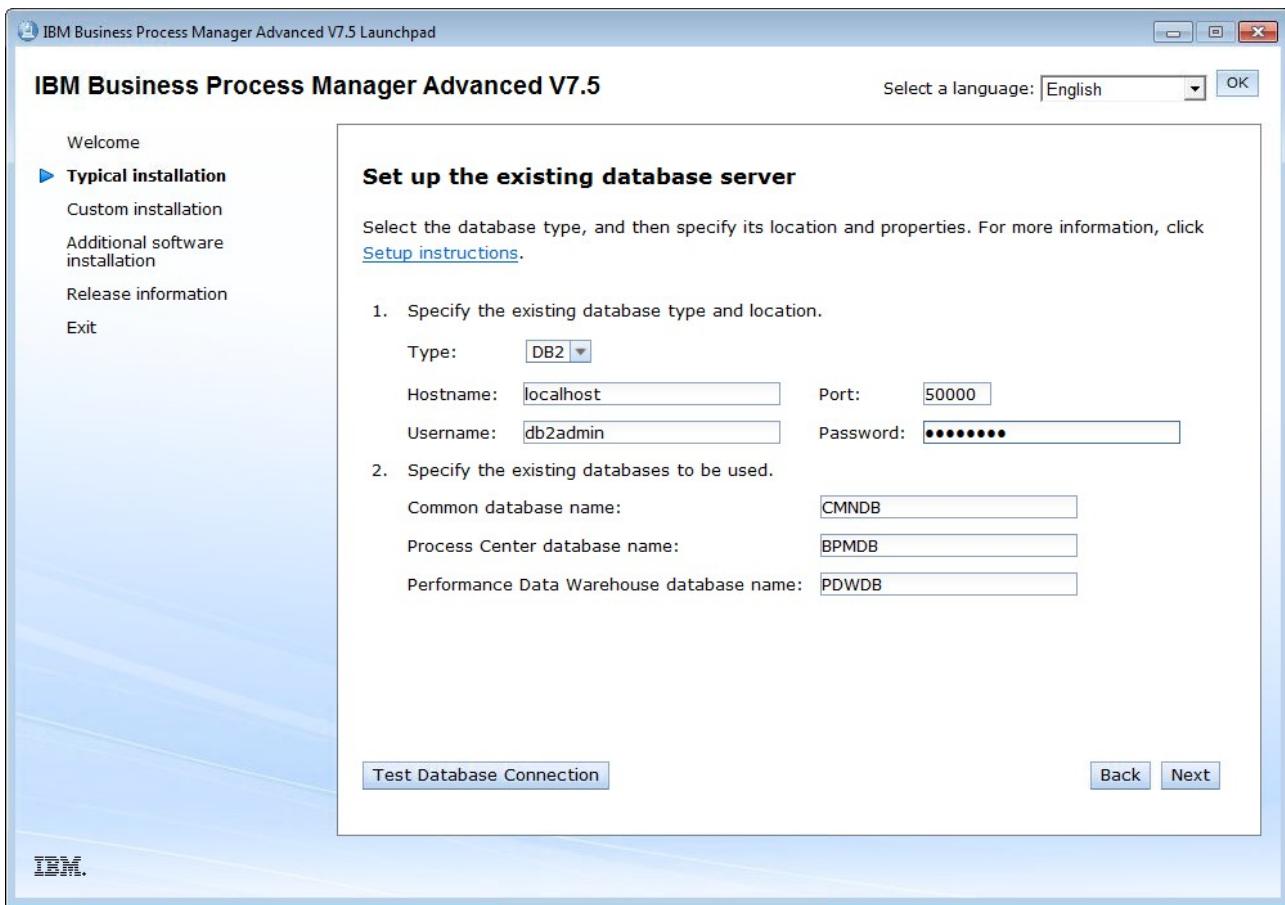
The installation is started by running the `Launchpad` command which asks whether this is a typical or custom install. The typical install runs IBM Installation Manager in silent mode. This means that you don't see what is happening in as much detail during the installation



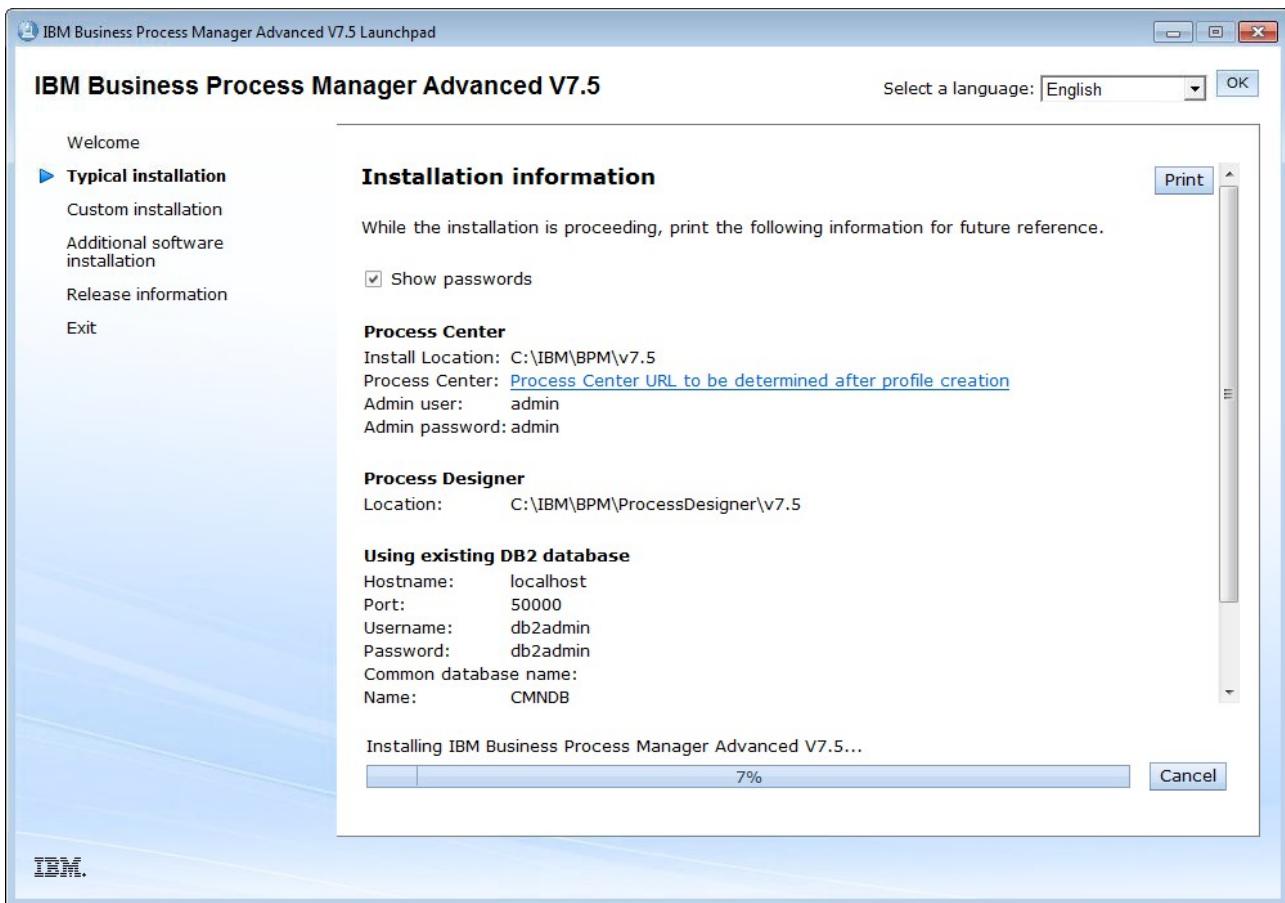








The installation now progresses with a silent version of Installation Manager.



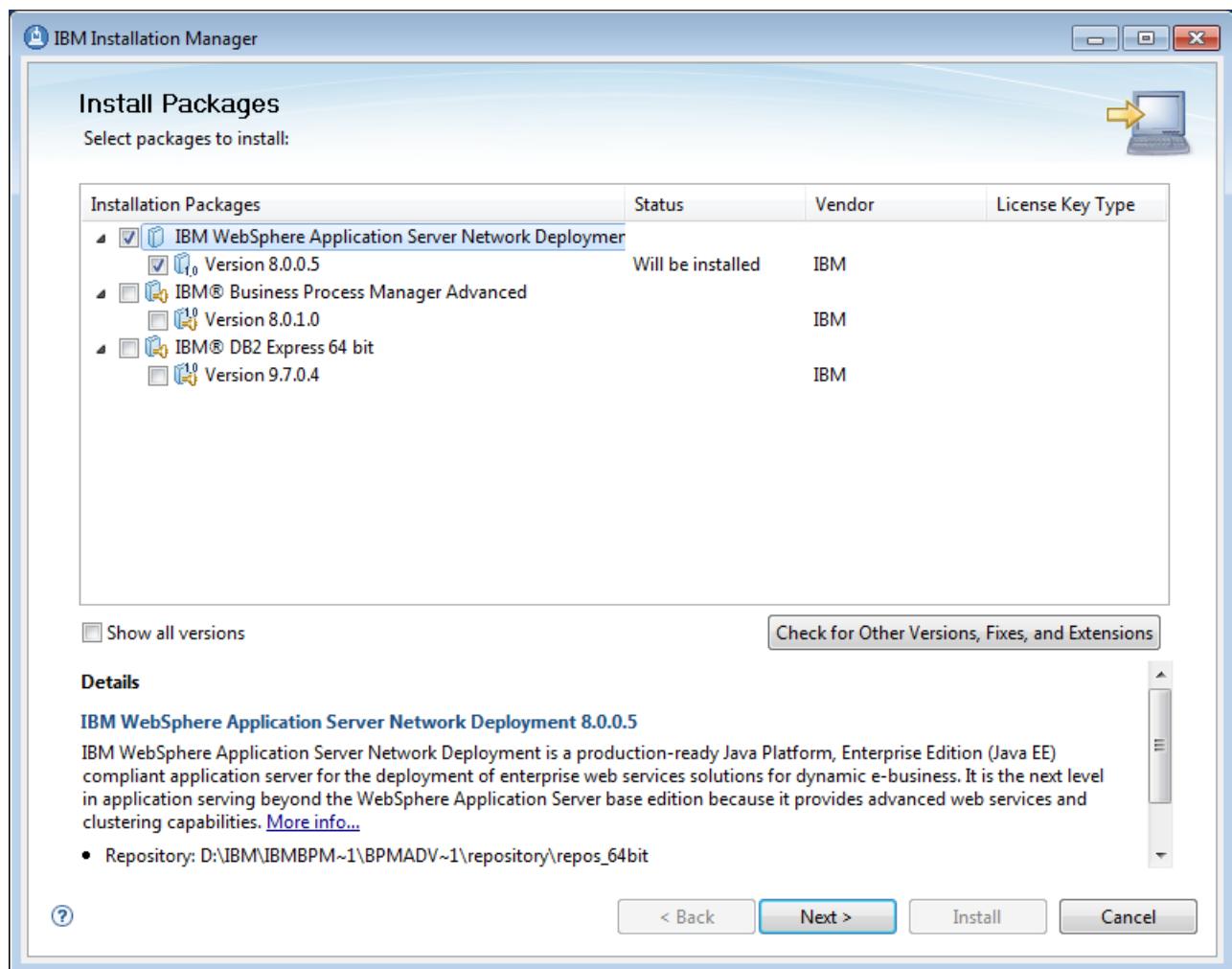
Performing a custom Install of Process Manager Advanced

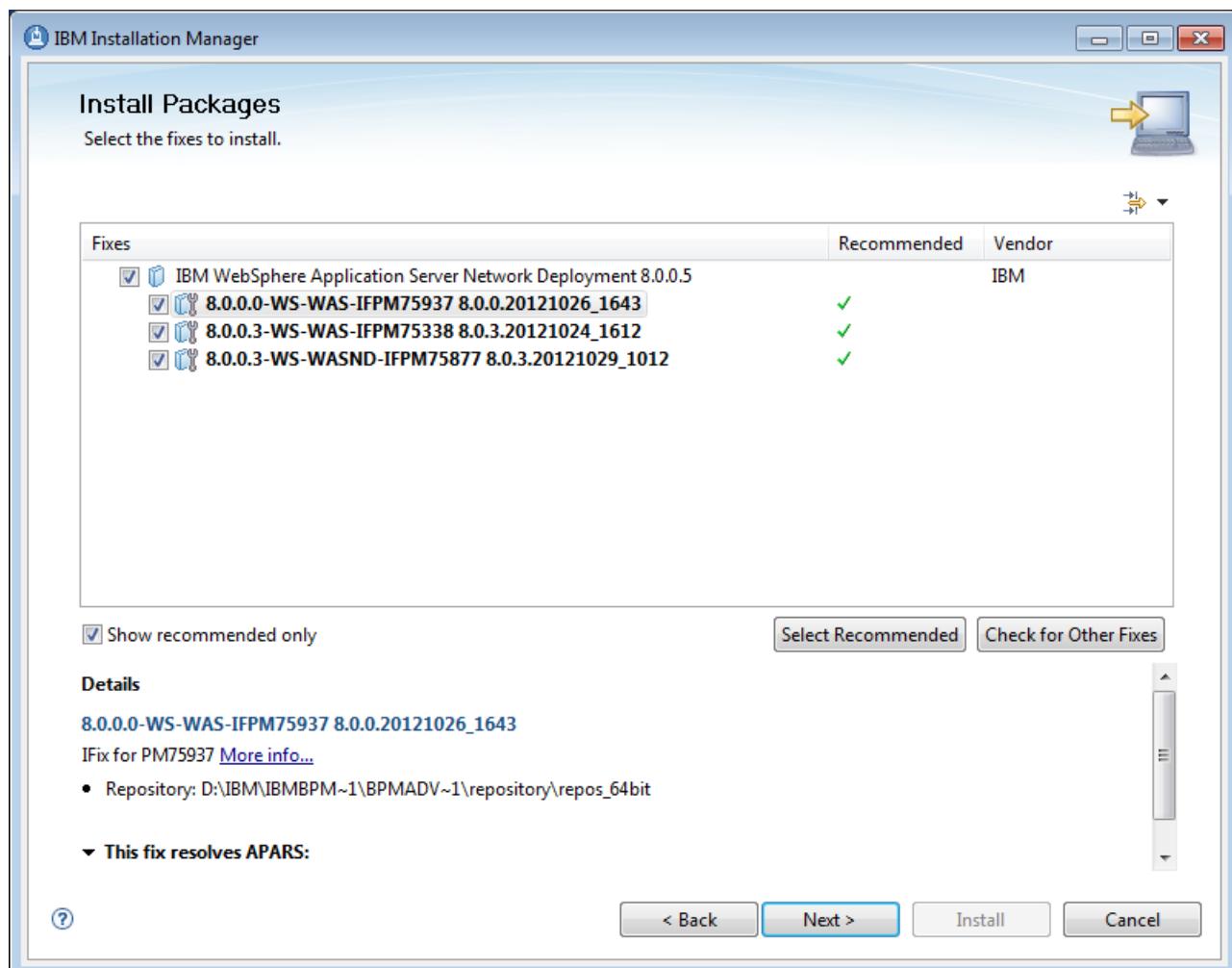
When a Custom Install is requested, IBM's Installation manager is launched and used explicitly. I think I like this technique better than one large and silent install. Personally, I prefer to install each component one piece at a time in the following order:

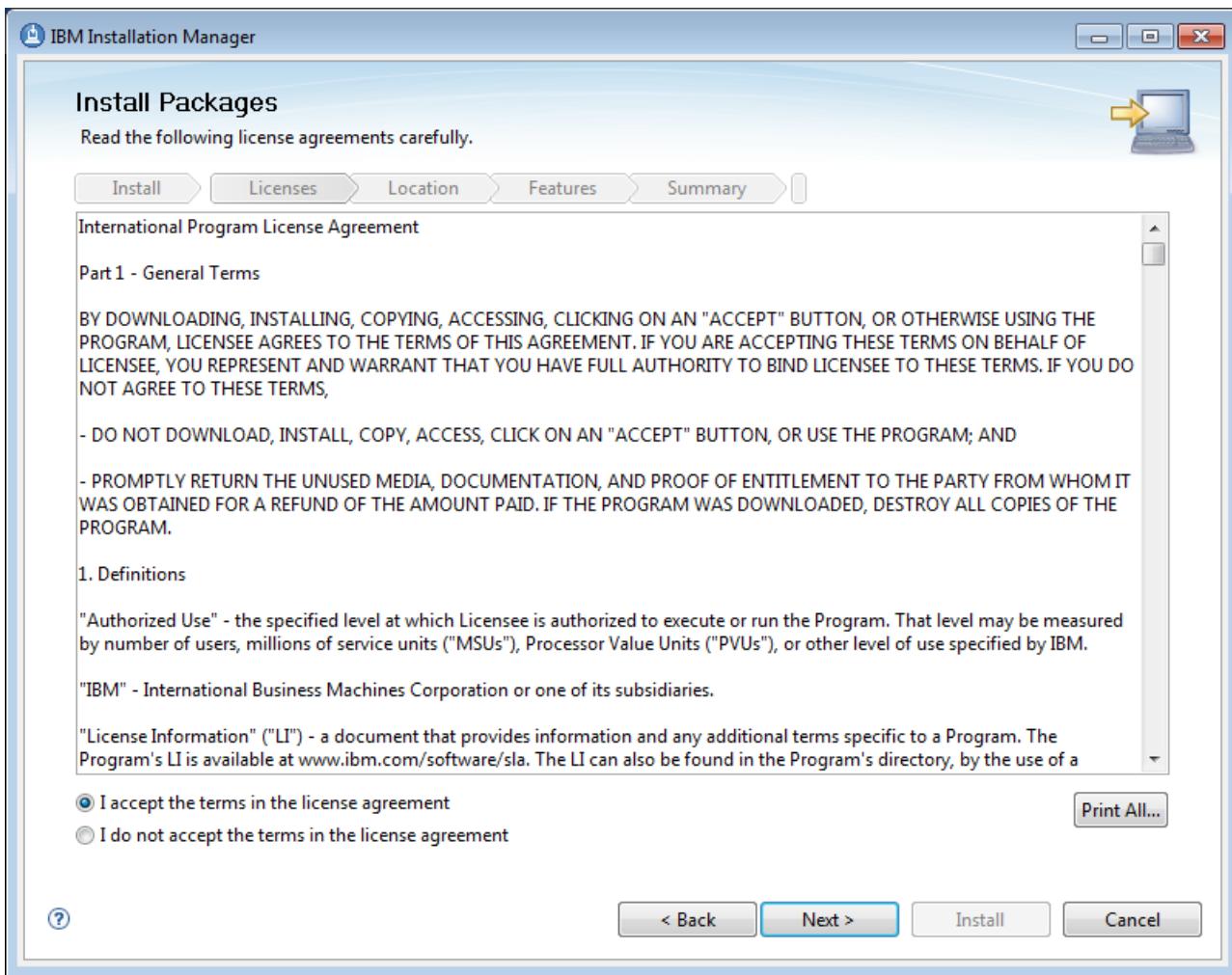
- WAS ND 8.0.0.5
- IBM BPM Advanced 8.0.1

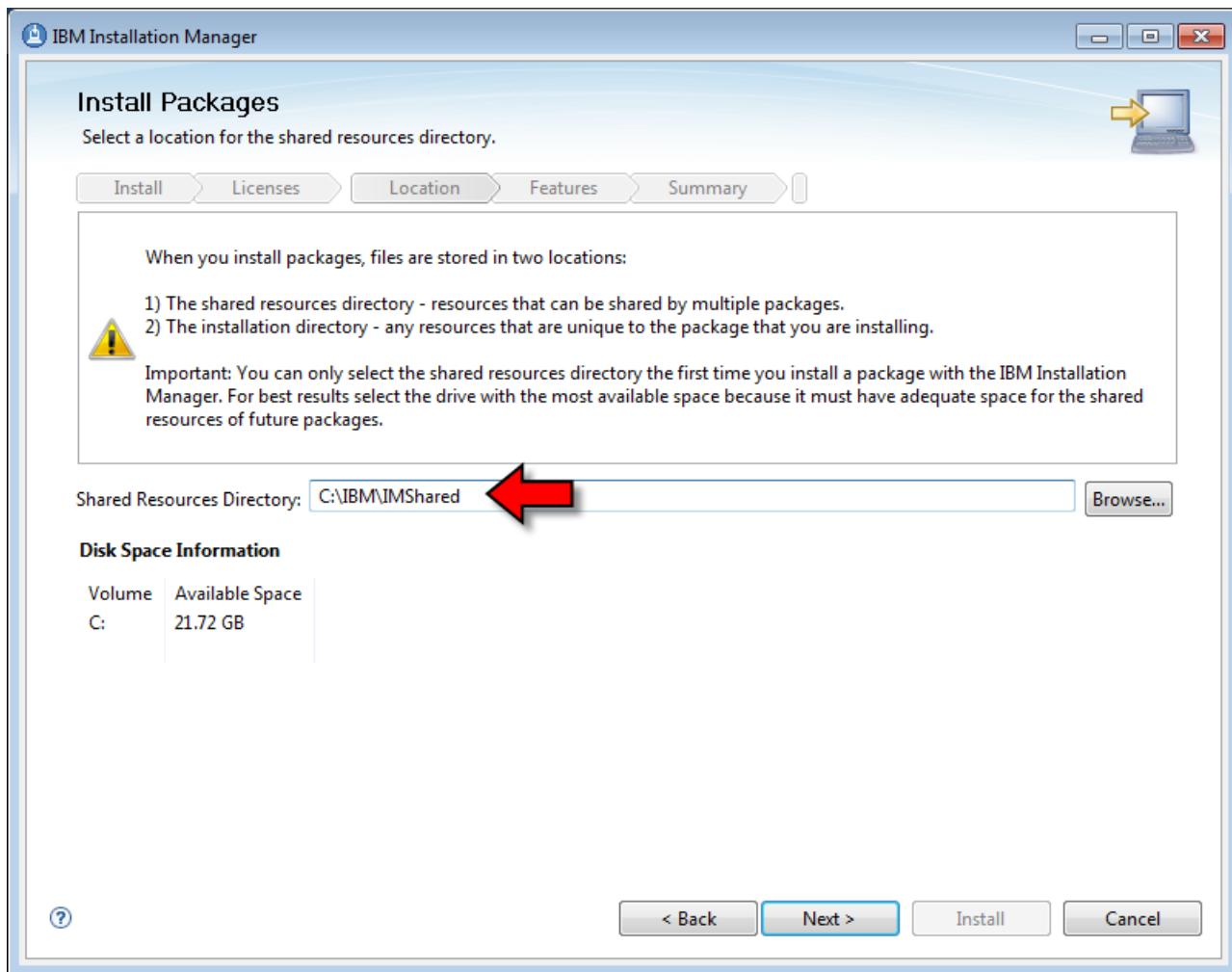
Only when the preceding component has successfully been installed, do I proceed to the next. In principle, I should be able to install all the packages in one shot but I seem to believe that if there is an installation problem with a later package that all the previously installed packages are also rolled back when a clean-up is performed. This is probably the correct semantics as at the end of the failed install, we are back to where we were before the install started but to my mind, this means re-installing those successful packages the next time the install is attempted.

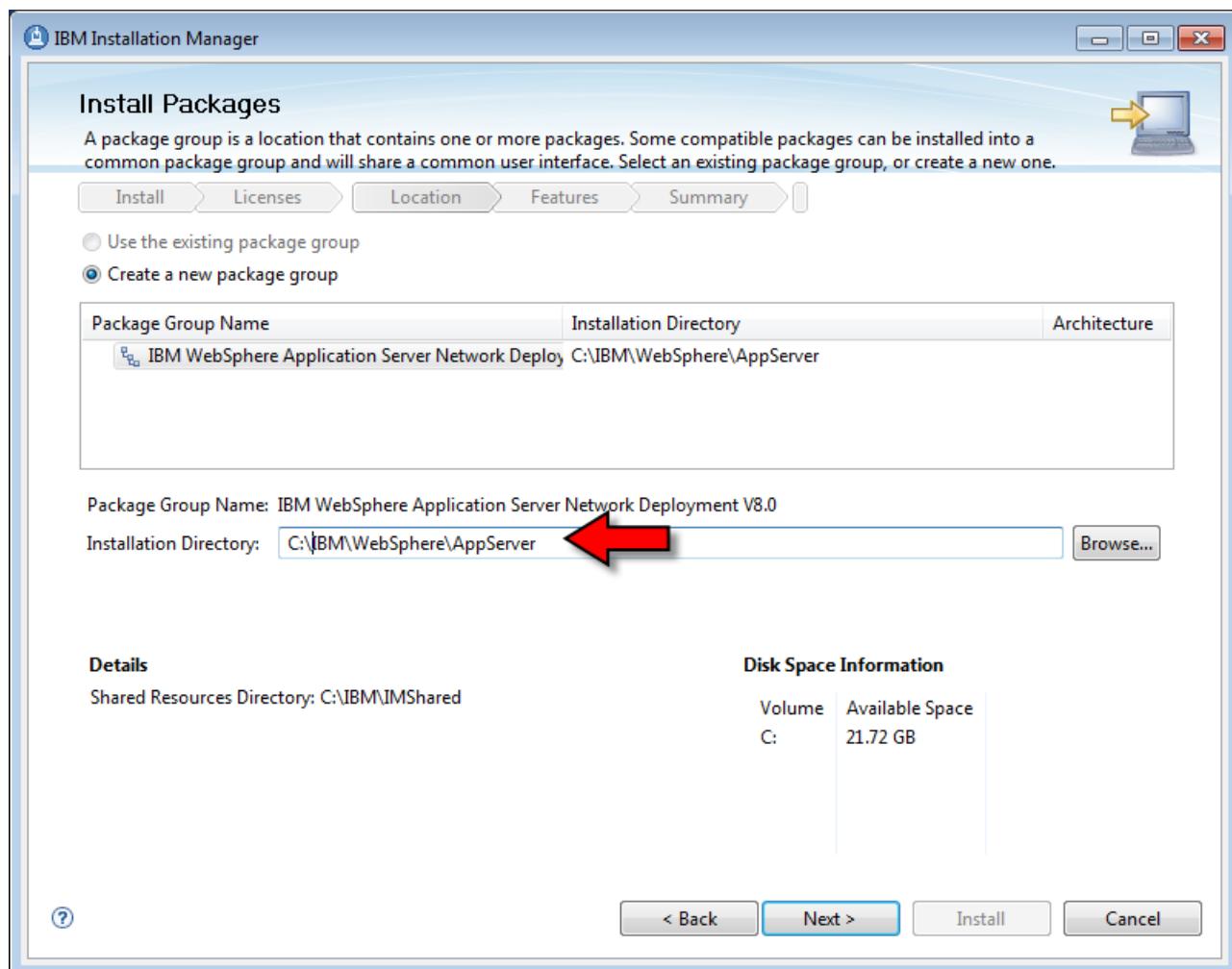
First I install based WebSphere Application Server ND

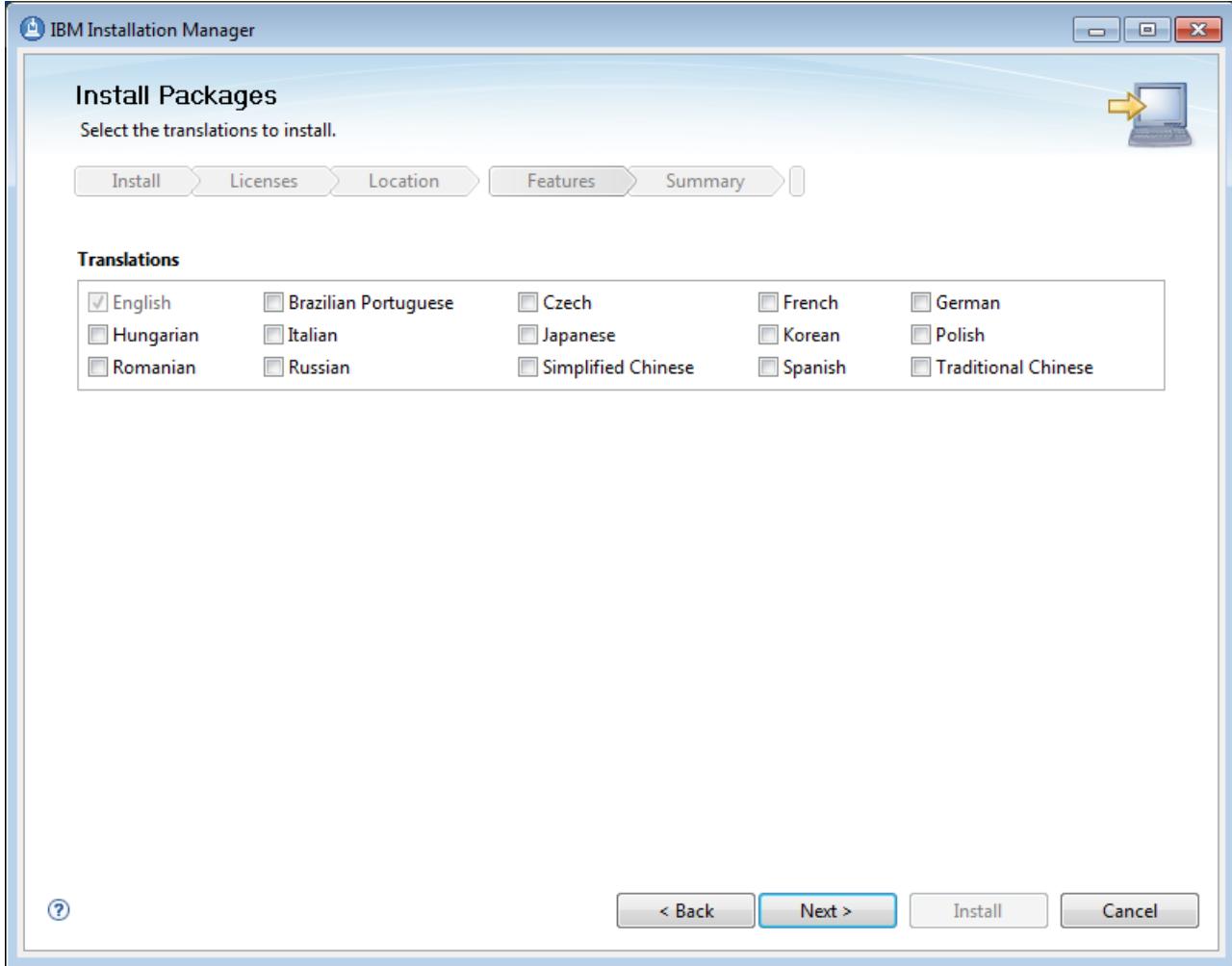


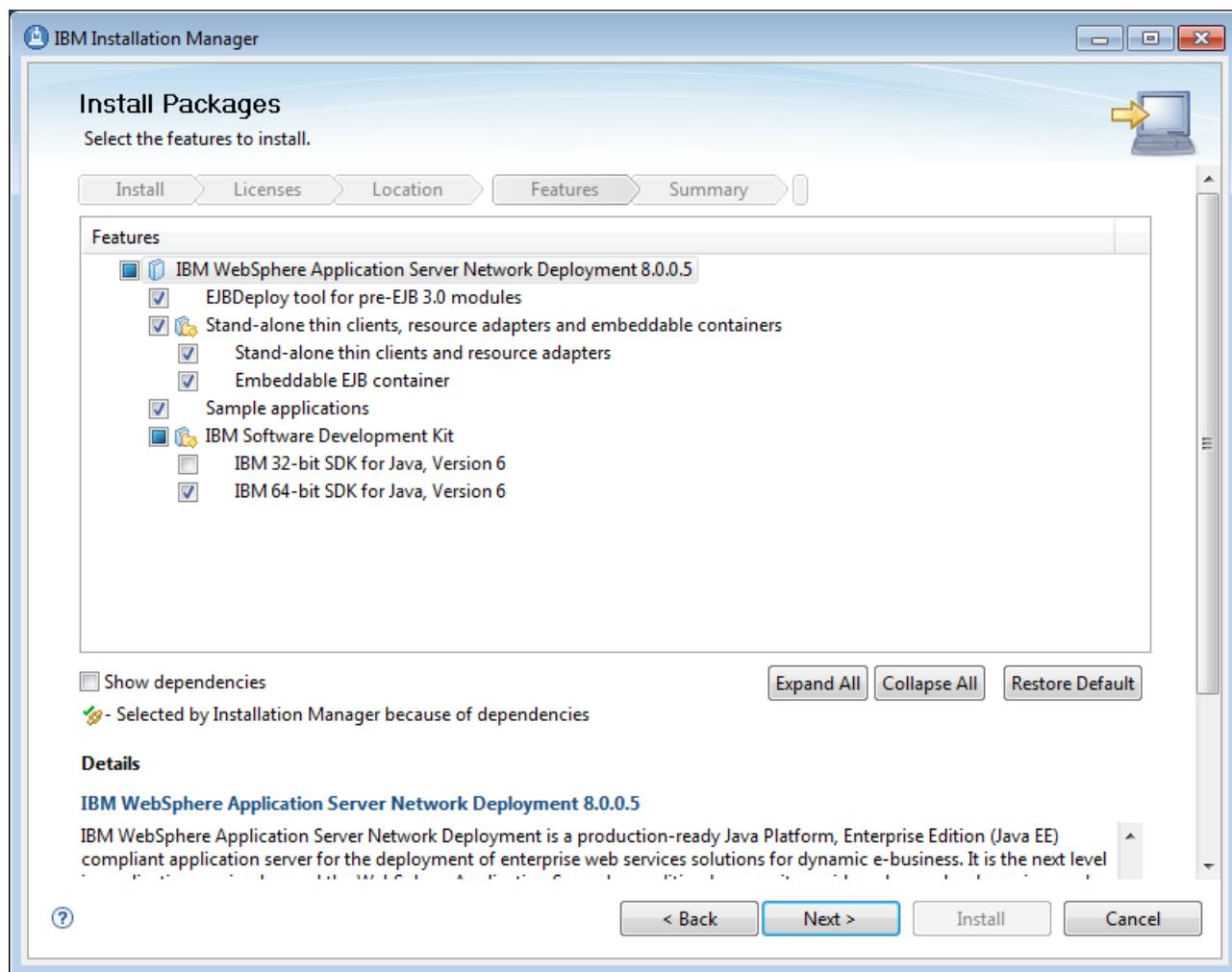


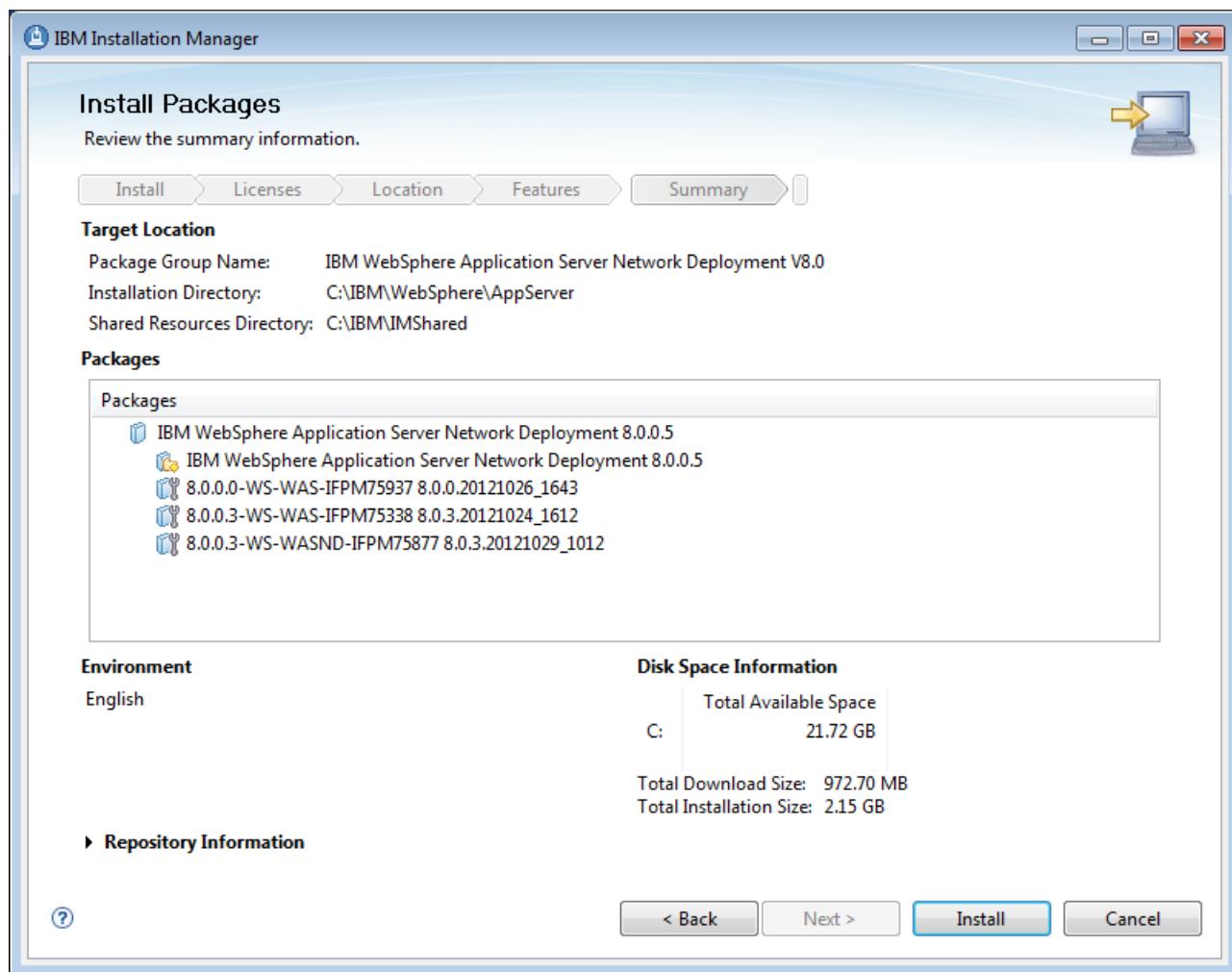


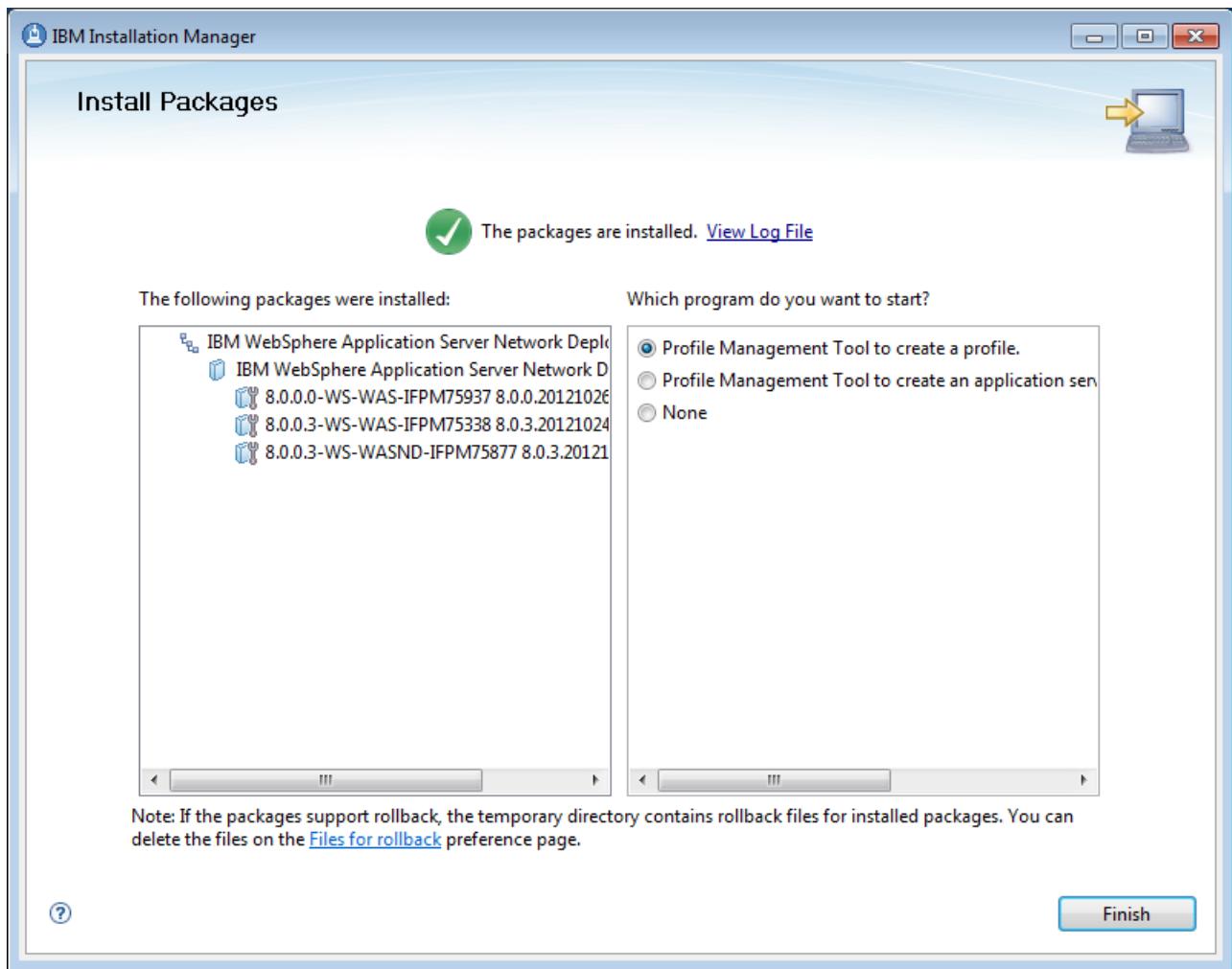




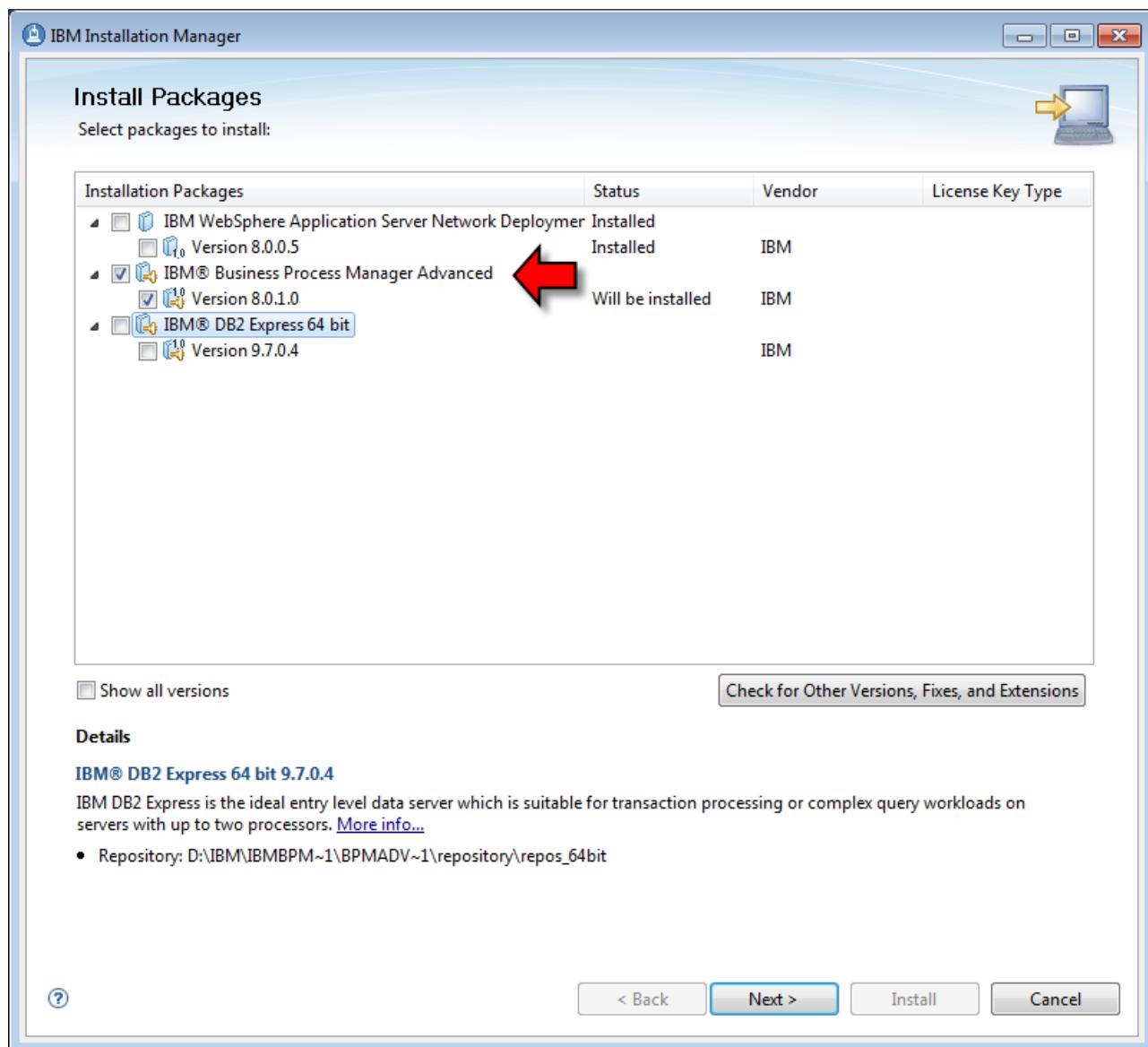


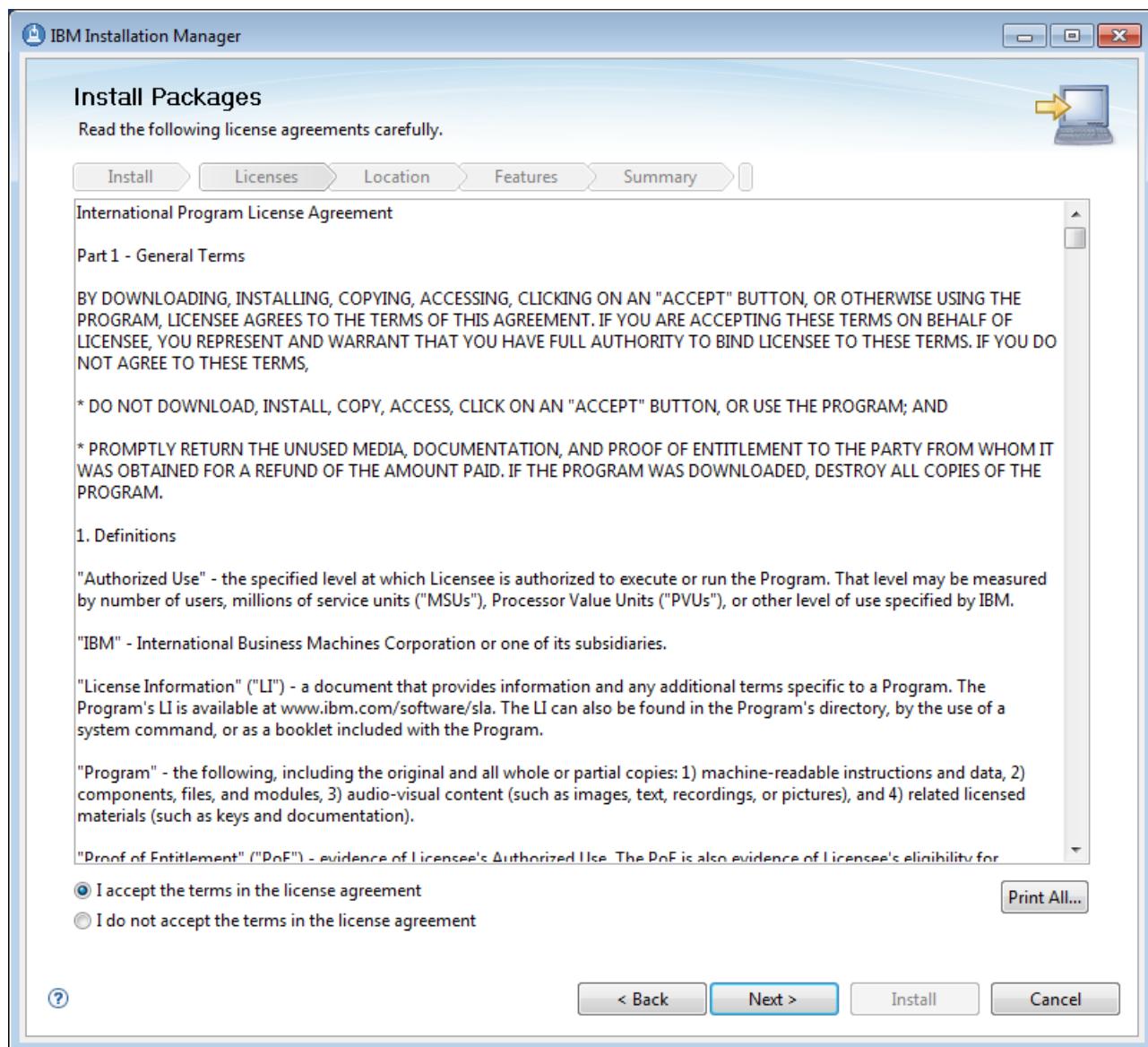


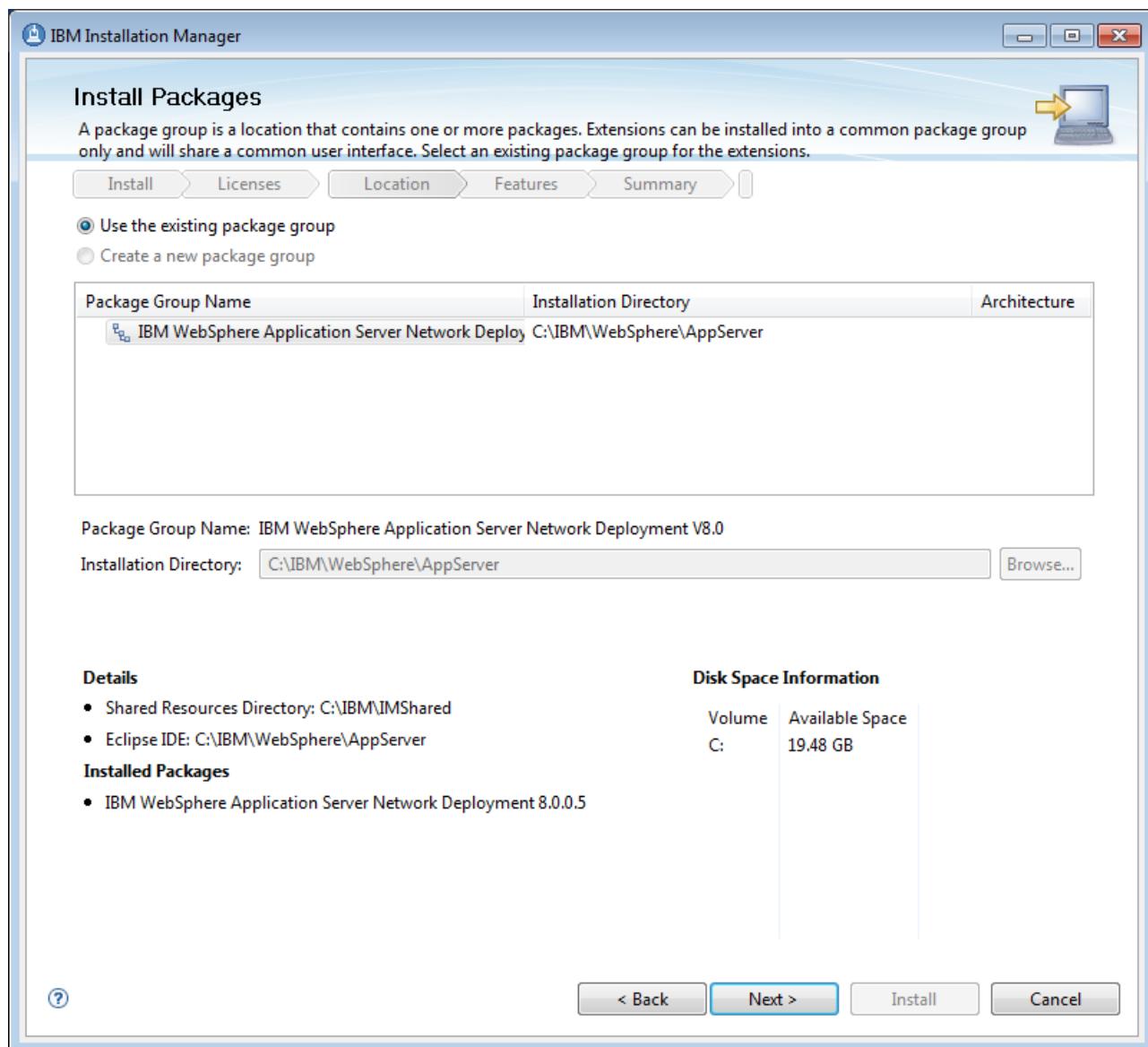


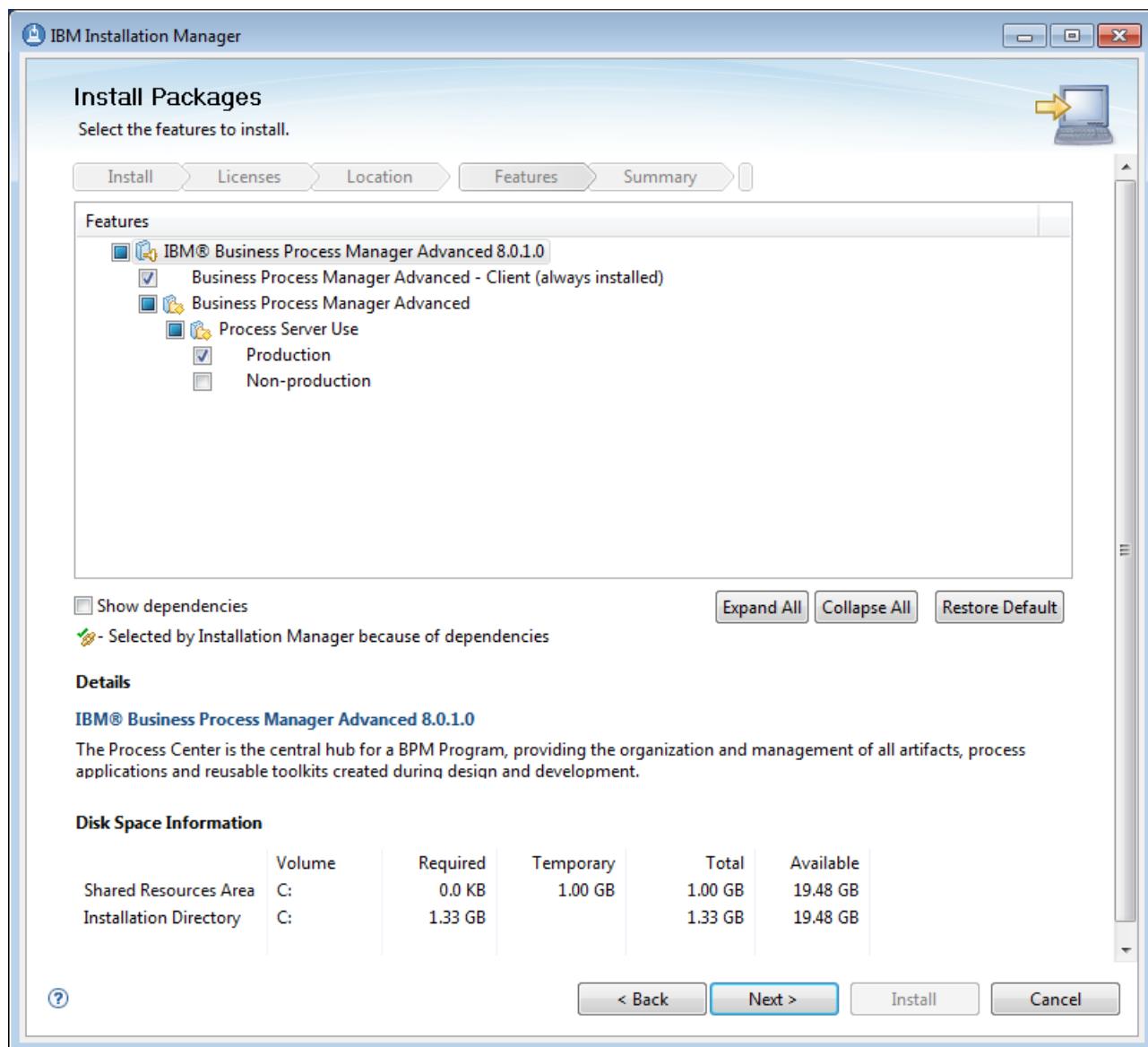


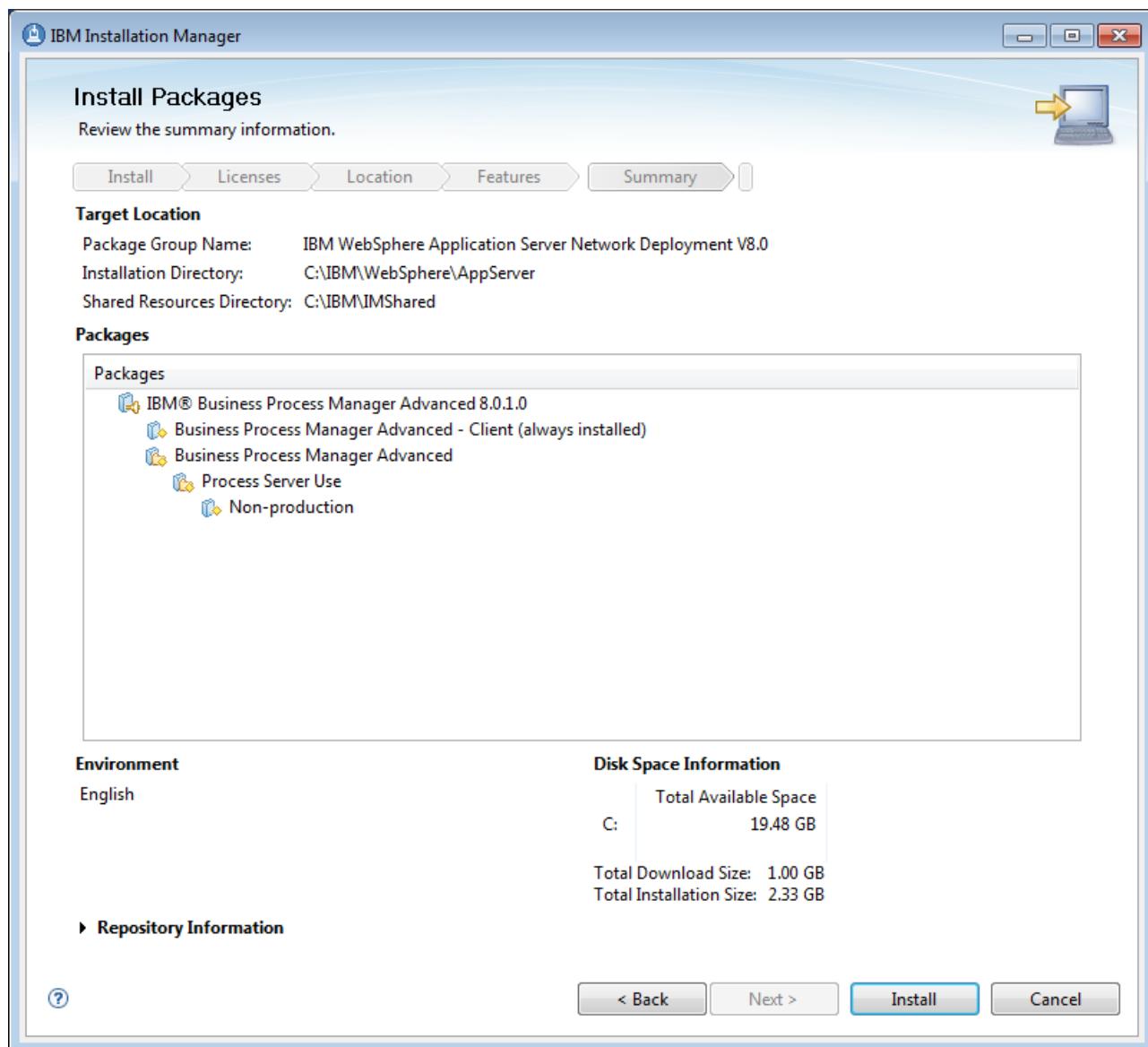
Following this, WAS ND v8.0.0.5 is now installed. Next it is time to install IBM BPM Advanced:

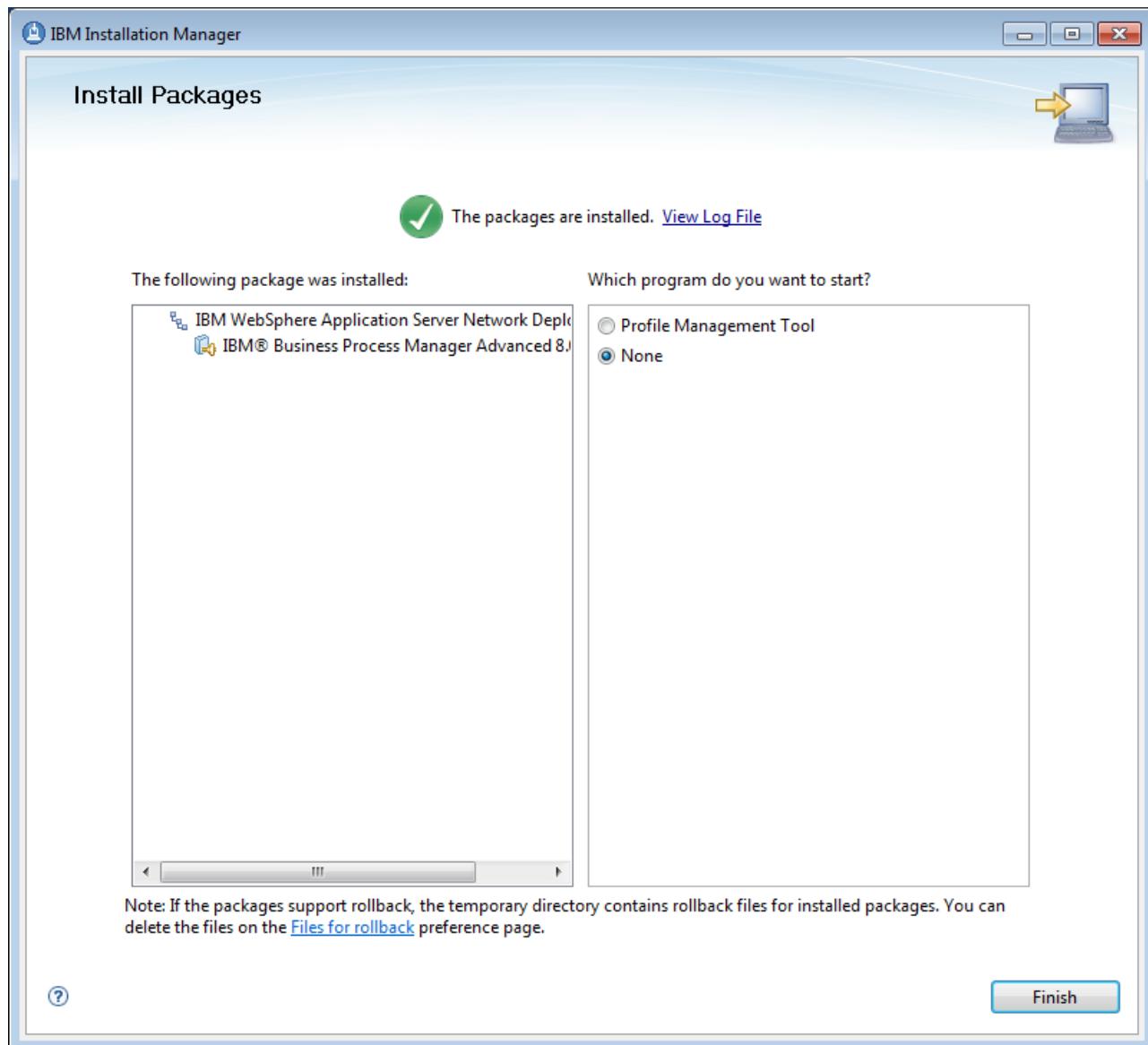






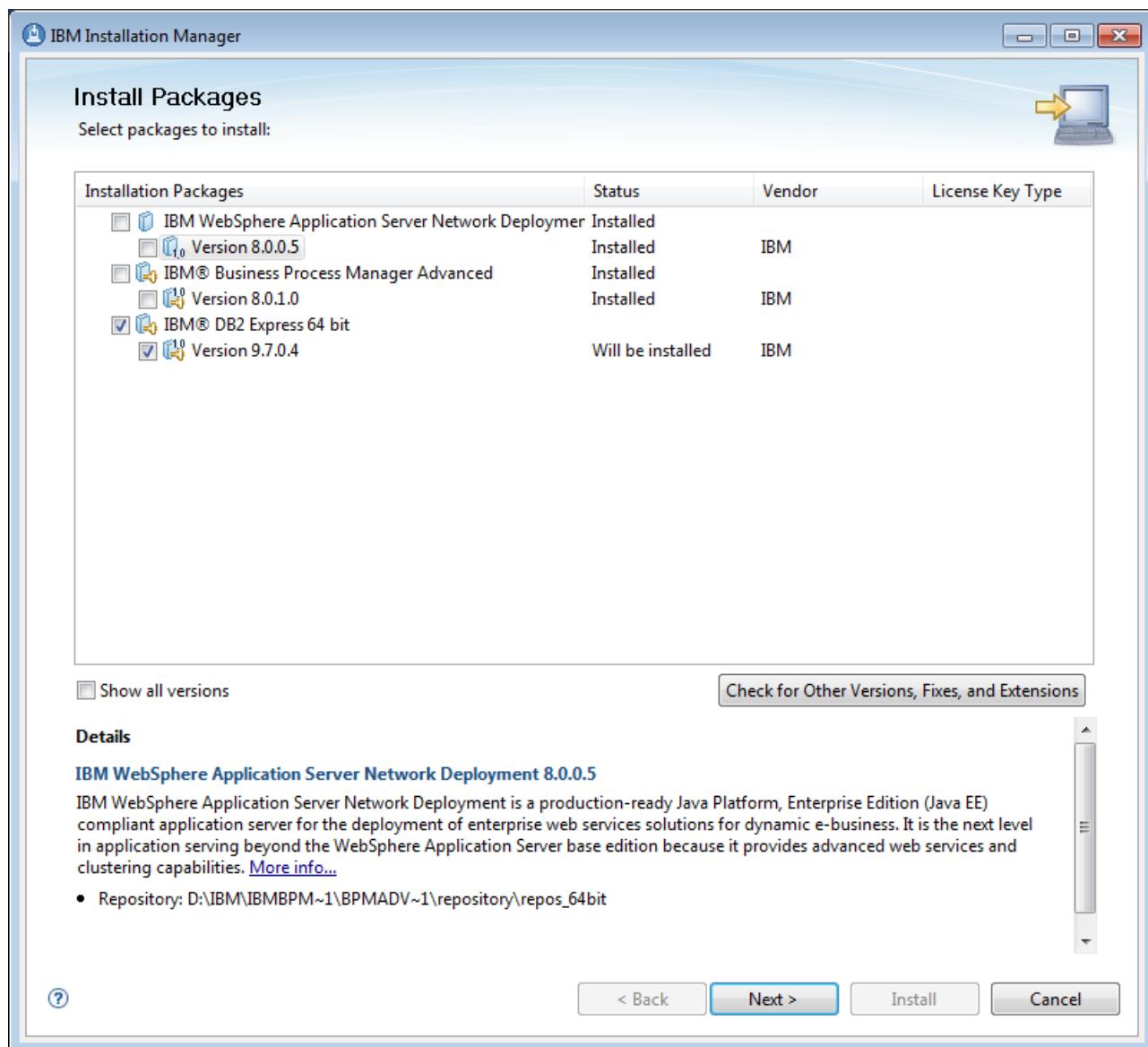


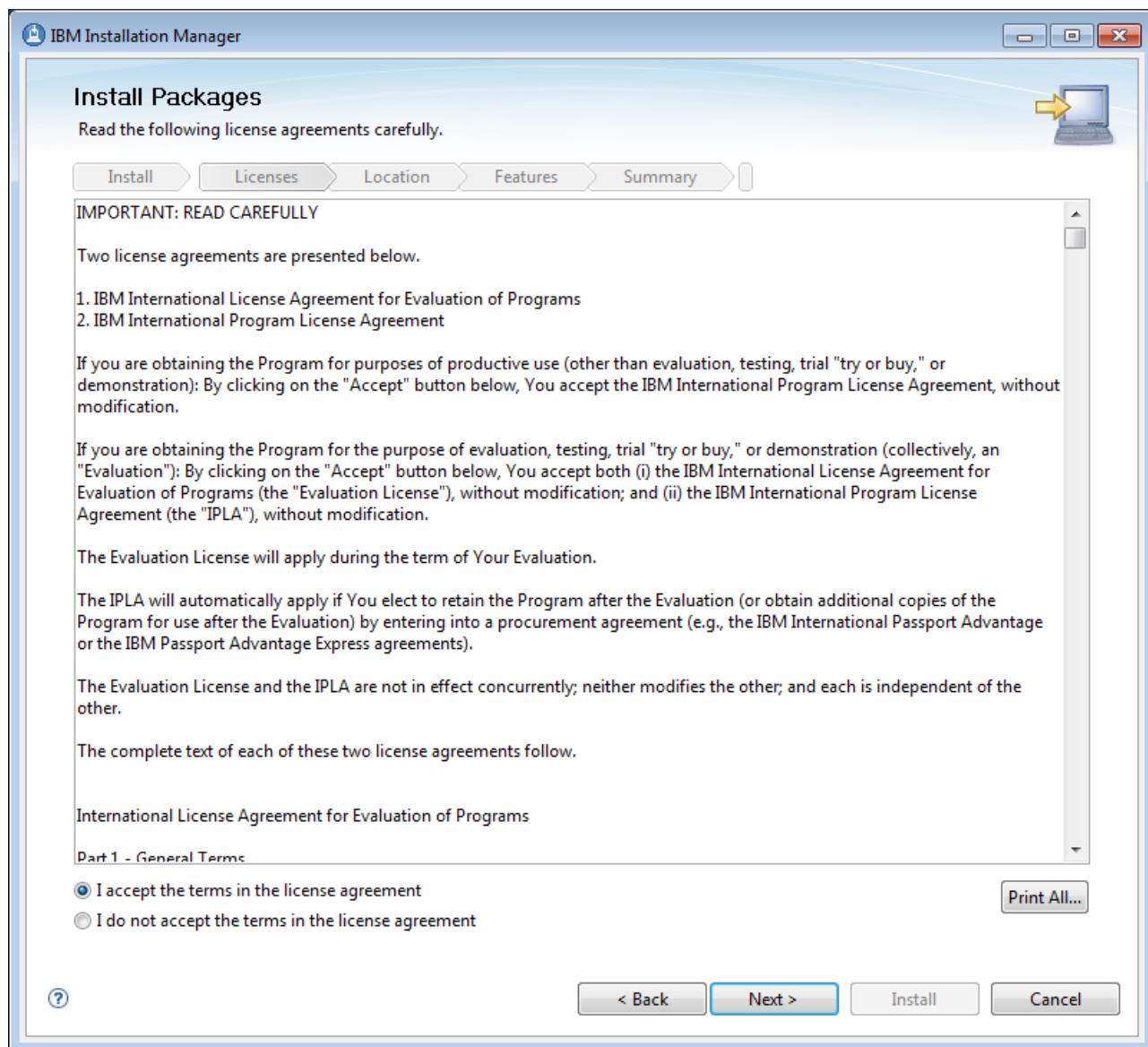


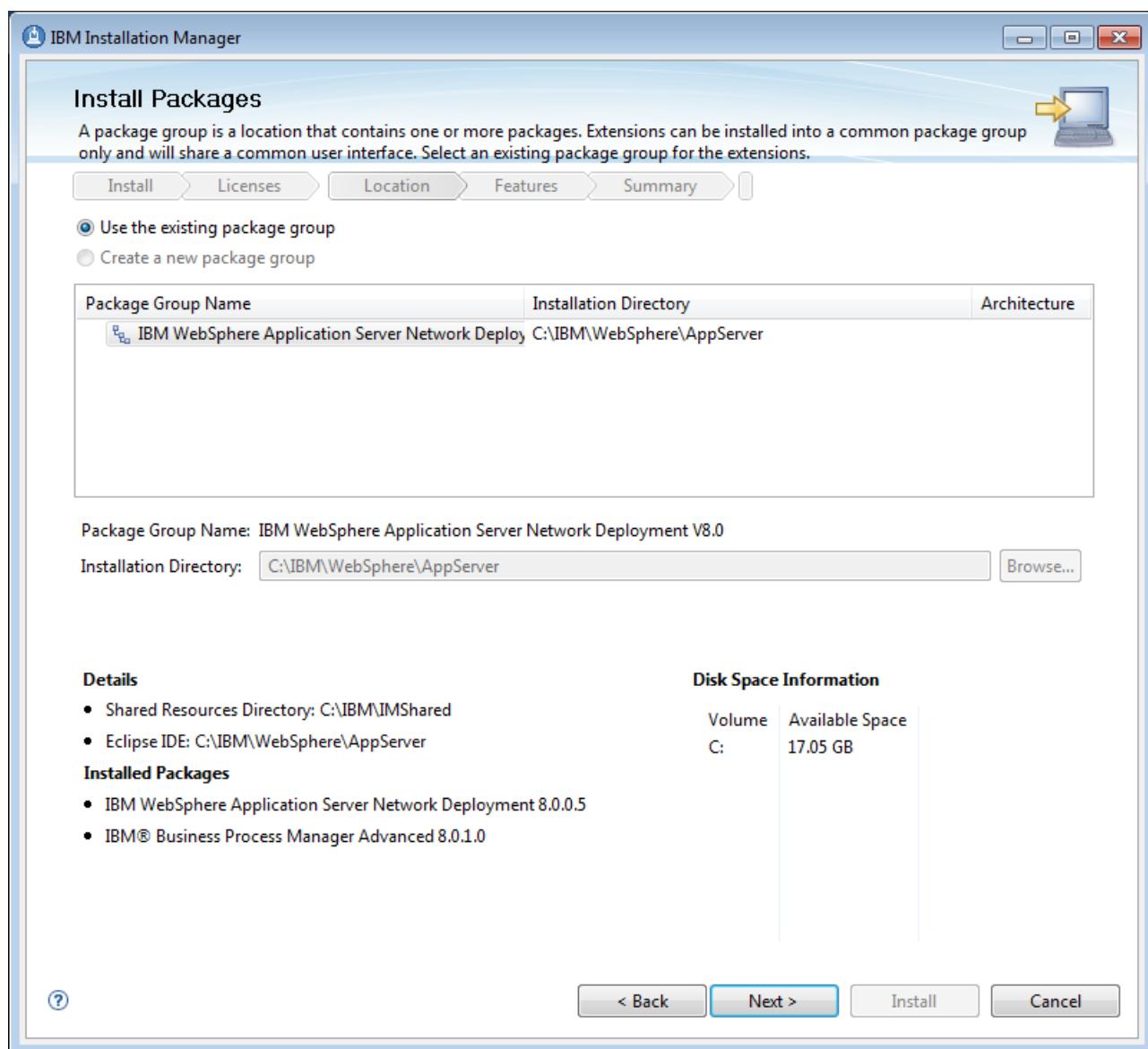


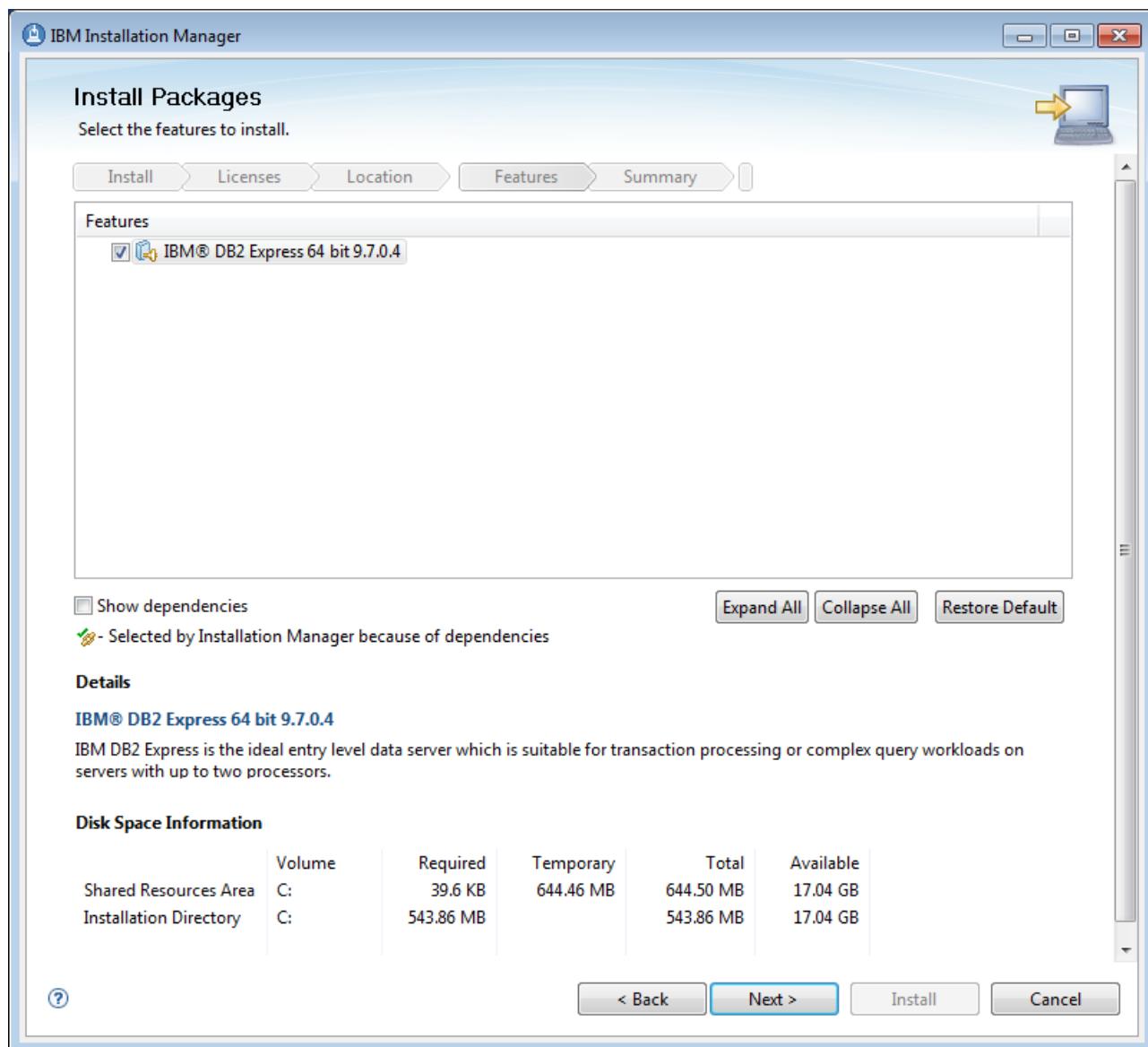
At the conclusion of the installation we are offered the choice of opening up the Profile Management Tool (PMT). PMT is used to create instances of servers including Process Servers and Process Centers.

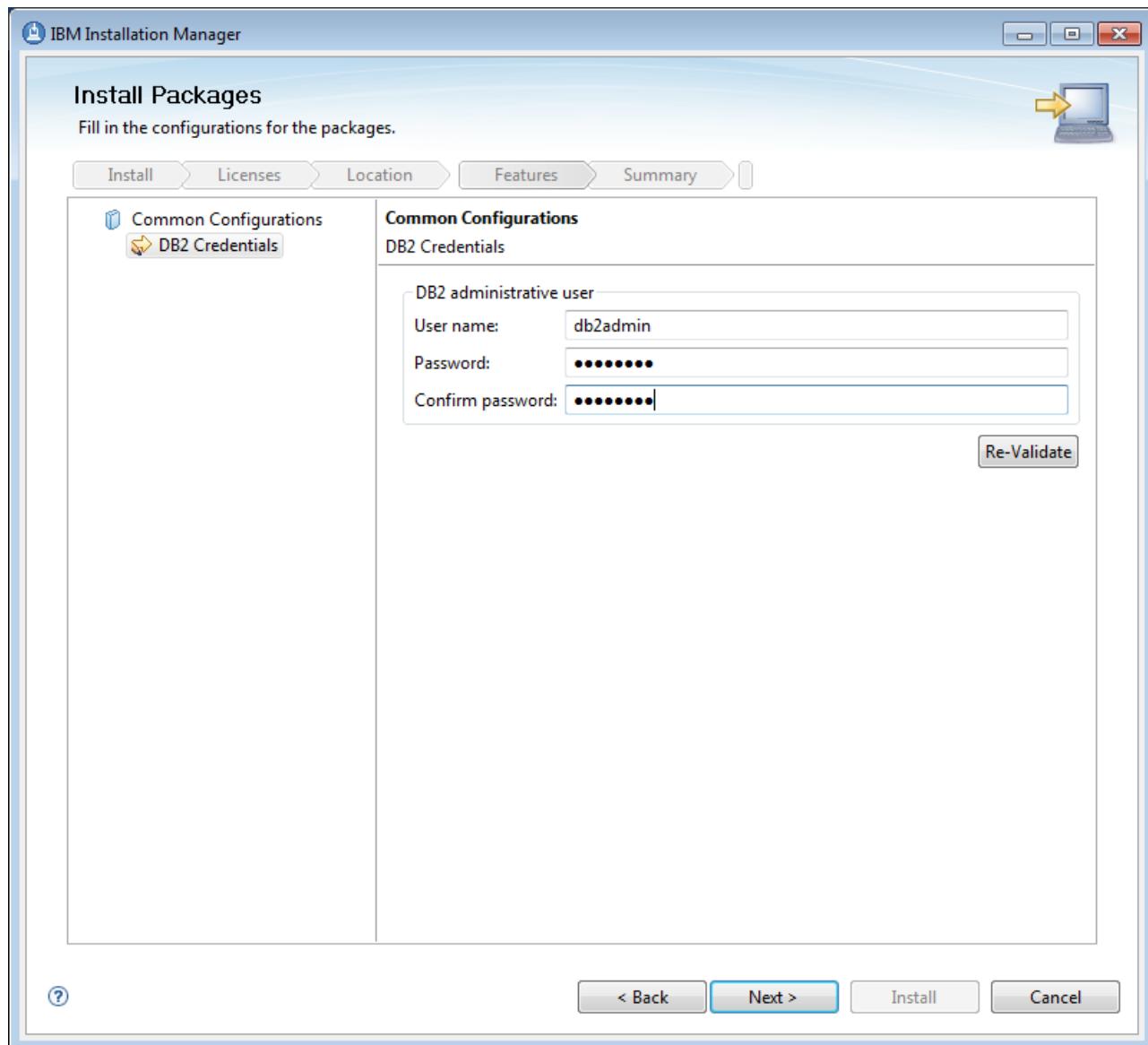
Finally, I install IBM DB2 as the Database Provider for my environment.

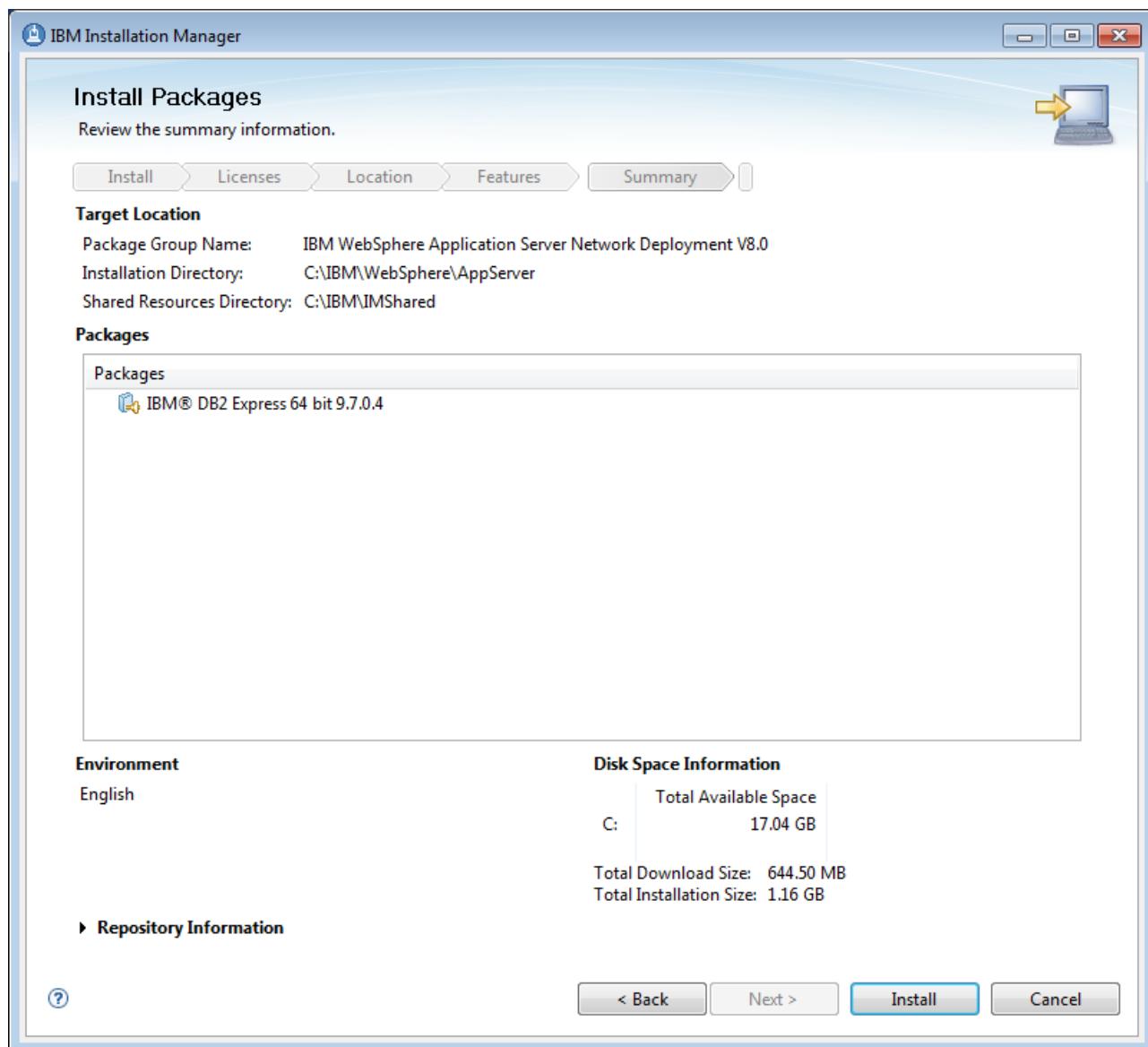


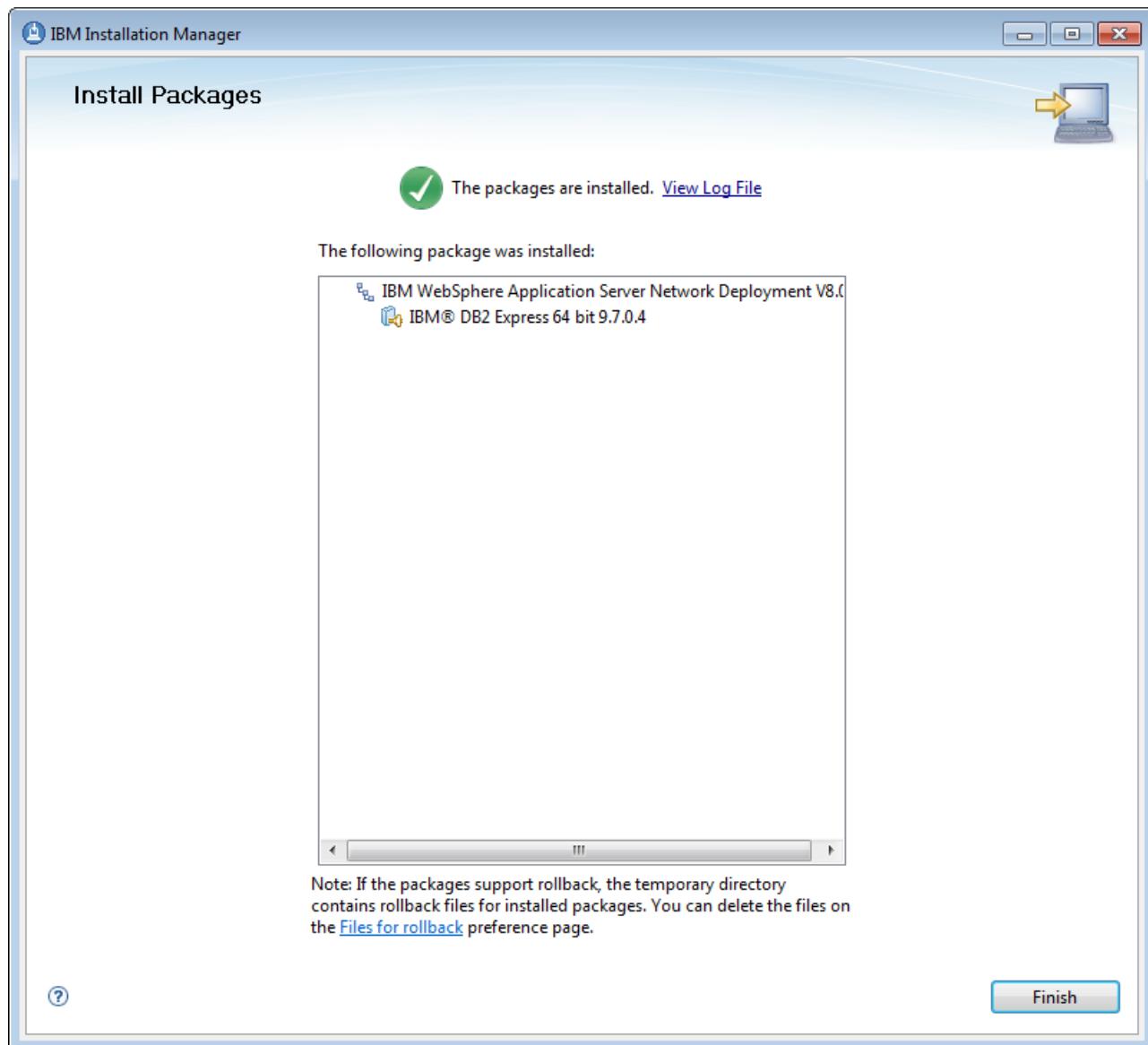












Using BPMConfig to create servers

Starting from BPM v8.5, a command called "BPMConfig" has been supplied which will perform core BPM configuration functions.

BPMConfig is governed by a properties file. A sample can be found in

<Root>\BPM\samples\config

Always take a copy of this and never work with the master directly.

Within the configuration file, there are a number of entries that you can supply many with defaults already provided. Keep a copy of the file for your records. Also build a table:

Property	Default	Your value
bpm.de.name	De1	
bpm.de.authenticationAlias.1.user	<i>dadmin</i>	
bpm.de.authenticationAlias.1.password	<i>password</i>	
bpm.de.authenticationAlias.2.user	<i>db2admin</i>	

bpm.de.authenticationAlias.2.password	<i>password</i>	
bpm.cell.name	PCCell1	
bpm.cell.authenticationAlias.1.user	<i>cadmin</i>	
bpm.cell.authenticationAlias1.password	<i>password</i>	
bpm.dmgr.installPath	<i>C:/IBM/BPM/v8.5</i>	
bpm.de.node.1.installPath	<i>C:/IBM/BPM/v8.5</i>	

Make sure that your DB2 database is already running and ensure that the databases called:

- CMNDB
- BPMDB
- PDWDB

are already created. A sample SQL file called `createDatabase.sql` can be found in

```
<ROOT>/BPM/dbscripts/DB2/create
```

Take three copies of this file (one for each of the databases) and edit the files to reflect the names and userid to be used. Finally, run the scripts to create the databases.

The BPMConfig command to run to create the environment is:

```
BPMConfig -create -de <properties file>
```

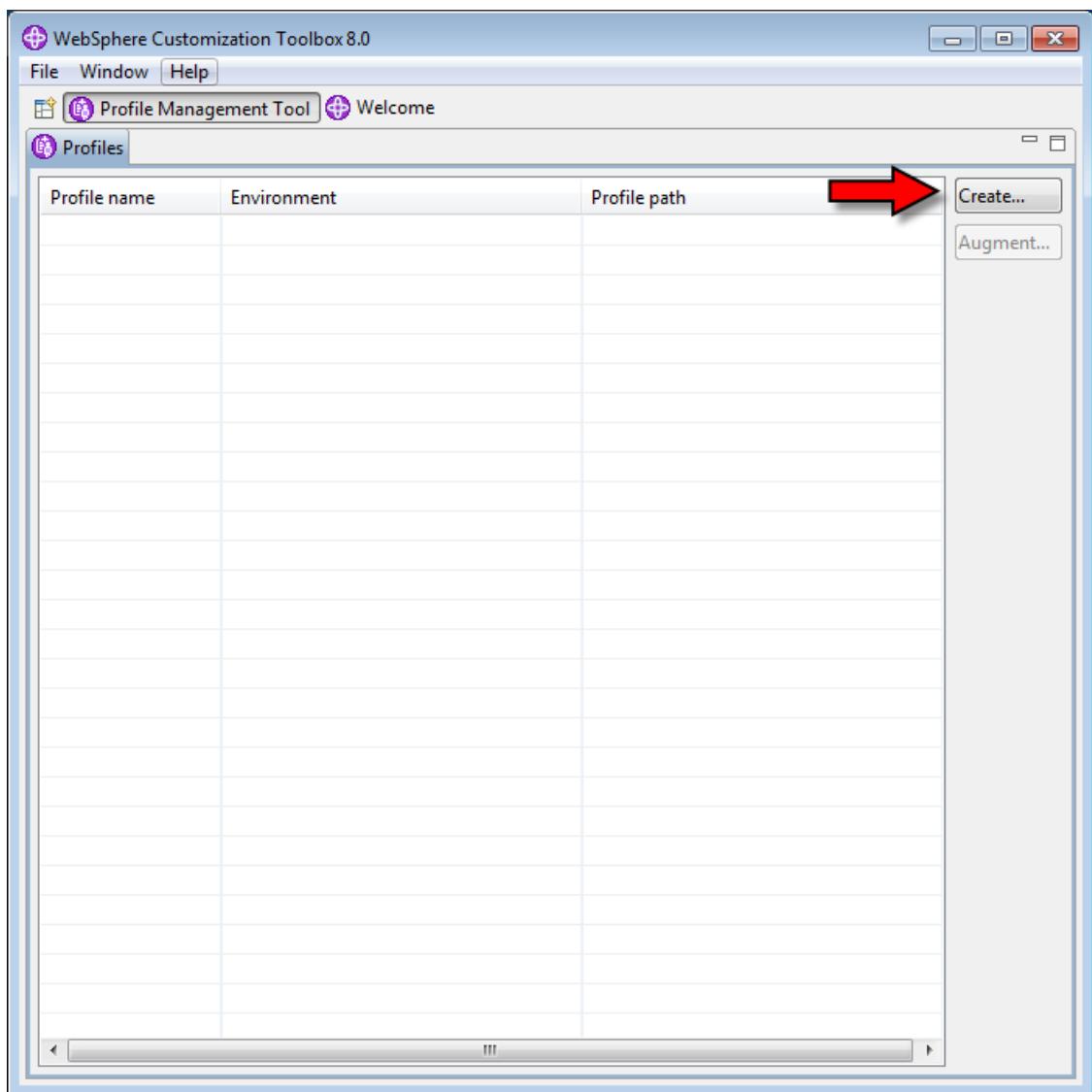
Using the PMT to create servers

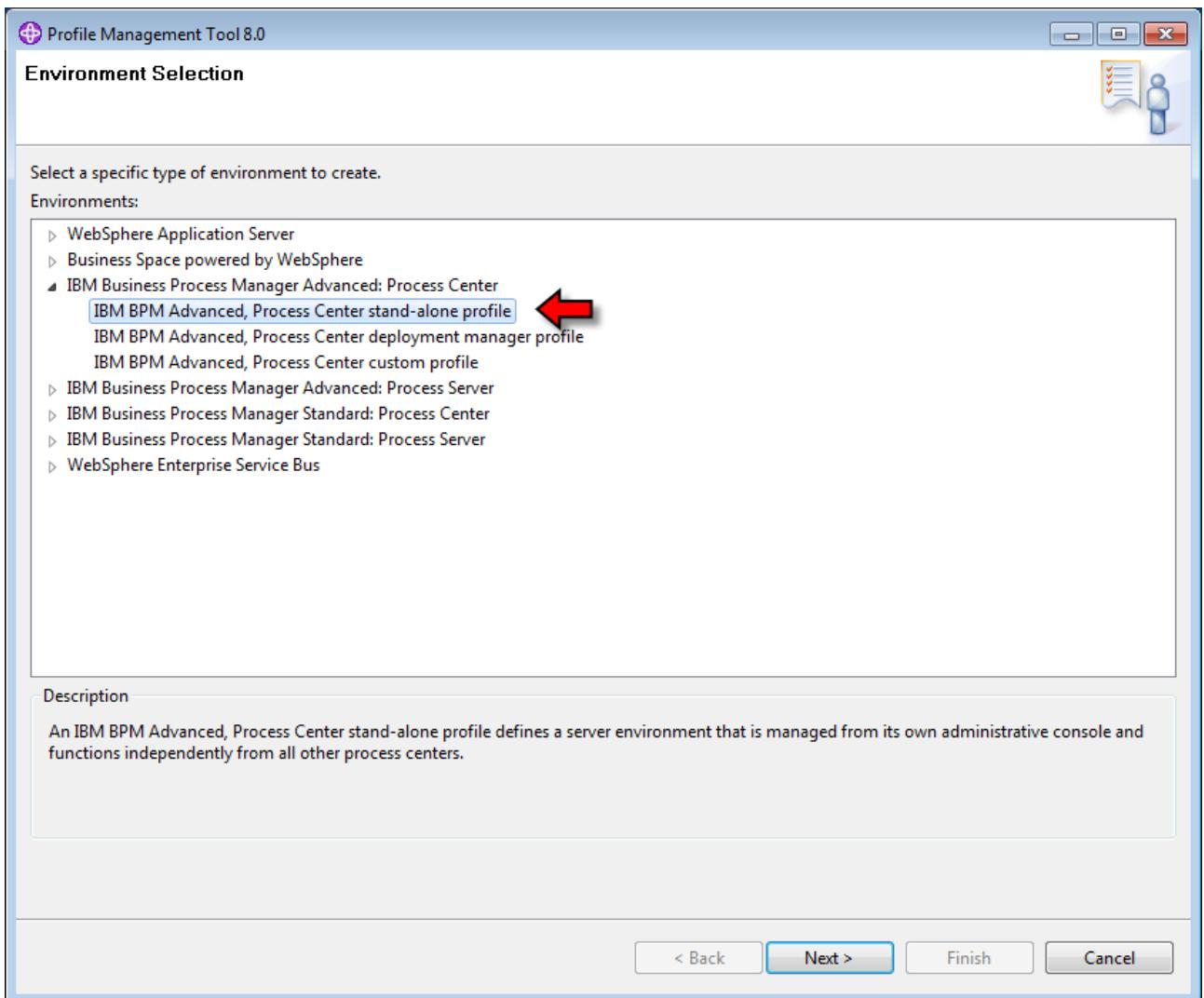
The Profile Management Tool (PMT) is a GUI interface for creating and augmenting instances of WAS profiles. These profiles are the core to running Process Centers and Process Servers.

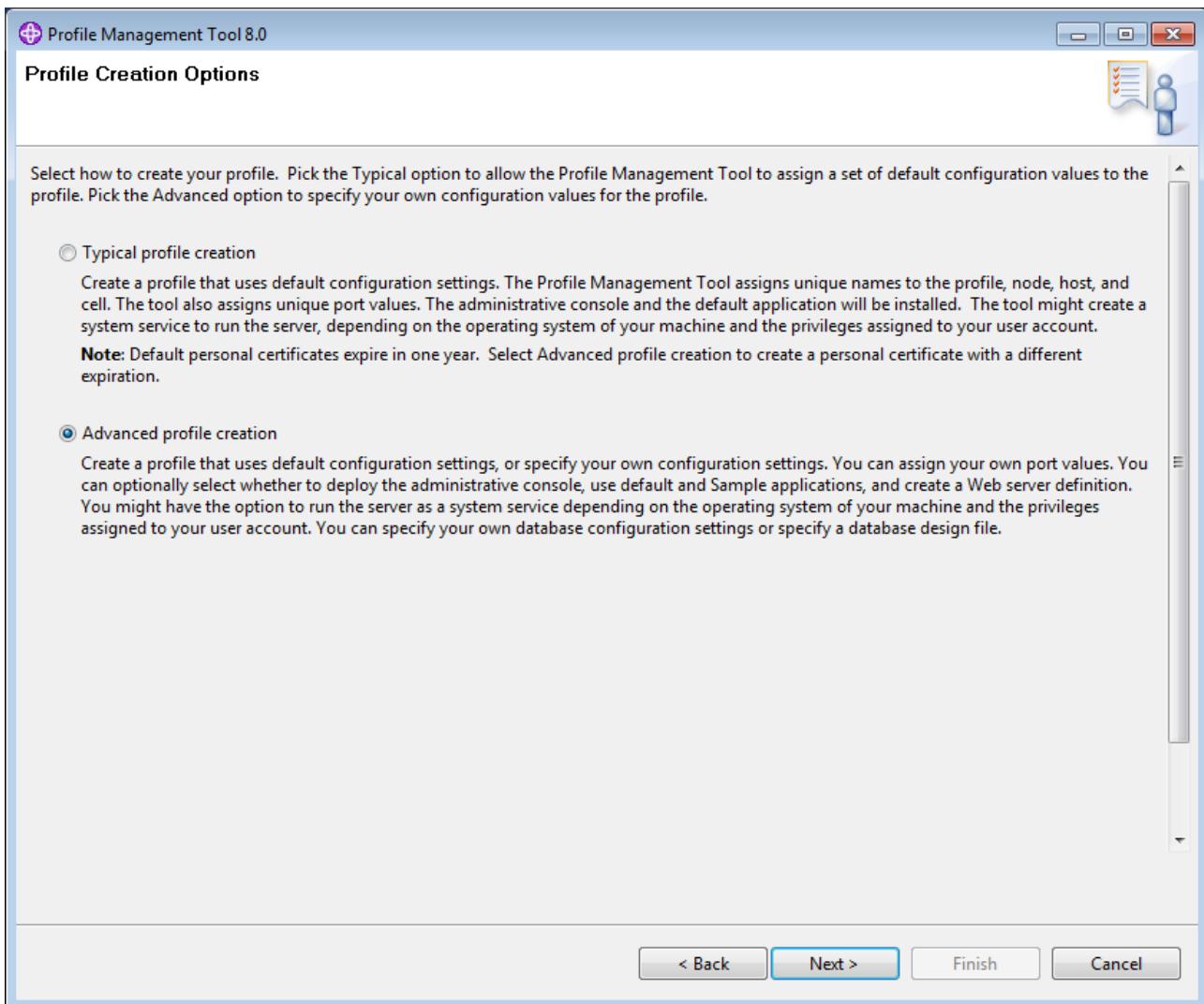
The PMT tool is launched from the file called:

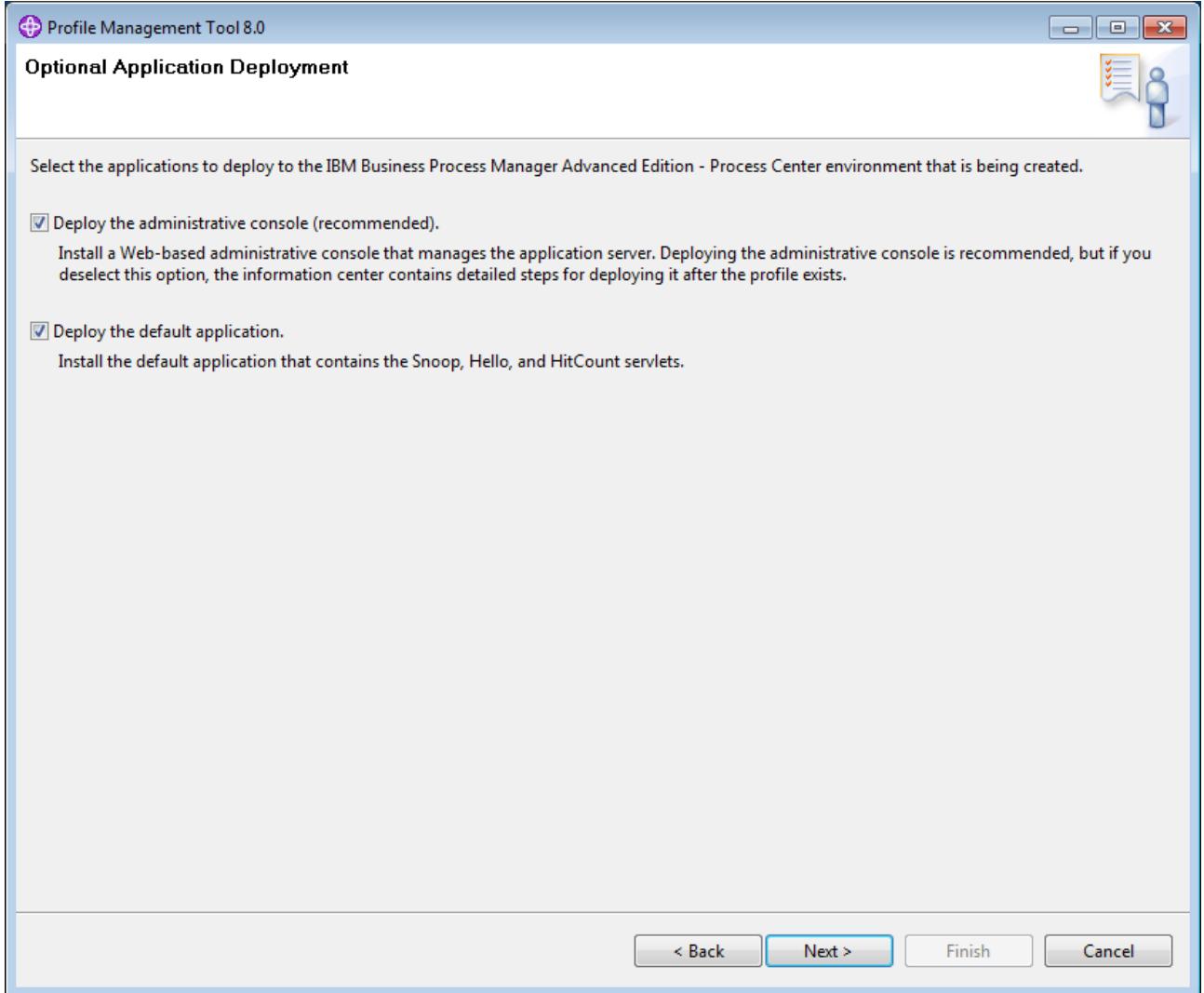
```
<Install>/WebSphere/AppServer/bin/ProfileManagement/pmt.bat
```

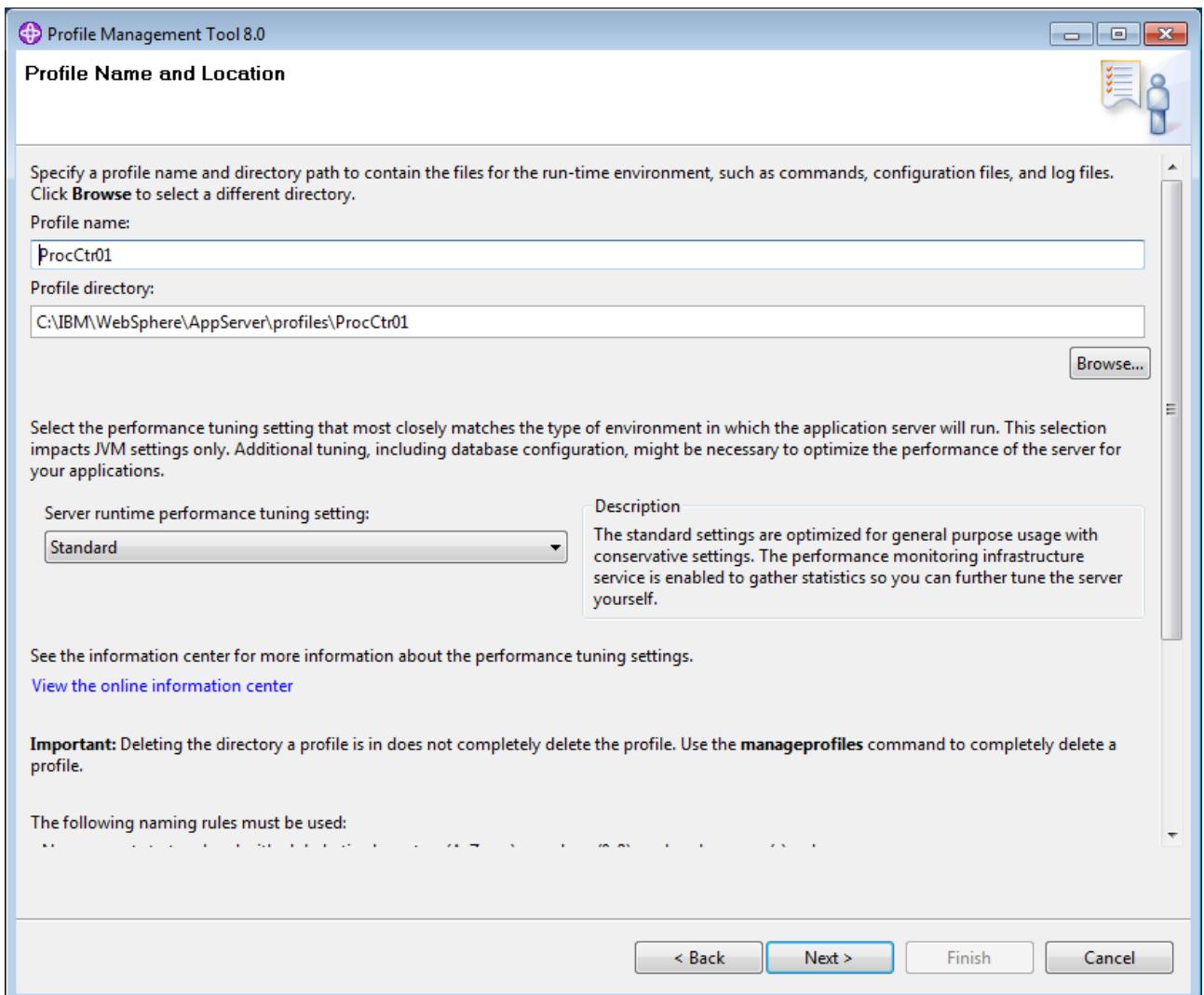
The first page of the wizard shows a list of the products that may be configured using PMT. There is a button to launch the profile management tool proper.

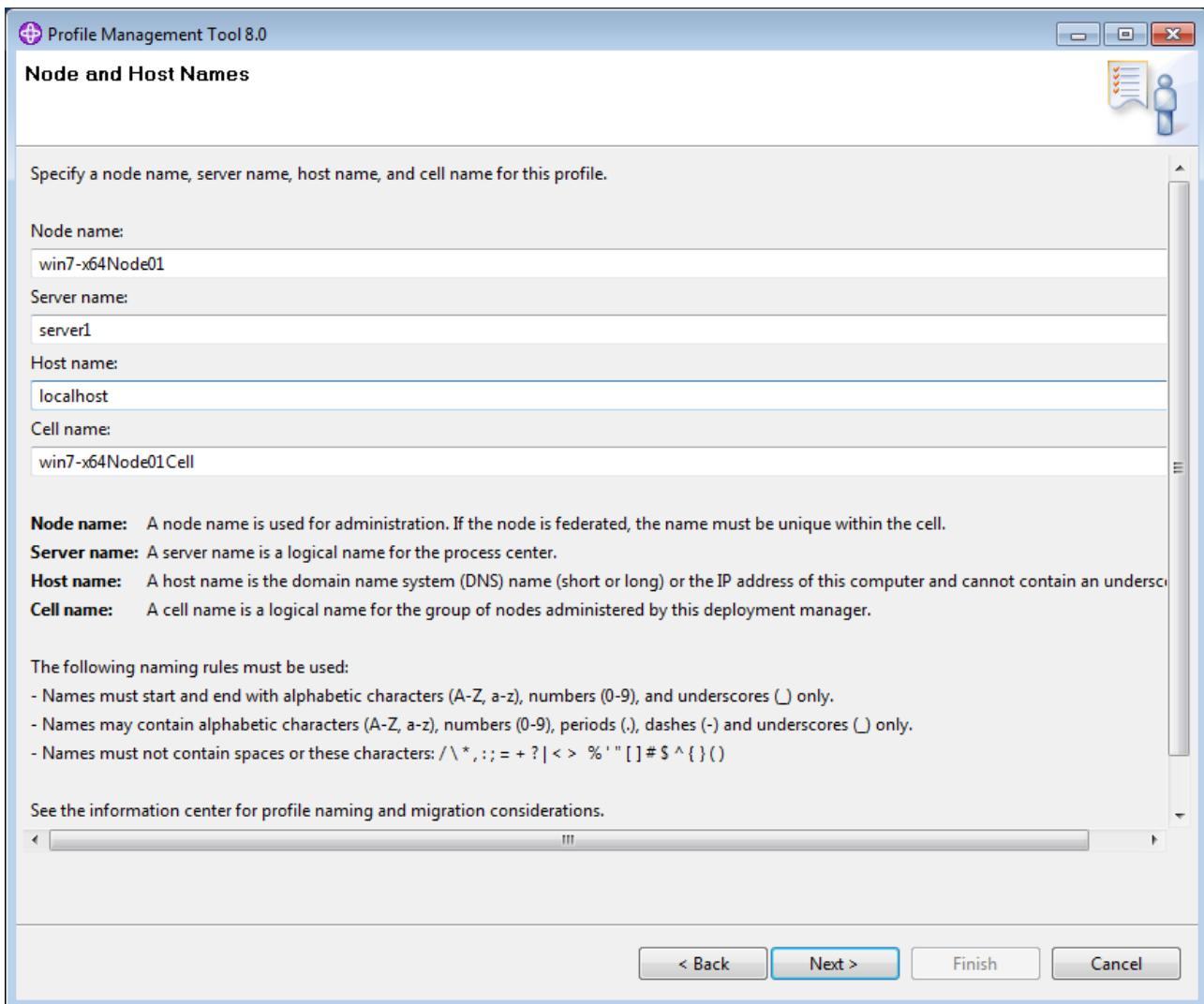












Profile Management Tool 8.0

Administrative Security

Security is always enabled. Supply the user name and password for logging into administrative tools. Permissions with administrative authority are created in a repository within the process center. After profile creation completes, you can use the administrative console to add additional users, groups, or external repositories.

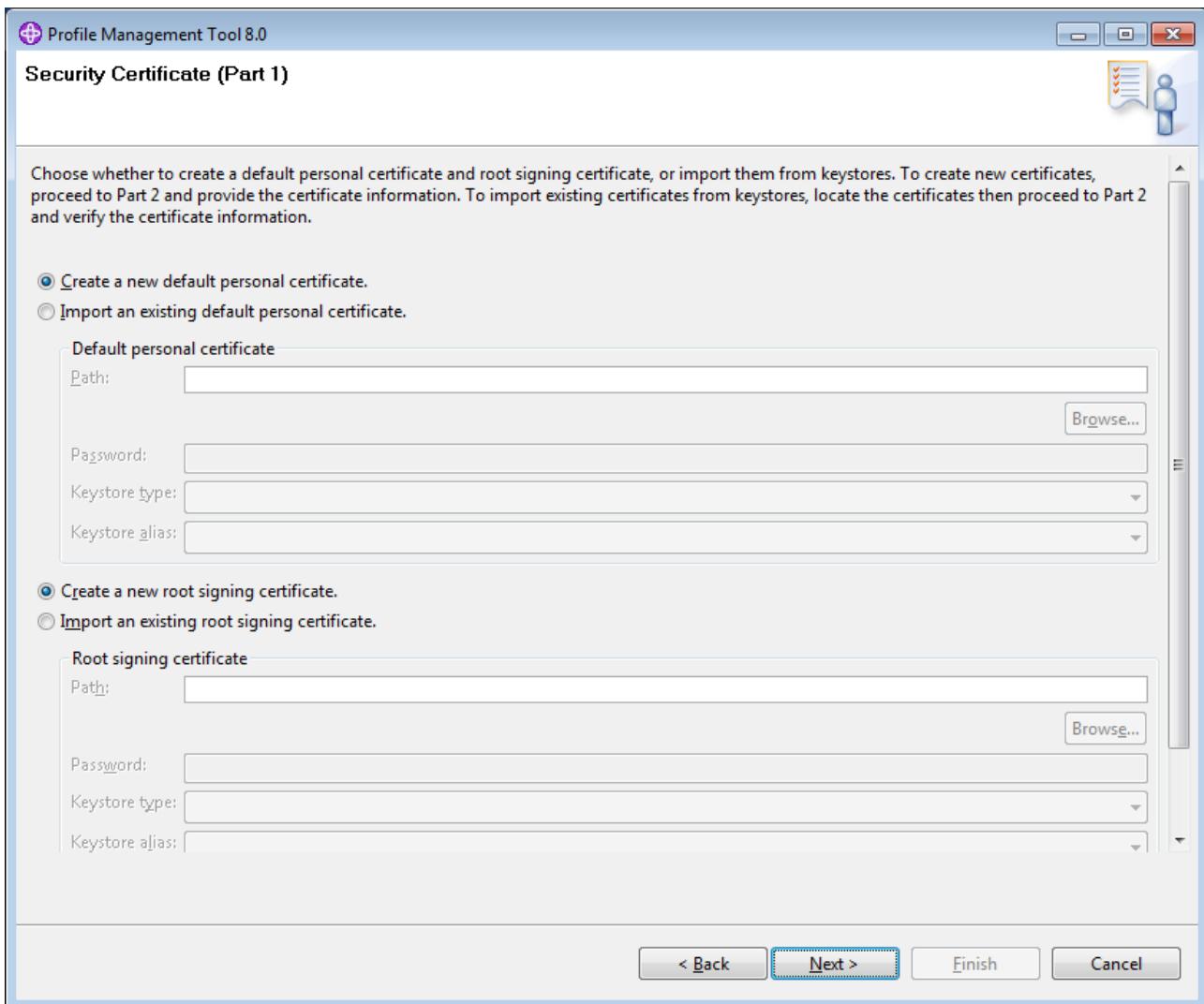
User name:
admin

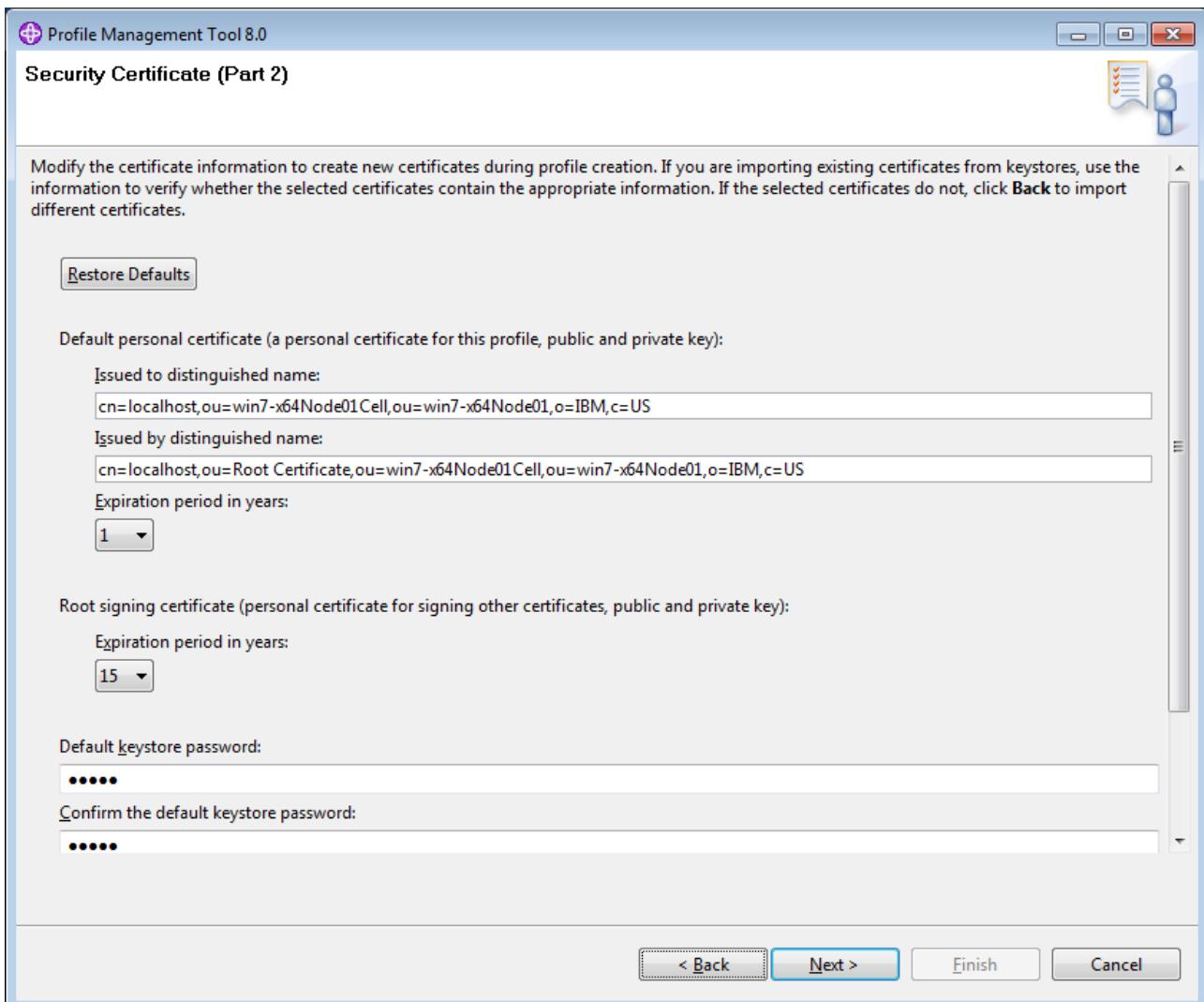
Password:

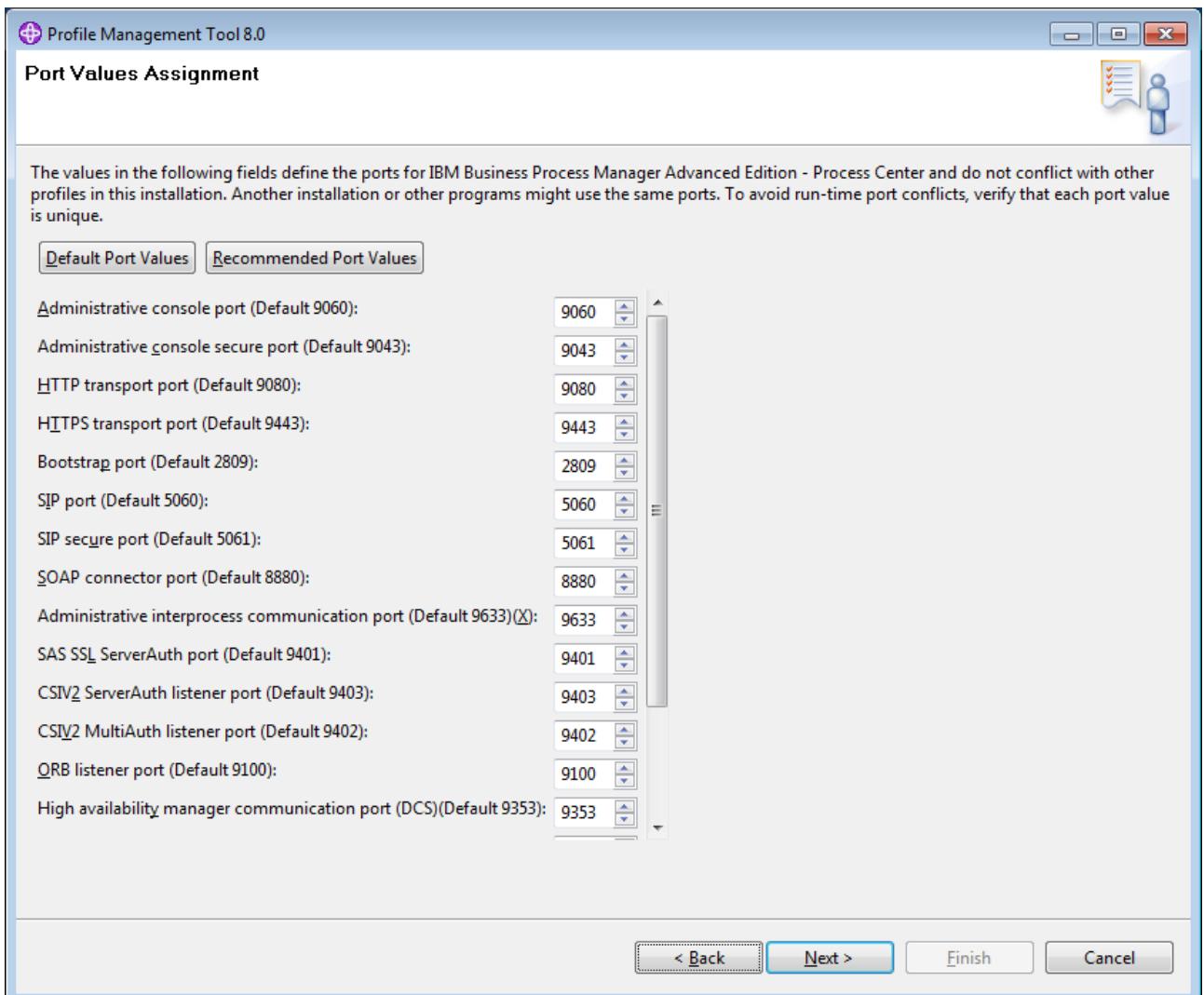
Confirm password:

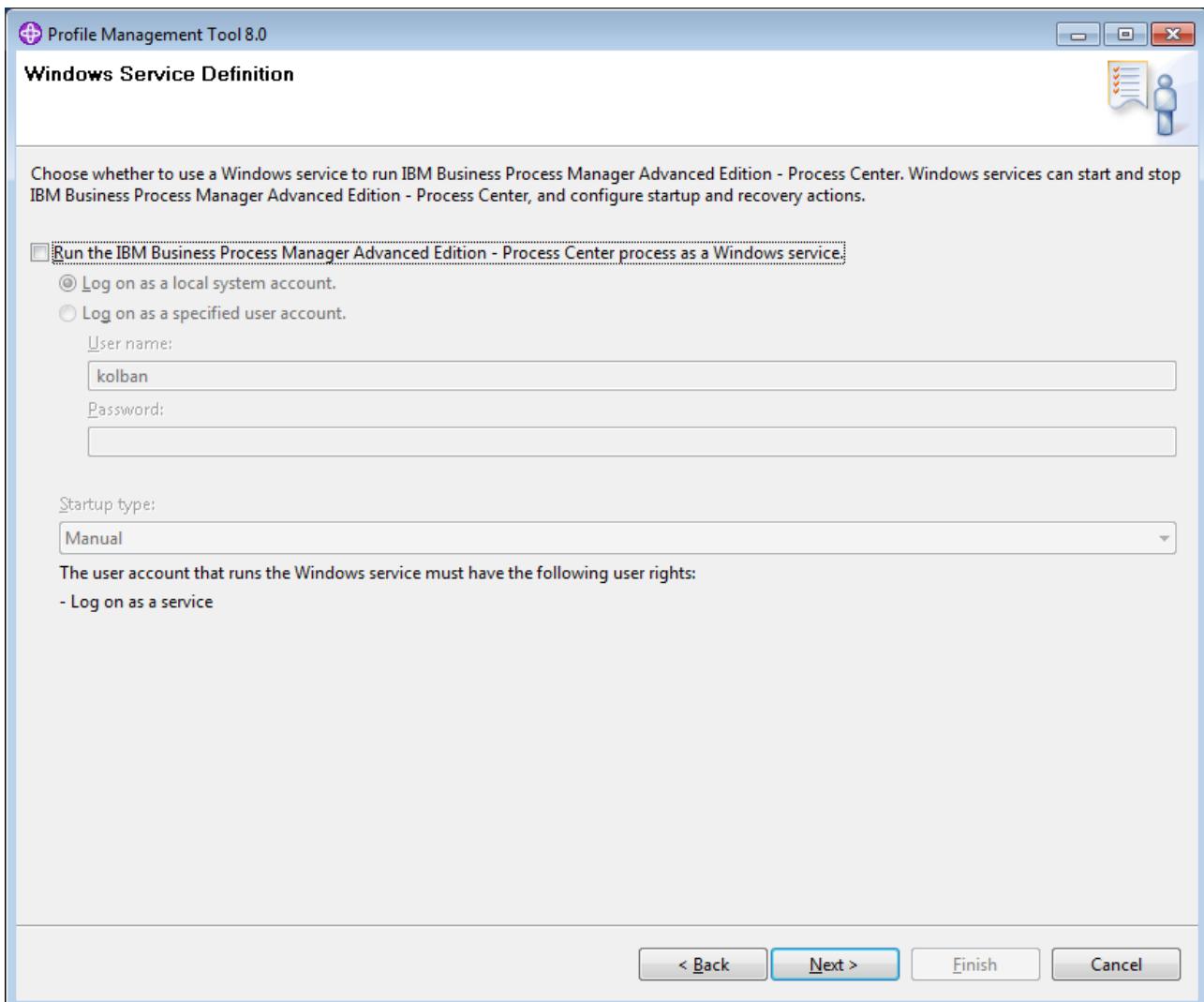
See the information center for more information about administrative security.
[View the online information center](#)

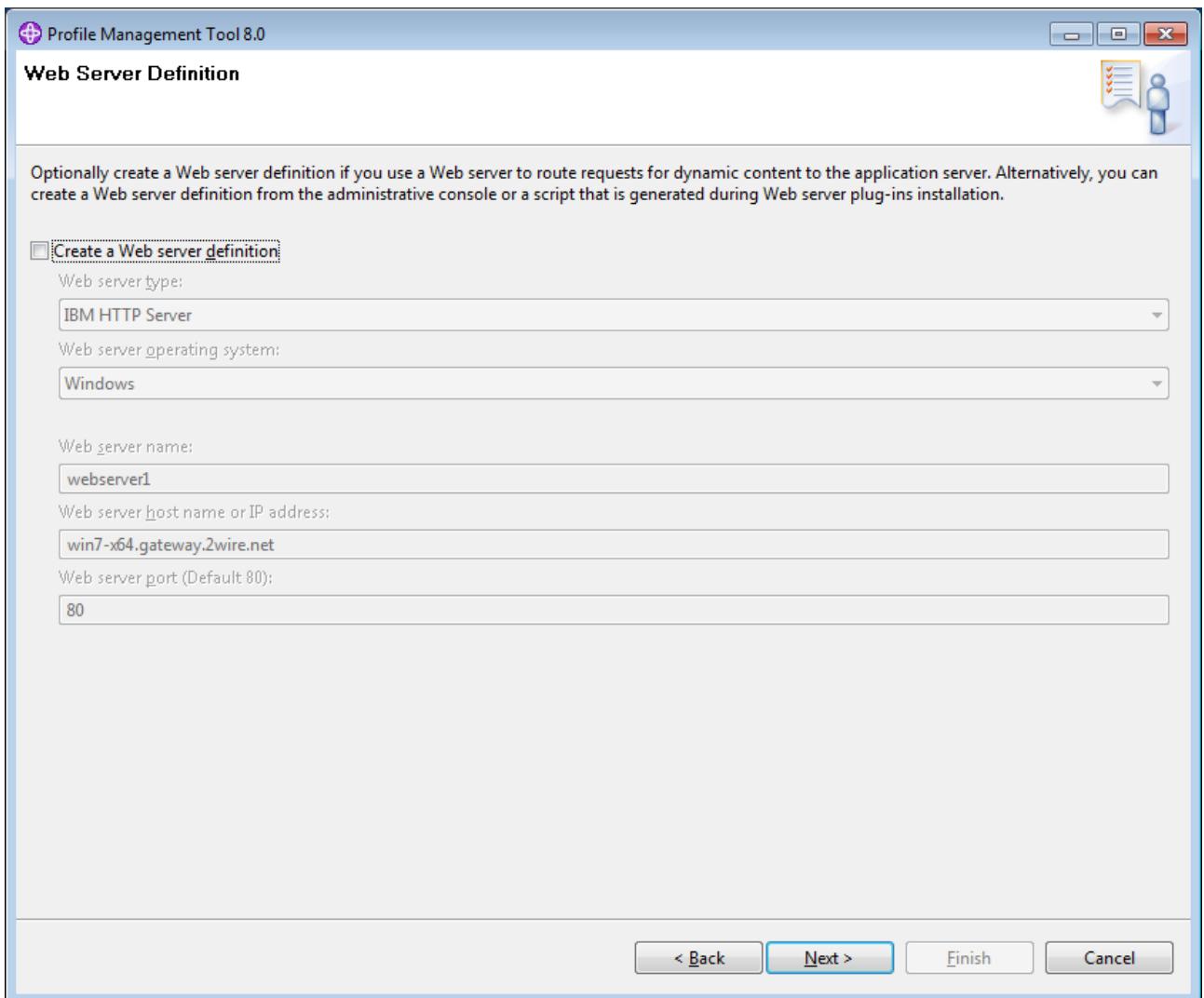
< Back [Next >](#) Finish Cancel

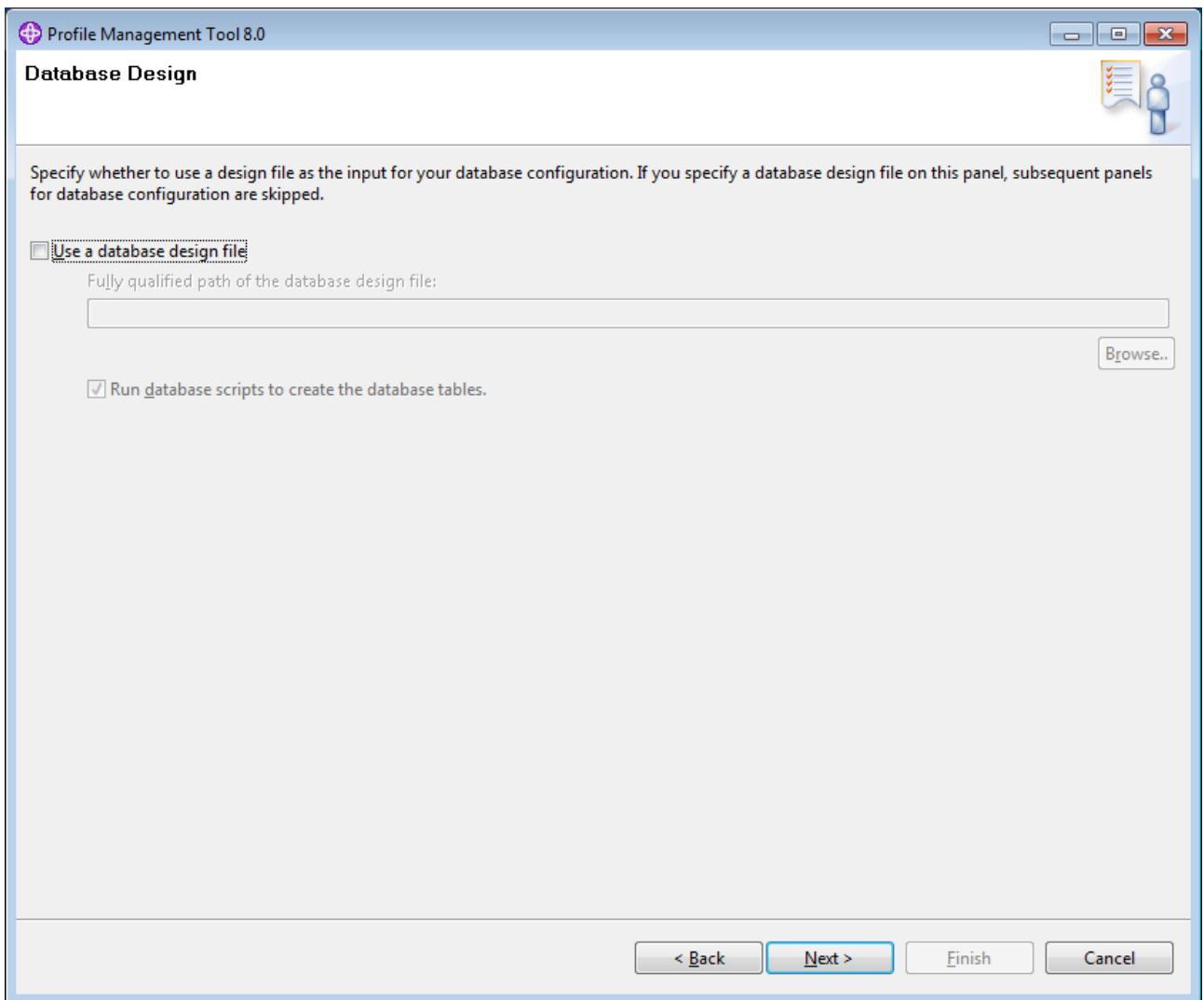












Profile Management Tool 8.0

Database Configuration - Part 1

Provide information about the database and how to process the database scripts for use in IBM Business Process Manager Advanced - Process Center.

If the Profile Management Tool will run the database scripts as part of profile creation, do not specify an existing database in the Database name fields.

Select a database product:

DB2

For each database:

Create a new local database.
 Use an existing local or remote database.

Process server database name:

BPMDB

Performance Data \Warehouse database name:

PDWDB

Common database name:

CMNDB

Run database scripts to initialize the databases
(Do not select this if this profile will be used for migration from WebSphere Lombardi Edition 7.x)

< Back Next > Finish Cancel

Profile Management Tool 8.0

Database Configuration - Part 2

Provide additional information to complete the configuration for the DB2 database.

User name for database authentication:
db2admin

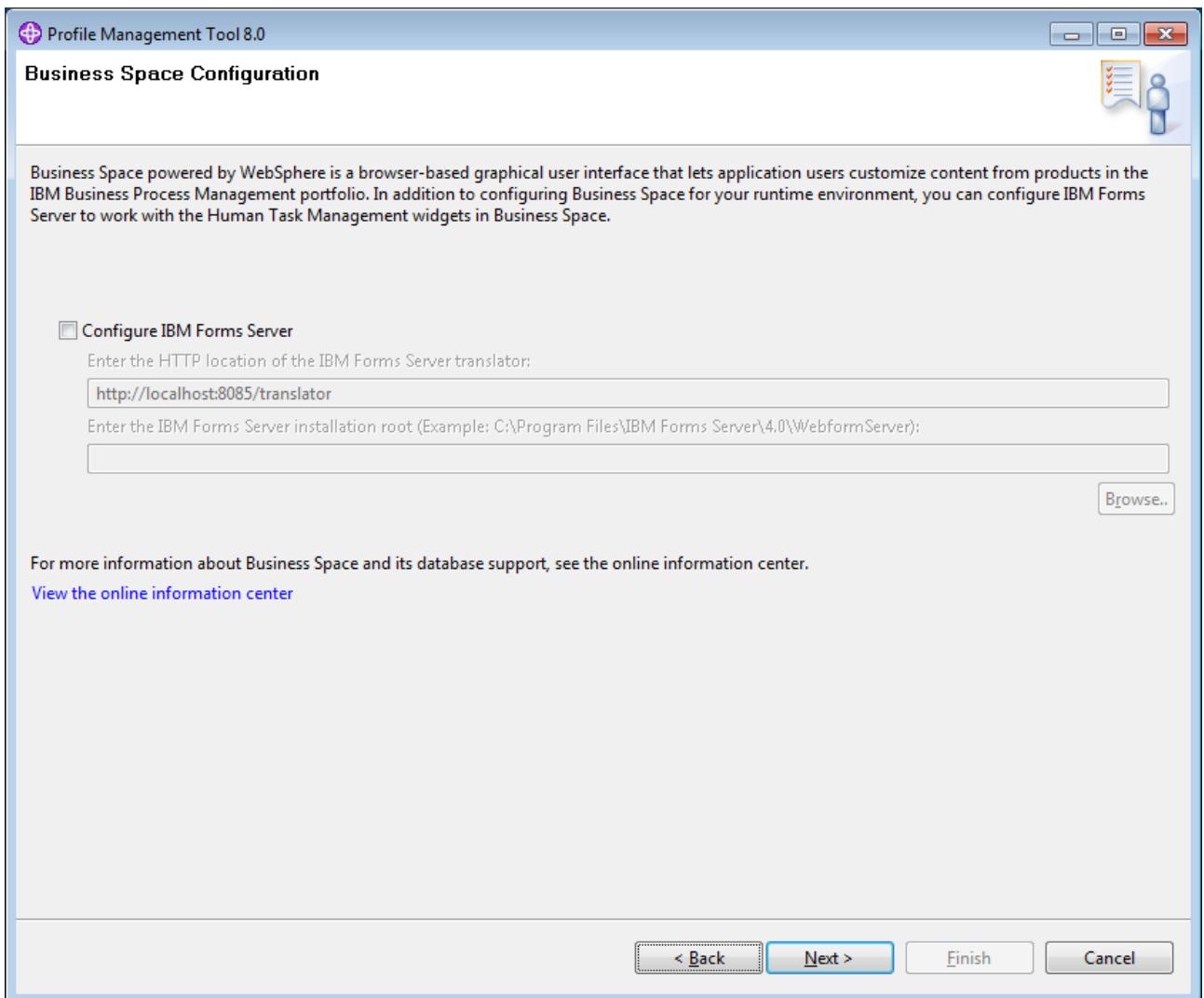
Password for database authentication:

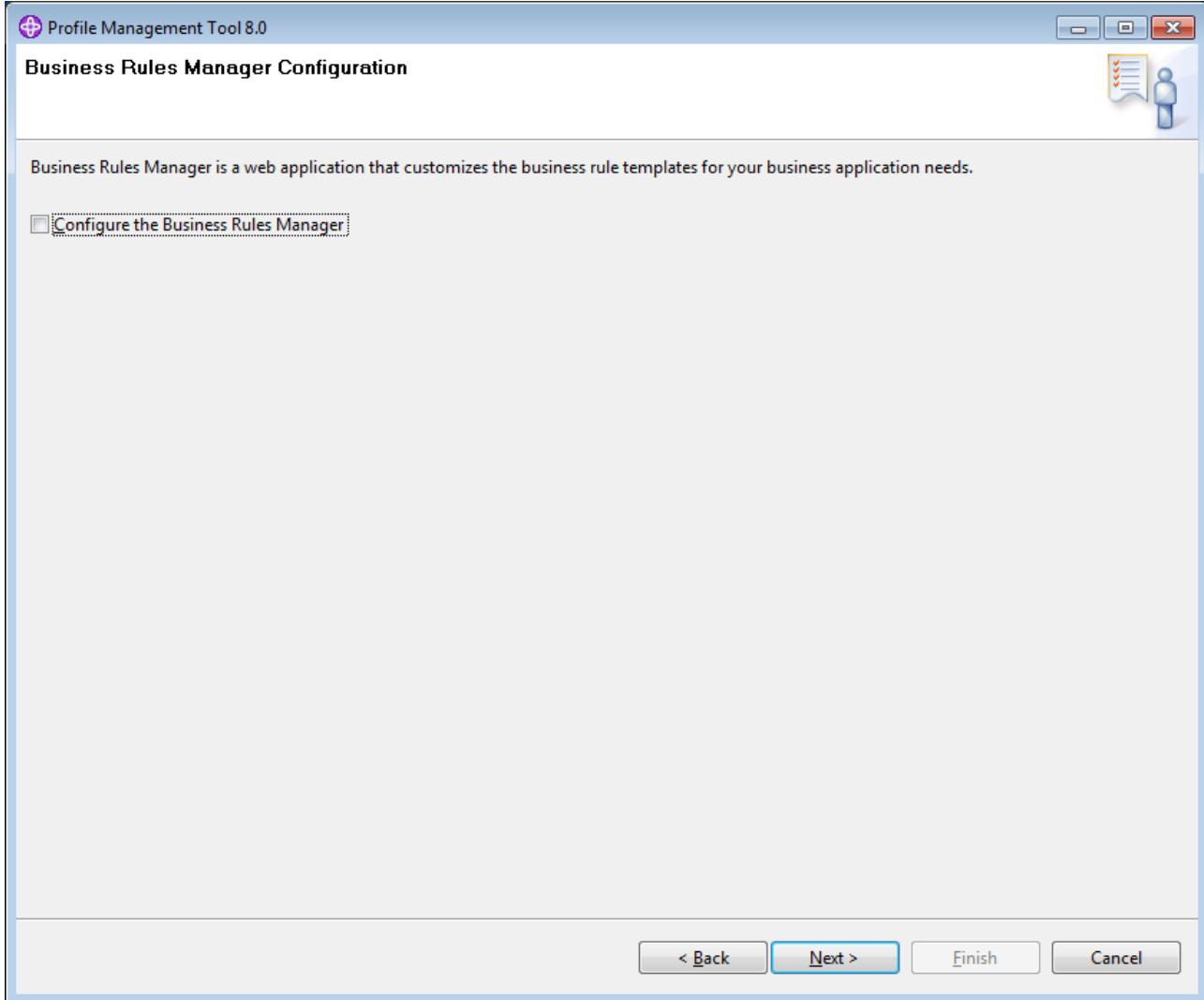
Confirm password:

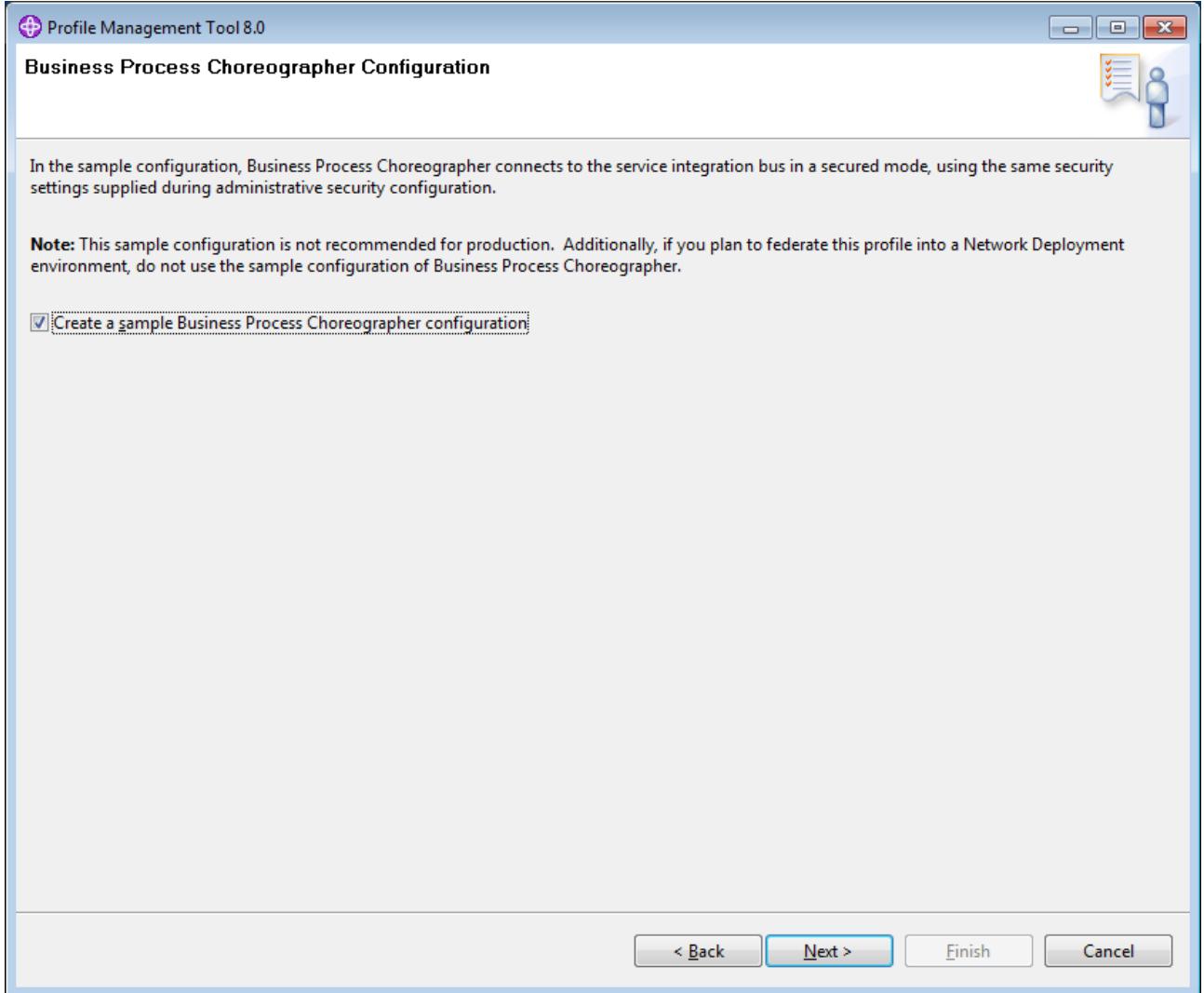
Directory location of JDBC driver classpath files:
\${WAS_INSTALL_ROOT}\jdbcdrivers\DB2

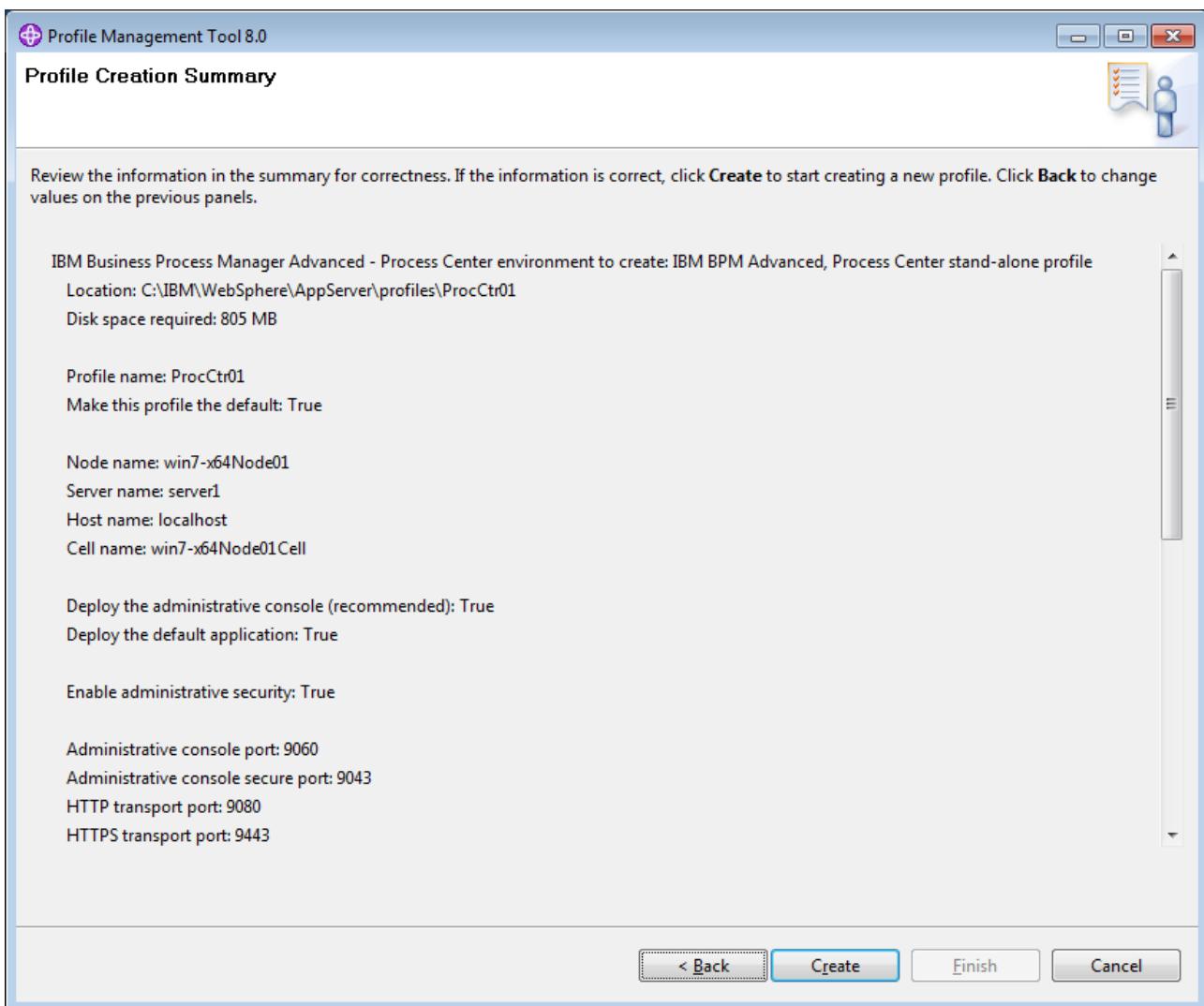
Database server host name (for example IP address):
win7-x64.gateway.2wire.net

Server port:
50000









After configuring the profile, it is ready to be started. In Windows, a set of menu items have been added to the Start menu.

Using the command line to create server profiles

The PMT tool is graphical in nature. As an alternative to creating profiles using PMT, a command line tool called `manageprofiles` can be used to perform a profile creation from text files that are used as input to describe the parameters to be supplied. One advantage of using this mechanism is that the record of how the server was created is now made permanent as opposed to having to remember the values entered interactively in the PMT tool. In addition, the recreation of the servers can be scripted and re-executed in the event of a need to recreate the servers.

There are quite a few flags/parameters that need to be supplied. Some of the parameters are listed below. All the parameters can be found well documented and fully explained in the BPM InfoCenter.

-create	
-templatePath	The path to the template used to control the creation of the profile. The profile templates can be found in the <install>/profileTemplates folder. The BPM templates are in the BPM folder. These include: <ul style="list-style-type: none">• default.procctr

	<ul style="list-style-type: none"> • default.procctr.advanced • default.procsvr • default.procsvr.advanced
-profileName	Name of the profile to be created
-adminUserName	Name of the admin userid
-adminPassword	Password for the admin userid
-dbType	Type of the database to be used: <ul style="list-style-type: none"> • DB2_UNIVERSAL • DB2_DATASERVER • DB2UDBOS390 • MQSQLSERVER_MICROSOFT • ORACLE
-procSvrDbName	Name of the database for Process Server (eg. BPMDB)
-dbProcSvrUserId	Userid used by Process Server to connect to DB
-dbProcSvrPassword	Password for userid used by Process Server to connect to DB
-perfDWDdbName	Name of the database for Performance Data Warehouse (eg. PDWDB)
-dbPerfDWUserId	Userid used for the Performance Data Warehouse
-dbPerfDWPassword	Password for userid used for Performance Data Warehouse
-dbDelayConfig	Should the configuration of the databases be delayed. A boolean value of either true or false
-dbName	Name of the database for common (BPM Advanced) tables
-dbUserId	Userid for the common database
-dbPassword	Password for the common database
-configureBSpace	Should Business Space be configured? (true/false) The default is true.
-configureBRM	Should Advanced Business Rules manager be configured? (true/false) The default is false.
-winServiceCheck	Should a Windows service be created to start this profile. Defaults to false.
-dbJDBCClassPath	Class path for JDBC drivers. Default is based on DB type. <ul style="list-style-type: none"> • \${WAS_INSTALL_ROOT}/jdbcdrivers/DB2
-dbHostName	Host where DB is installed
-dbServerPort	Port number where DB is listening (eg. 50000)
-dbCreateNew	? (true/false)
-configureBPC	? (true/false)
-defaultPorts	
-webServerCheck	Should Web Server definitions be made (true/false). Defaults to false.

It is strongly recommended to create a text file which contains all these parameters in the form of name=value pairs. This file is called a *response file* by the manageprofiles tool. The content of this response file can then be used as input to the manageprofiles command using the syntax:

```
manageprofiles -response <fileName>
```

An example configuration looks as follows:

```
create
templatePath=C:/IBM/WebSphere/AppServer/profileTemplates/BPM/default.procctr.adv
profileName=Test01
adminUserName=admin
```

```
adminPassword=admin
configureBSpace=false
dbType=DB2_UNIVERSAL
dbDelayConfig=true
dbName=CMNDB2
dbUserId=db2admin
dbPassword=db2admin
procSvrDbName=BPMDB2
dbProcSvrUserId=db2admin
dbProcSvrPassword=db2admin
perfDWDdbName=PDWDB2
dbPerfDWUserId=db2admin
dbPerfDWPassword=db2admin
defaultPorts=true
omitAction=samplesInstallAndConfig
```

Installing to (DBA) managed databases

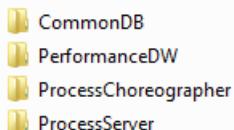
There are occasions where the database used for operation of IBPM is not accessible during installation. This is commonly because the database is owned by a Database Administrator (DBA) who does not want to grant broad authority to the IBPM connected userid. Instead, installation and configuration should proceed without connecting to the database and, once installed, a series of scripts are provided to the DBA which he can examine their contents and convince himself that all is well. The DBA will then execute the scripts with his administrative authority which will create the necessary artifacts for the product's operation. Once the database has been configured, the product will then be able to run.

After configuring a server without applying the database changes, the scripts for configuration can be found at:

```
<Profiles>/<Profile Name>/dbscripts
```

DB2

After profile creation for an IBPM Advanced, the following artifacts can be found in
`<Profiles>/<Profile Name>/dbscripts`



Running the CommonDB scripts

In the CommonDB folder, we will find a command called `configCommonDB.bat`. This is a shell script that should be run with the flag "createDB" (eg. `configCommonDB.bat createDB`). When executed, this creates the CommonDB database and creates tables within it.

Loading the PerformanceDW database

Run the following commands:

```
db2 -tvf createDatabase.sql
```

This will create the PDW database (eg. PDWDB)

Next run the commands

```
db2 connect to PDWDB user db2admin  
db2 -tvf createTable_PerformanceDW.sql  
db2 terminate
```

Loading the Process Server database

Run the following commands:

```
db2 -tvf createDatabase.sql
```

This will create the Process Server database (eg. BPMDB)

```
db2 connect to BPMDB user db2admin  
db2 -tvf createTable_ProcessServer.sql  
db2 -tdGO -vf createProcedure_ProcessServer.sql  
db2 terminate
```

Loading the ProcessChoreographer Database

No explicit connect to a database is needed for this step as the SQL itself contains a CONNECT.

```
db2 -tvf createSchema.sql
```

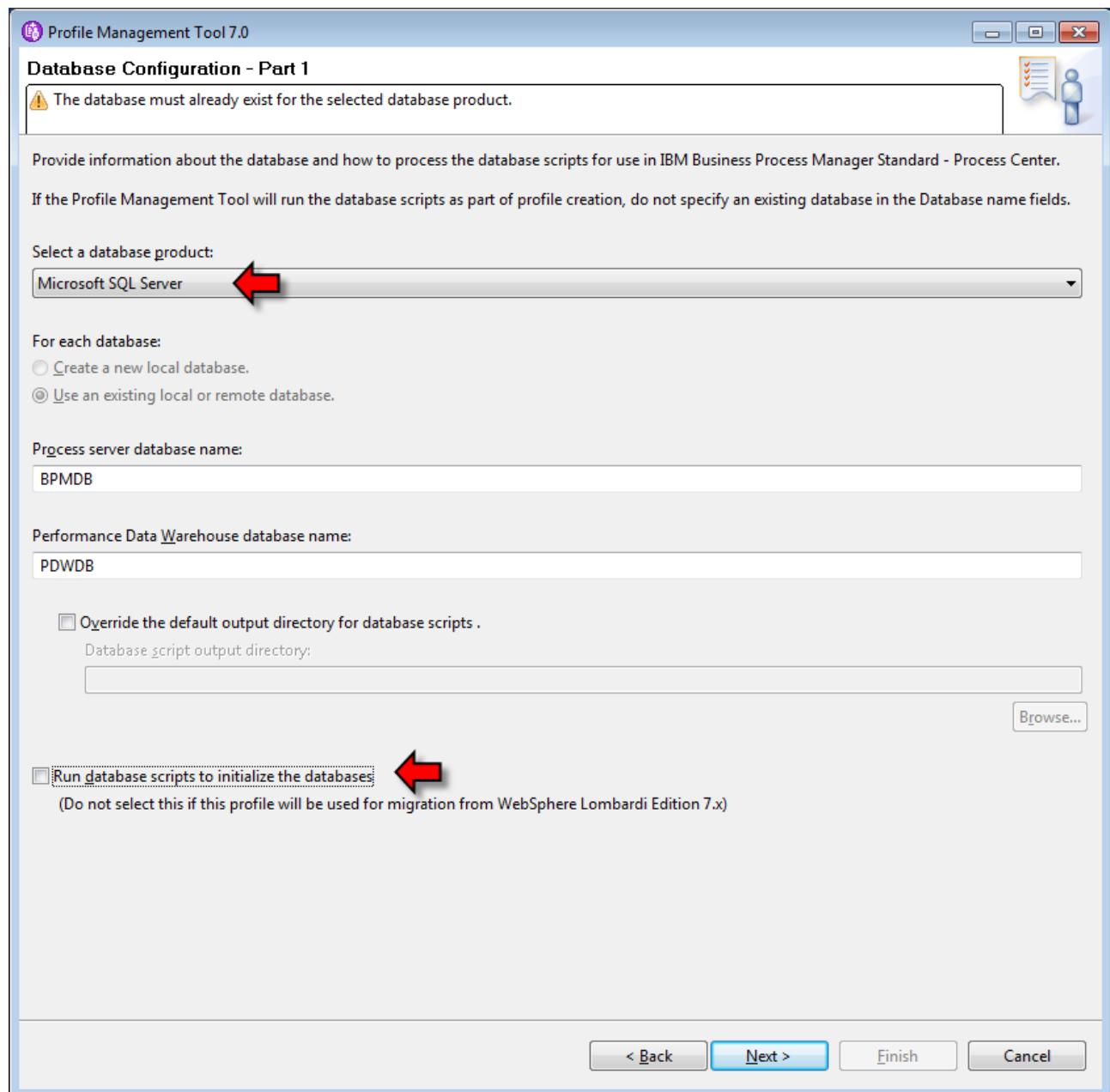
Finally see the section on common steps for all database types. A sample Unix script for running all these commands together might be:

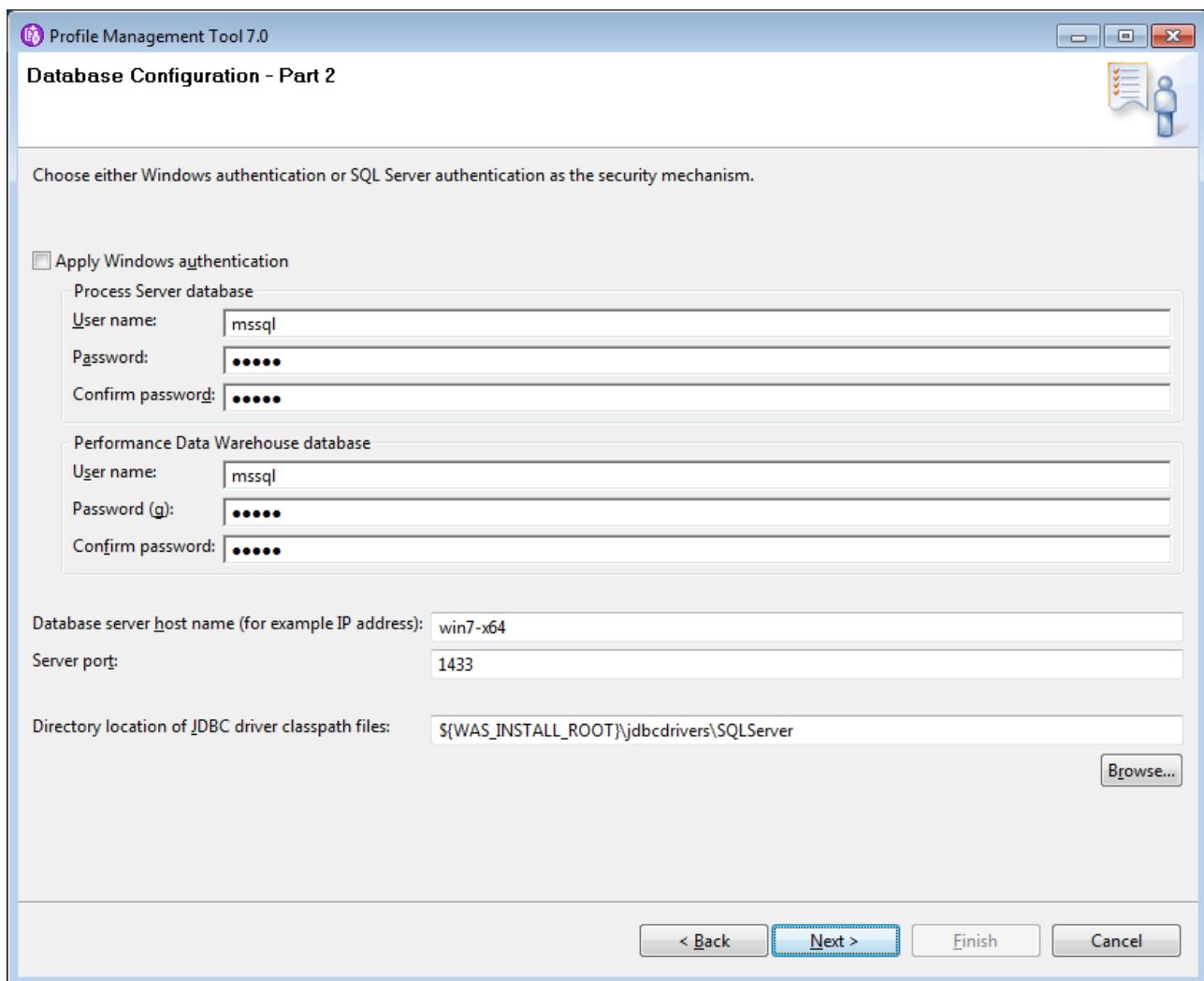
```
# Run from dbscripts folder  
HERE=$PWD  
DBUSER=db2admin  
NODESERVER=emoNode01_server1  
  
db2 drop database CMNDB  
db2 drop database PDWDB  
db2 drop database BPMDB  
  
cd $HERE/CommonDB/DB2/CMNDB  
../configCommonDB.sh createDB  
db2 terminate  
  
cd $HERE/PerformanceDW/DB2/PDWDB  
db2 -tvf createDatabase.sql  
db2 connect to PDWDB user $DBUSER  
db2 -tvf createTable_PerformanceDW.sql  
db2 terminate  
  
cd $HERE/ProcessServer/DB2/BPMDB  
db2 -tvf createDatabase.sql  
db2 connect to BPMDB user $DBUSER  
db2 -tvf createTable_ProcessServer.sql  
db2 -tdGO -vf createProcedure_ProcessServer.sql  
db2 terminate  
  
cd $HERE/ProcessChoreographer/DB2/CMNDB/BPEDB  
db2 -tvf createSchema.sql  
  
cd $HERE/BusinessSpace/$NODESERVER/DB2/CMNDB  
../configBusinessSpaceDB.sh  
  
cd $HERE/..../bin
```

```
./bootstrapProcessServerData.sh
```

This script seems to be needed to run as a userid with write permissions in the <Install> directories of WAS.

MSSQL

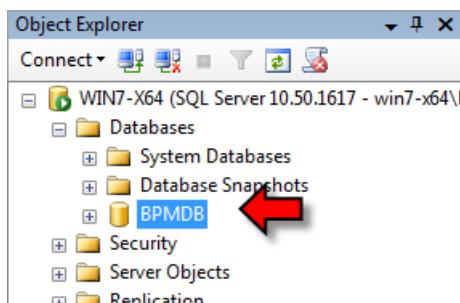




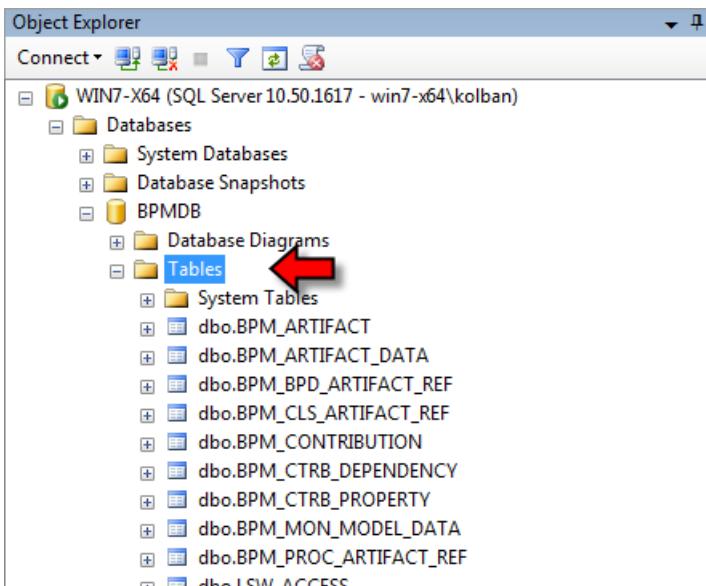
For Process Server

createDatabase	12/5/2011 1:00 PM	Microsoft SQL Ser...	1 KB
createProcedure_ProcessServer	12/5/2011 1:00 PM	Microsoft SQL Ser...	12 KB
createTable_ProcessServer	12/5/2011 1:00 PM	Microsoft SQL Ser...	201 KB
wle_upgradeSchema_ProcessServer	12/5/2011 1:00 PM	Microsoft SQL Ser...	64 KB

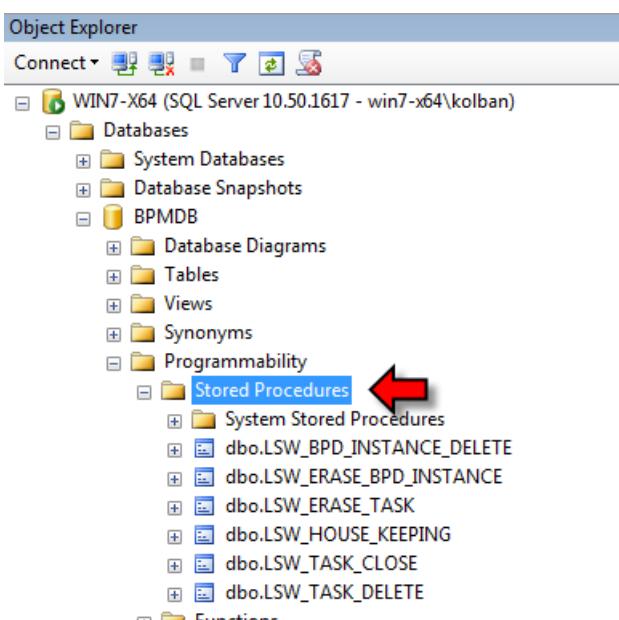
- `createDatabase` – This will create the database required for Process Server operation. At the completion, it may look as follows:



- `createTable_ProcessServer` – This will create the tables required for Process Server operation. At the completion, it may look as follows:



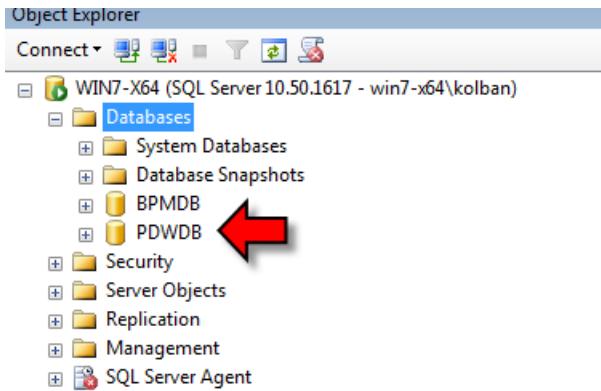
- `createProcedure_ProcessServer` – This will create the stored procedures for Process Server operation. At the completion, it may look as follows:



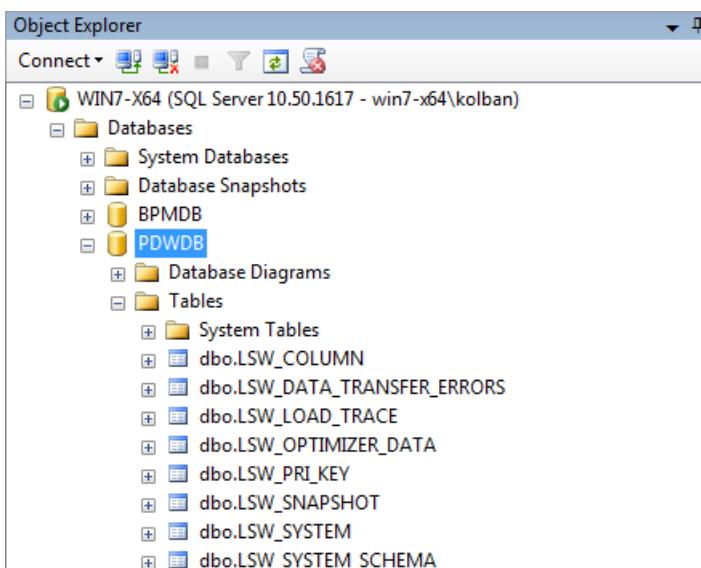
For Performance Data Warehouse

<code>createDatabase</code>	12/5/2011 1:02 PM	Microsoft SQL Ser...	1 KB
<code>createTable_PerformanceDW</code>	12/5/2011 1:02 PM	Microsoft SQL Ser...	23 KB
<code>wle_upgradeSchema_PerformanceDW</code>	12/5/2011 1:02 PM	Microsoft SQL Ser...	9 KB

- `createDatabase` – Creates the database needed for Performance Data Warehouse operation. After execution, it may look as follows:



- `createTable_PerformanceDW` – Creates the tables needed for Process Server operation.
After execution, it may look as follows:



Common for all Databases

After creating the databases and tables, the tables must be populated. Run the command `bootstrapProcessServerData` from the `<Profiles>/<Profile Name>/bin` directory. This needs to be run from a DOS command prompt. It will take a few minutes to execute. It is vital that the configured userid have sufficient permissions to perform this task. For Unix systems this appears to be a user with the ability to write into the `<Profile>` directory.

Operation of the Process Server requires that there be SI Bus definitions. These are made during configuration. However, the databases needed to host these Messaging Engine Data Stores are not necessarily automatically created. If messages appear about message buses not being started, it may be that the following recipe needs to be executed.

An IBM BPM Standard server needs two SI Buses for its operation. These are:

- PERFDW.<Cell Name>.Bus
- PROCSV.R.<Cell Name>.Bus

By default, these are configured to use BPMDB and PDWDB databases. They are also configured

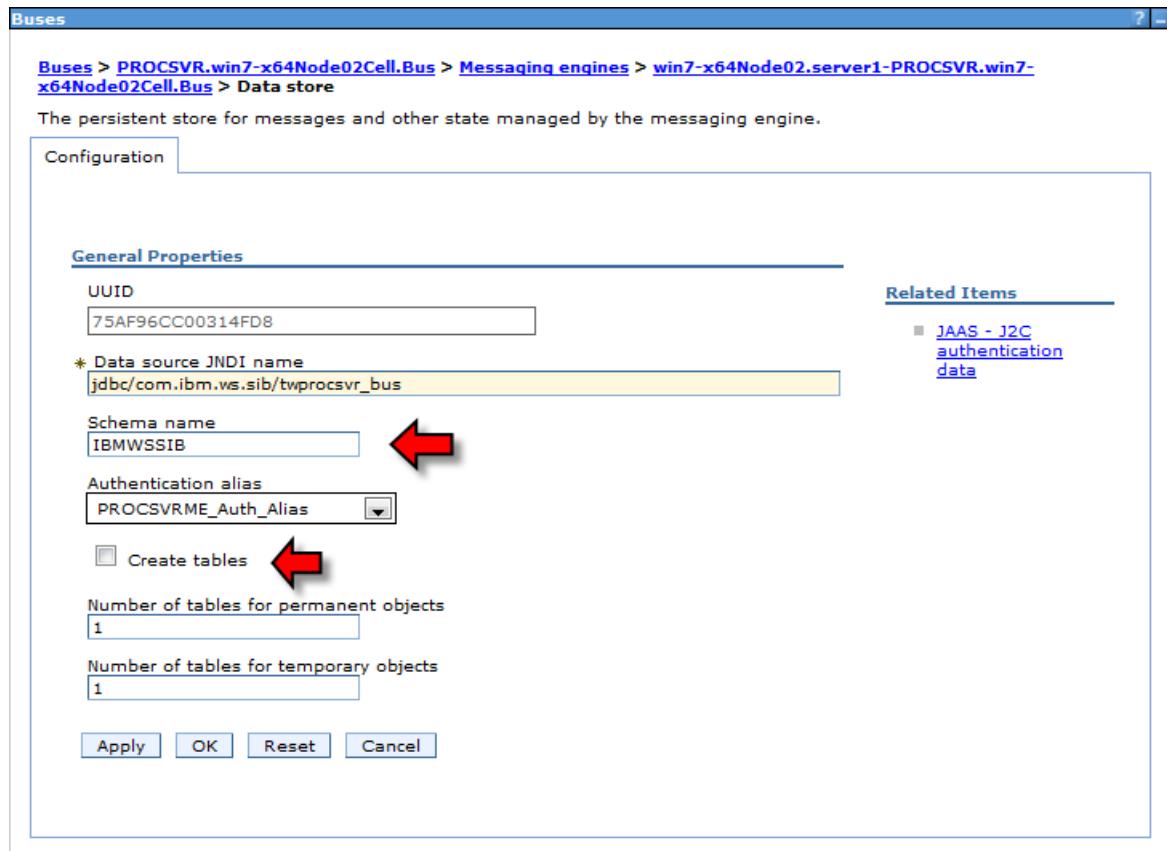
to attempt to create the databases and tables which, in this story, won't work. As such, some work is required before starting the server. WAS provides a command called `sibDDLGenerator` which is used to generate the DDL required to create the tables. Run the following command to generate the DDL:

```
sibDDLGenerator.bat -system sqlserver -version 2008 -platform windows -create -user mssql > C:\temp\sib.sql
```

Next, have the DBA run this DDL against **both** the PROCSV database and the PERFDW database.

Now, we can start the server **but** it will **not** function correctly. We need to make some further administrative changes that can only be made while the server is running. Start the WAS Admin console and navigate to Service Integration > Buses > PROCSV.<Cell Name>.Bus > Messaging engines > <Messaging Engine Name> > Message Store.

Change the schema name to be IBMWSSIB and un-check the "Create tables" check box.



Save and repeat this for the PERFDW bus. Stop and restart the server. If all has gone well, the Buses should show "green" when the server is re-started.

Buses > PERFDW.win7-x64Node02Cell.Bus > Messaging engines

A messaging engine is a component, running inside a server, that manages messaging resources for a bus member. Applications are connected to a messaging engine when they access a service integration bus.

Preferences

Start Stop

You can administer the following resources:

Select	Name	Description	Status
<input type="checkbox"/>	win7-x64Node02.server1-PERFDW.win7-x64Node02Cell.Bus		

Total 1

For the Performance Data Warehouse, there are stories about a command called `configurePerfDW` but I have not yet investigated this area.

Installing Process Designer

The Process Designer component is used to build and test BPMN based processes. It can be downloaded directly from an already running instance of Process Center. With Process Center running, go to its main page in the browser:

IBM Business Process M... X

localhost:9081/ProcessCenter/repository/com.lombardisoftware.repository.Repository.jsp

Process Apps Toolkits Servers Admin

Logged in as Neil Kolban | Preferences | Logout

Sort By: Recently Updated All Favorites Archived

- Procurement Sample (STPPS1)
- Hiring Sample Advanced (HSAV1)
- Hiring Sample (HSS)
- Saved Search Admin (SSA)
- Process Portal (TWP)

Create New Process App
Import Process App
Download Process Designer
Launch Getting Started

Process Applications

Process applications are containers in the Process Center repository for the process models and supporting implementations created in IBM Process Designer. Ordinarily, a process application includes process models, the services to handle implementation of activities and integration with other systems, and any other assets required to run the processes.

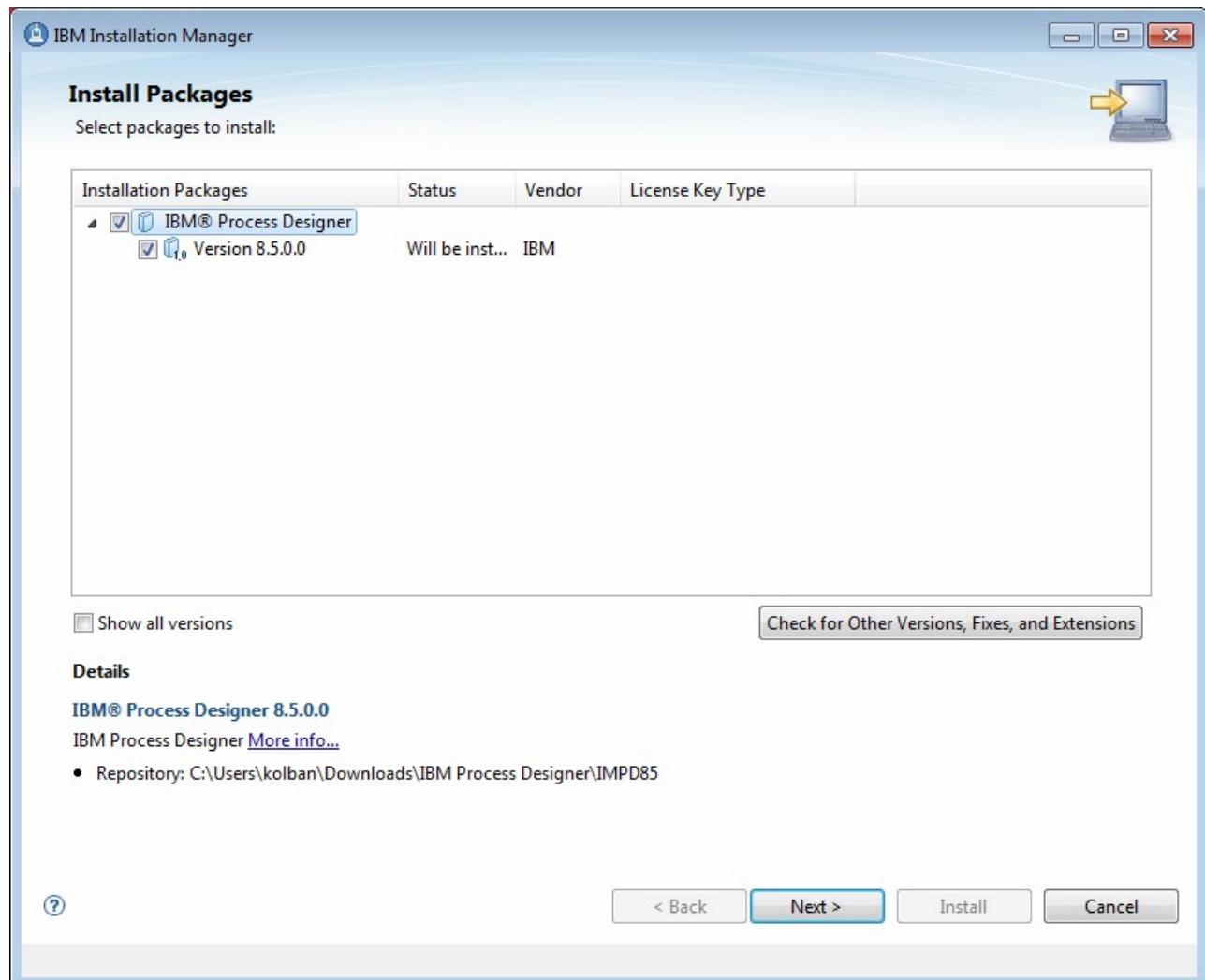
[Managing process applications](#)

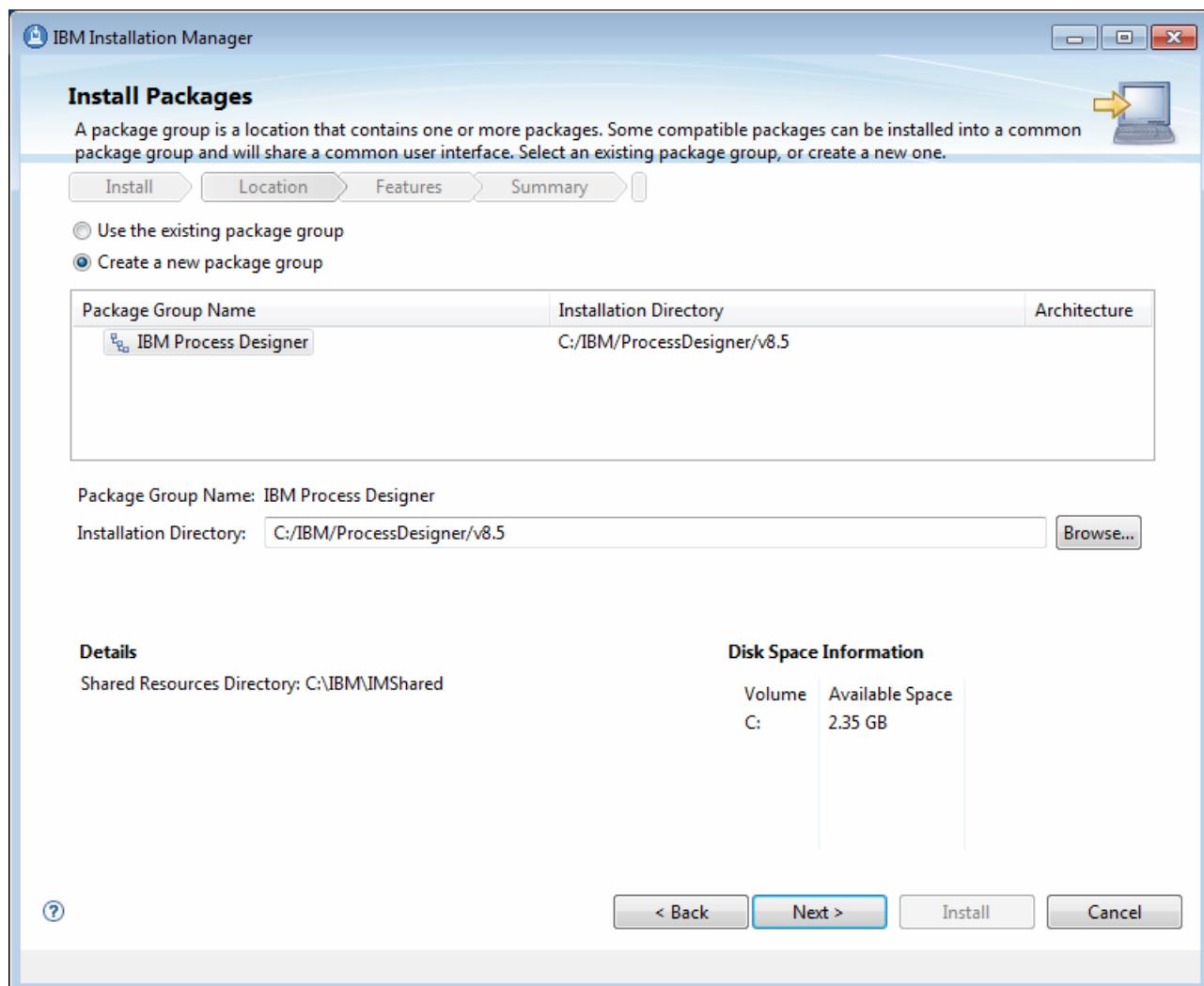
From there, a link on the right can be found called Download Process Designer. Click on this and download/save the Process Designer installer. The installer for Process Designer is nearly 600 MBytes in size. The download arrives as a ZIP. When unzipped and the installer run, the Process Designer is installed as a native application using IBM Installation Manager.

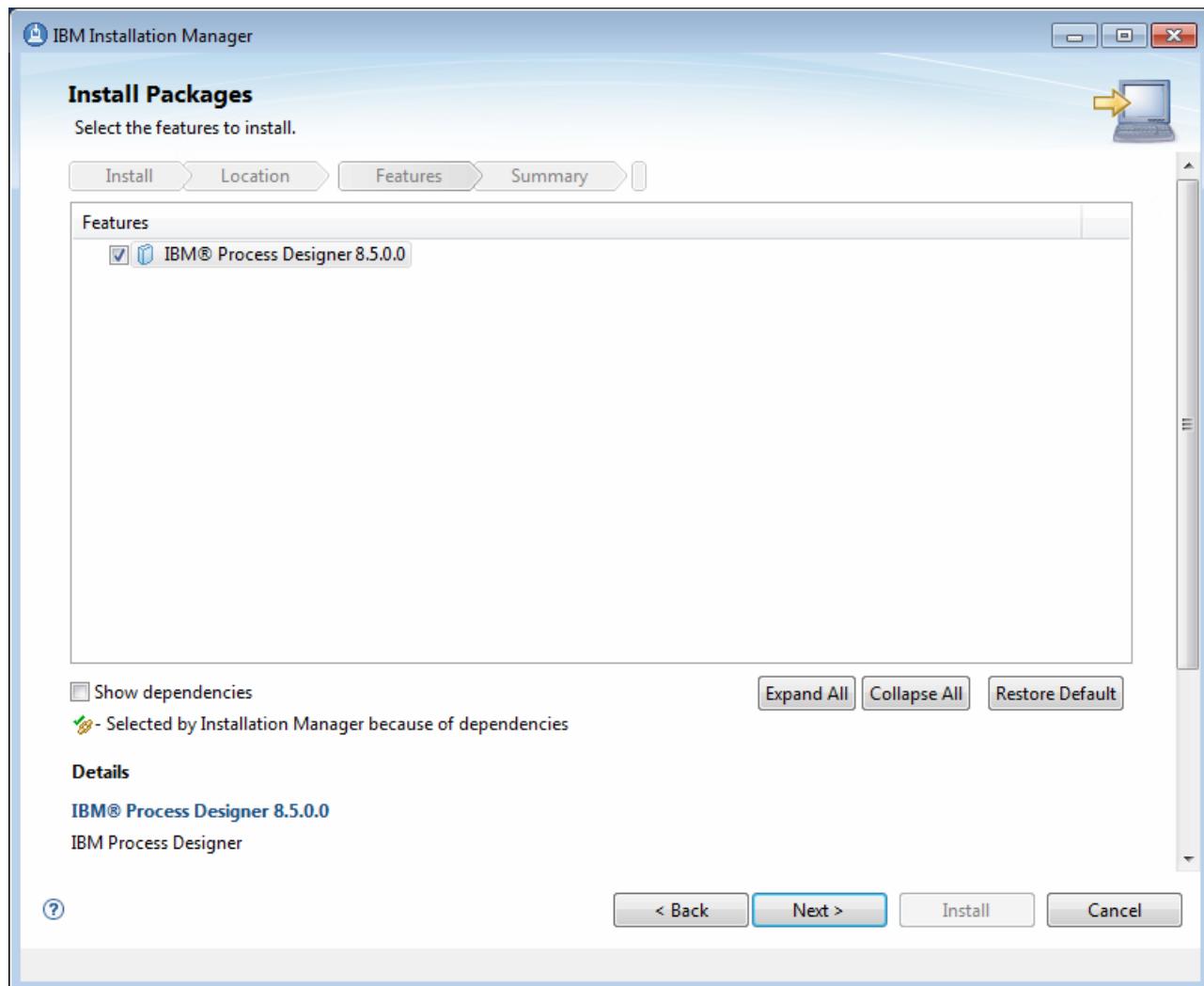
Name	Date modified	Type	Size
IM	6/19/2013 2:06 PM	File folder	
IM64	6/19/2013 2:07 PM	File folder	
IMPD85	6/19/2013 2:07 PM	File folder	
eclipse	6/19/2013 2:07 PM	Configuration sett...	1 KB
installProcessDesigner_admin	6/19/2013 2:07 PM	Windows Batch File	10 KB
installProcessDesigner_nonadmin	6/19/2013 2:07 PM	Windows Batch File	9 KB
trust	6/19/2013 2:07 PM	Personal Informati...	3 KB

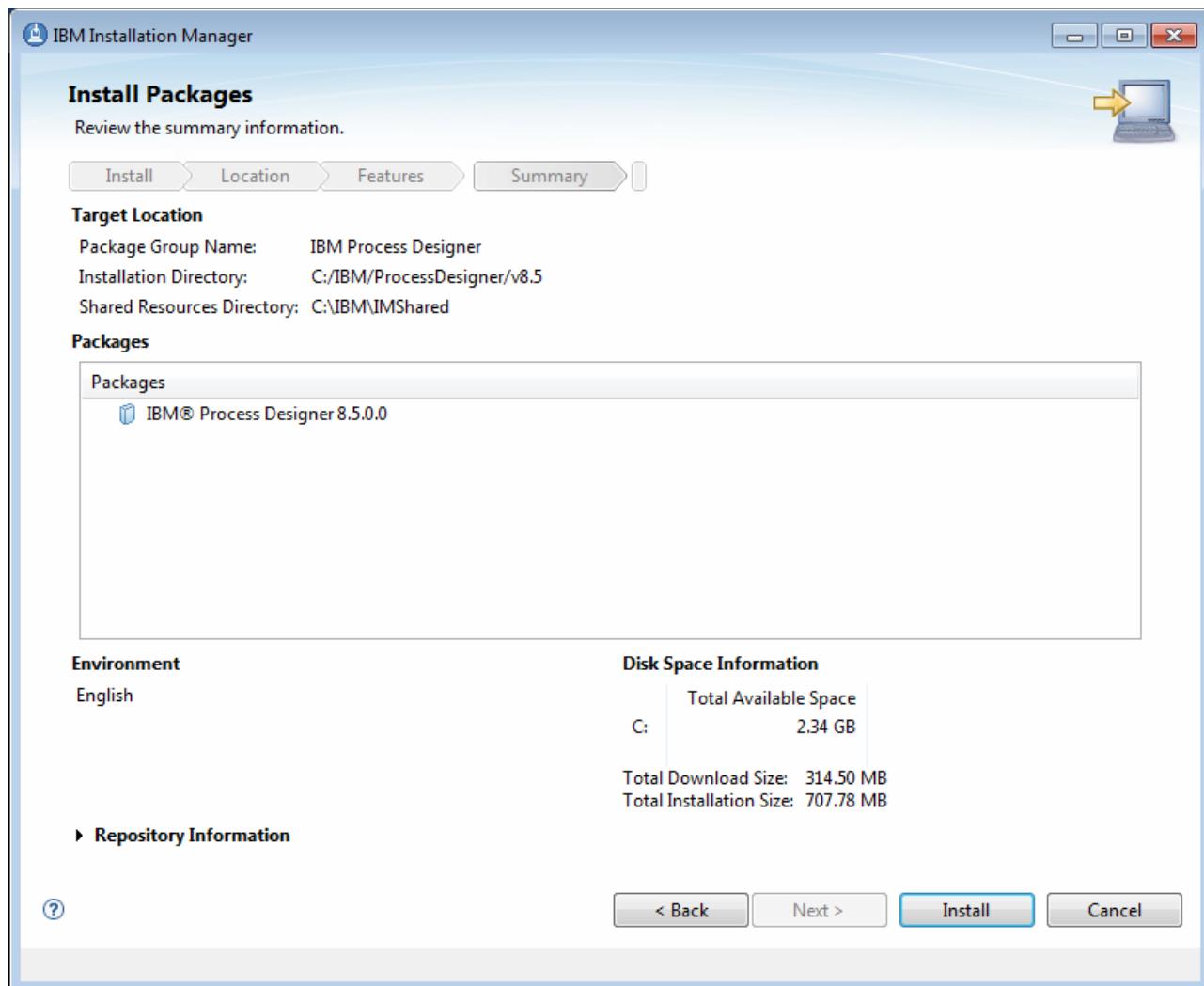
The installation happens silently. After installation, launch Installation Manager to apply maintenance.

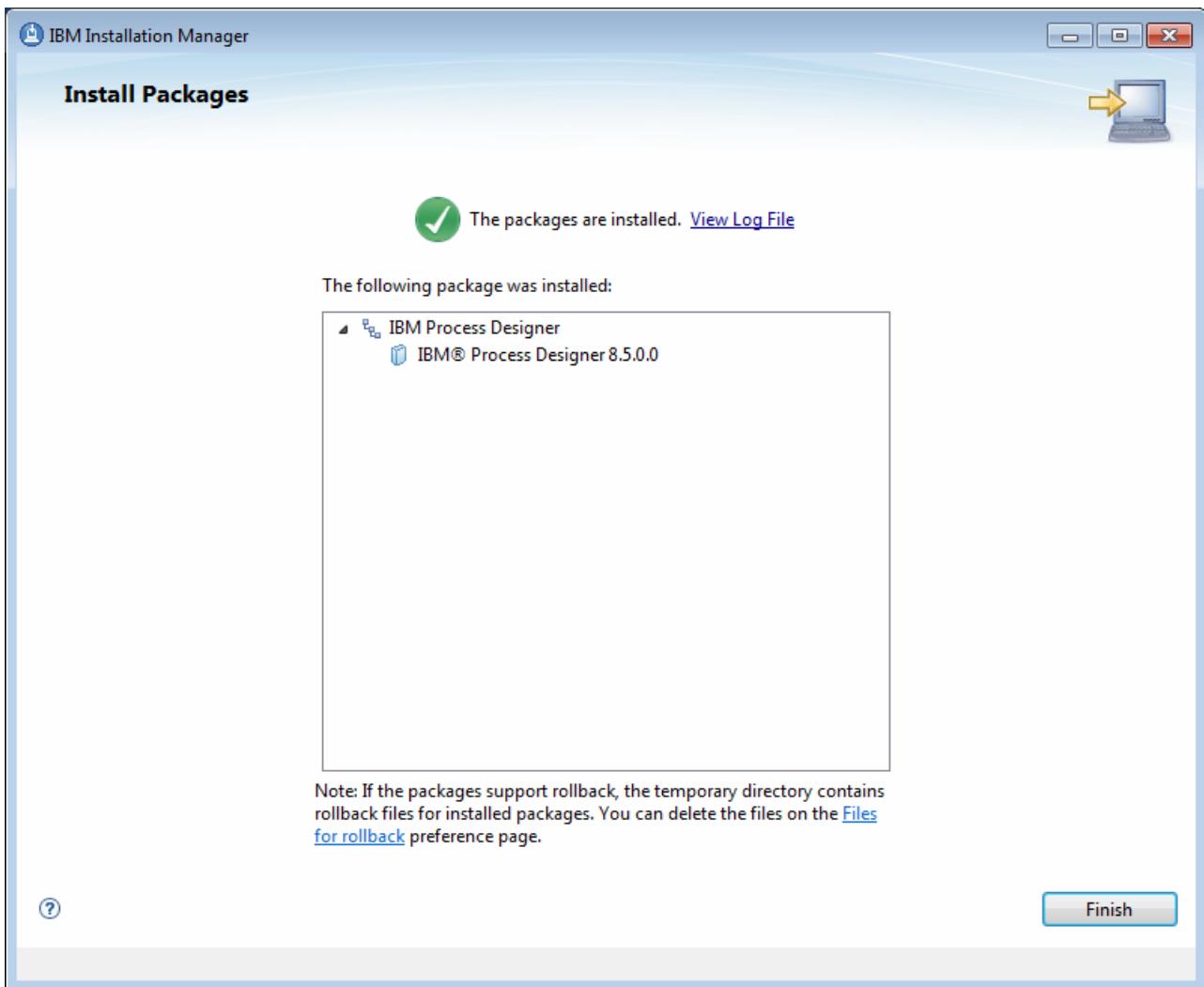
Alternatively, Process Designer can be installed through Installation Manager:











Starting BPM

Once BPM is installed, we can start it. There is an order to starting the environment. This is:

- Deployment Manager
- Node Agent
- Deployment Environment

For the deployment manager, use the "startManager" command found in the <DmgrProfile>/bin directory. It seems to take about 5 minutes to start.

For the Node Agent, use the "startNode -profileName <NodeProfile>" command found in the <Root>/bin directory. It seems to take about 2 minutes to start.

For the deployment environment, use the "BPMConfig -start <propertiesFile>" command. The log file for this will be found in <Node1Profile>/logs/<SingleClusterMember1>.

Stopping BPM

BPM can be stopped by shutting down the components in the following order

- Deployment Environment
- Node Agent
- Deployment Manager

For the Node Agent, use the "stopNode" command found in the <Root>/bin directory.

For the deployment manager, use the "stopManager" command found in the DmgrProfile bin directory. Use the Cell Admin userid/password.

Resetting the databases

The databases used by the product contain the repository of the Process Center. Should we desire, we can *reset* these databases. This is a solution that is very extreme but will result in a reset system (back to factory defaults).

The content of the Process Center database can be populated with the command called:

`bootstrapProcessServerData.bat`

This command is fully documented in the InfoCenter. The scripts to create the tables needed for operation can be found in:

`<WAS Profile>/dbscripts`

Un-configuring BPM

If we wish to un-configure BPM, we should consider what that would mean. Specifically, it would mean the deletion of the WAS profiles plus any databases used by the product.

To see what profiles exist, run "`manageprofiles -listProfiles`"

To delete a profile run "`manageprofiles -delete -profileName <name>`"

On a typical configuration, the profiles would be:

- Node1Profile
- DmgrProfile

After deleting a profile using `manageprofiles`, also remove any directories for that profile found at <Root>/profiles. There may be directories with the same names as the profiles just deleted.

In addition, delete any log files found at:

- <Root>/logs/manageprofiles
- <Root>/logs/config

which pertain to the profiles just deleted. Examples that may be deleted include:

- Directory with same name as profile
- <profile>_create
- <profile>_delete

Hiding the sample applications

After installation and configuration of the product, some sample applications can be seen in the

Process Center. These also show up in the Process Portal as startable processes. It is likely that you will want to hide these so that they don't clutter up your environment. These samples are:

- Procurement Sample (STPPS1)
- Hiring Sample Advanced (HSAV1)
- Hiring Sample (HSS)

These samples can be hidden by archiving the process apps from within Process Center.

See:

- Archiving Process Applications

Frequently Asked Installation Questions

- **Q:** What other Java EE server types are supported?
A: Only IBM WebSphere Application Server is supported. Java EE servers from other vendors are explicitly **not** supported and simply will **not** work.
- **Q:** Can I record the options of IBM Installation Manager so that I can then later re-supply the same attributes?
A: IBM Installation Manager has a "-skipInstall" option that goes through the prompts but does not actually perform the installation. The results can then later be used to perform a silent install.
- **Q:** How can I tell what version of code the server is running?
A: Examining the start of the SystemOut.log console file will show the details of the server. Here is the first line of an example:

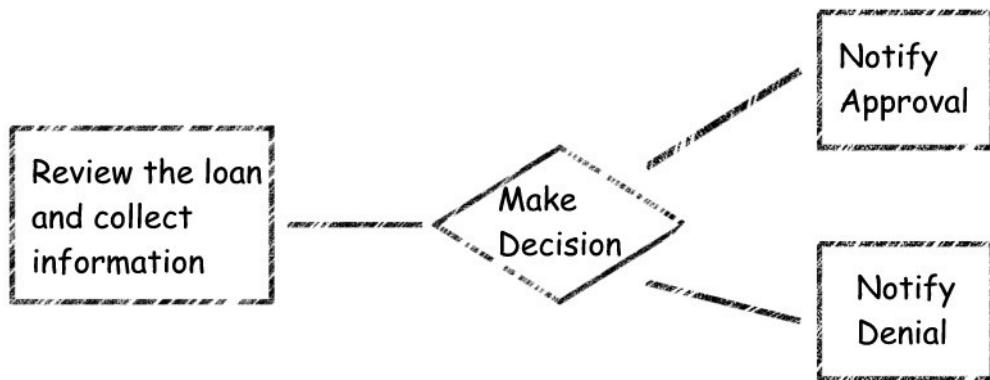
```
WebSphere [BPMPC 7.5.1.0 o1145.08] [SCA 1.0.1.11  
cf111118.05] Platform 7.0.0.19 [ND 7.0.0.19 cf191132.09] [XML  
1.0.0.9 cf091117.04] running with process name kolban-  
PCNode01Cell\kolban-PCNode01\server1 and process id 3500
```

- <more>

Business Process Definition - BPDs

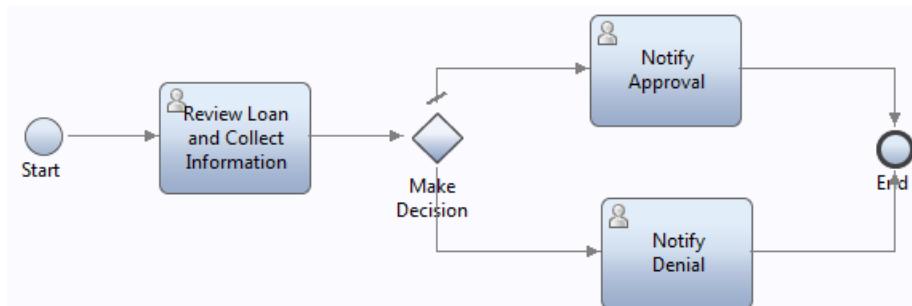
The Business Process Definition (BPD) is the core of an IBM BPM solution. The BPD represents the model of the business process and is captured early on in the design of the solution. It also allows the process to be easily modified as time passes. At a high level, a BPD is a diagram that represents the overall view and flow of the process. It is commonly constructed by the Business Process Analyst in conjunction with discussions and interviews with other members of staff.

Here is an example of a napkin sketch of a trivial business process for approving a loan. We could imagine a meeting room with a white board where the process of how a loan is handled is discussed. The flowchart like blocks could be drawn on a white board.



In IBPM, the exact same process can be captured as a Business Process Definition in the IBPM Process Designer (PD) tooling. It captures the exact same essence of the process. In fact, it is arguably easier to capture the process in IBPM PD as blocks can be moved around without having to erase the picture and redraw it.

We will further see later that participants at the meeting where the process is being sketched can be remote from each other and yet the same process diagram can be seen on each of their screens.

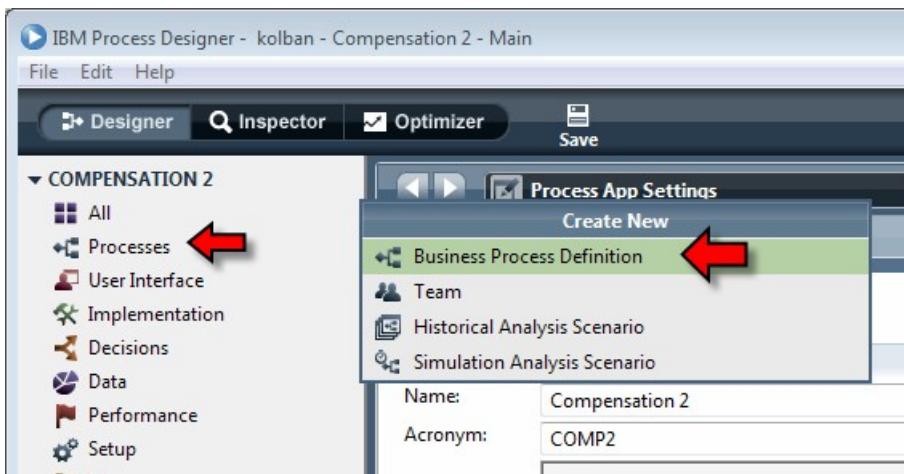


The diagram style of a Business Process Definition conforms to the industry standard Business Process Model and Notation (BPMN) format.

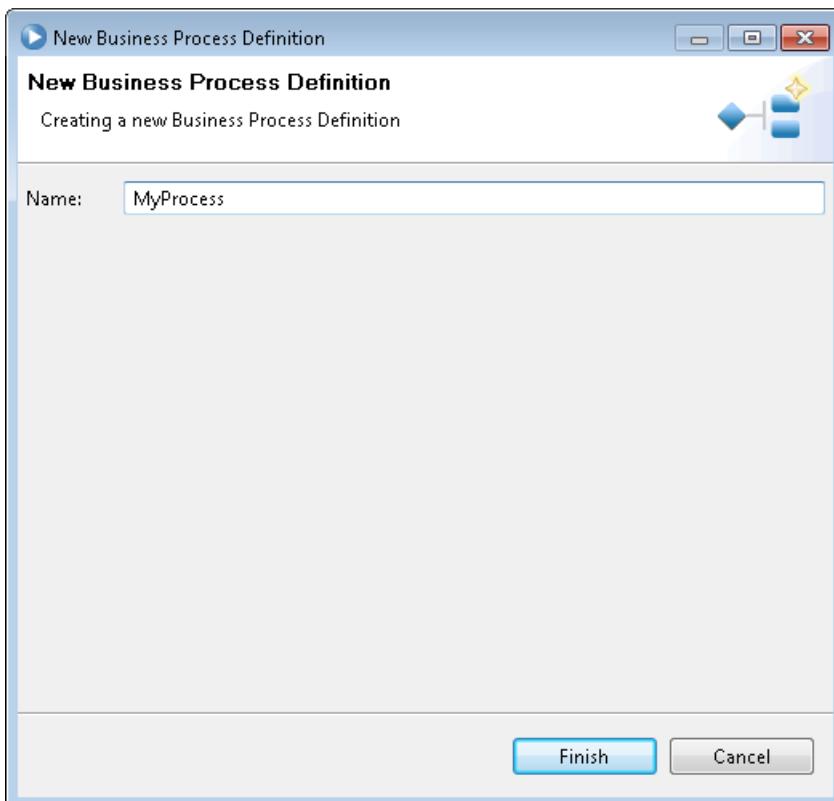
See also:

- Modeling a process

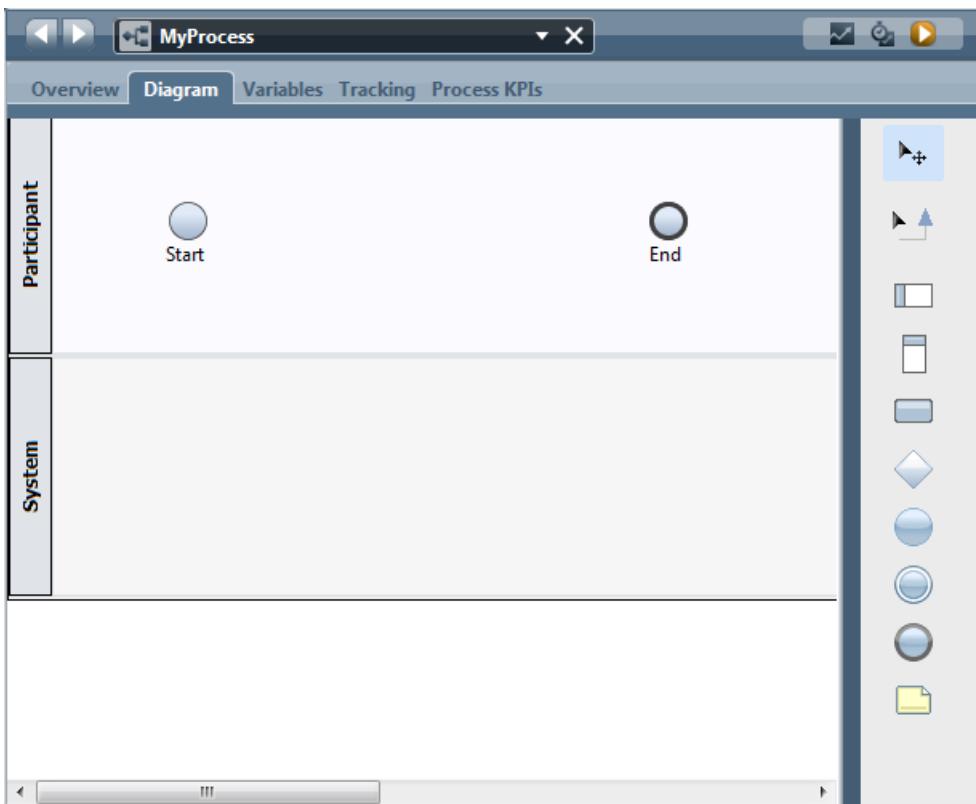
When a Business Process Definition is created it is given a name and lives inside a Process Application solution. A Business Process Definition is created inside the IBPM PD tooling from the library section within the Processes category:



The wizard for the Business Process Definition (BPD) asks for the name of the new BPD which is to be created.



Once created, the main canvas area of the BPD editor is shown:



This main area of the window is called the *canvas* and is where we visually model the process solution we wish to construct. Icons representing the activities and other elements of the process can be dragged from the palette of available types shown on the right of the window. These icons can be dropped onto the canvas and wired into place using Sequence Lines. Each icon represents a "step" in the process.

The icons conform to the Business Process Model and Notation (BPMN) style and their interpretation is also that of BPMN.

The palette of available icons are:

- Activity – see Activities
- Lane – see Pools and Lanes
- Milestone – see Pools and Lanes
- Decision Gateway – see Gateways, Conditionals and Joins
- Note
- Start Event – see Start Event
- End Event – see End Event
- Intermediate Event

Contained within these major palette entries are the other BPMN components:

	Task Types
	User Task
	System Task
	Decision Task

	Script
	Subprocess
	Linked Process
	Event Process
	None
	Gateways
	Exclusive Gateway
	Parallel Gateway
	Inclusive Gateway
	Event Gateway
	Start Events
	None Start Event
	Message Start Event
	Ad Hoc Start Event
	Content Start Event
	Intermediate Events
	Message Intermediate Event (Receive)
	Message Intermediate Event (Send)
	Timer Intermediate Event
	Content Intermediate Event
	Tracking Intermediate Event
	End Events
	None End Event
	Message End Event
	Error End Event
	Terminate End Event

See also:

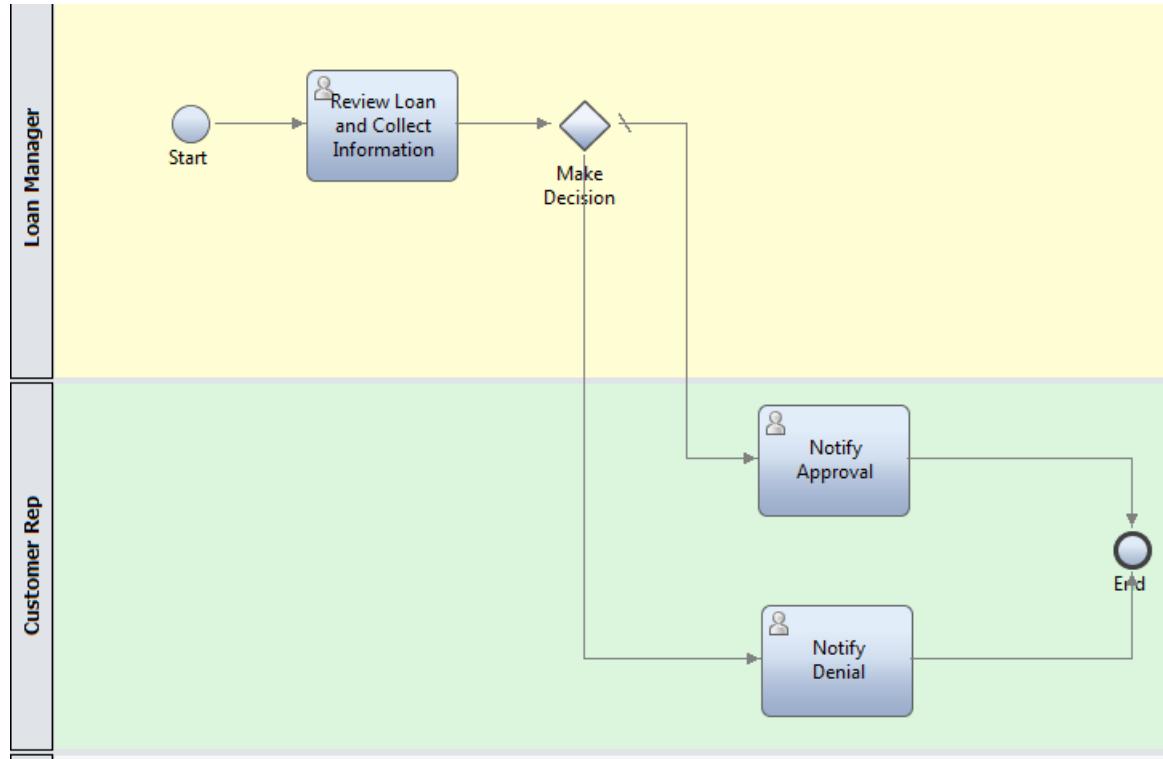
- Sequence Lines

Pools and Lanes

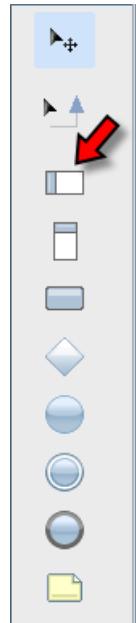
Think of an Olympics swimming competition where multiple swimmers swim against each other for the fastest time. Each swimmer swims in the same swimming pool but is constrained to swim only in their own swimming lane. The canvas drawing area of the BPD also contains such a model. The canvas as a whole represents the pool while multiple lanes can be added to segment the pool into horizontal sections. When an icon is added to the diagram, its vertical position on the diagram means that it "lives" in only one lane. This serves two purposes. First, if we associate a lane with a

particular "role" in the process, then we can quickly see simply by looking along the row of the lane which activities are associated with which roles. Conversely, if we assign attributes to a particular lane, then activities found in that lane can inherit those attributes.

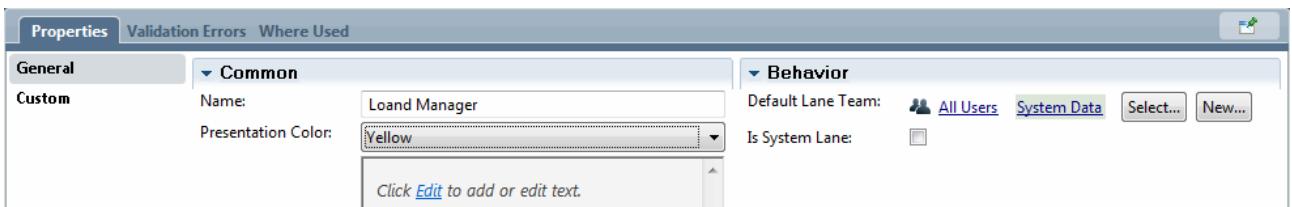
Here we see an example of a BPD pool with a couple of lanes:



Additional lanes can be added by dragging a lane icon from the tools palette:



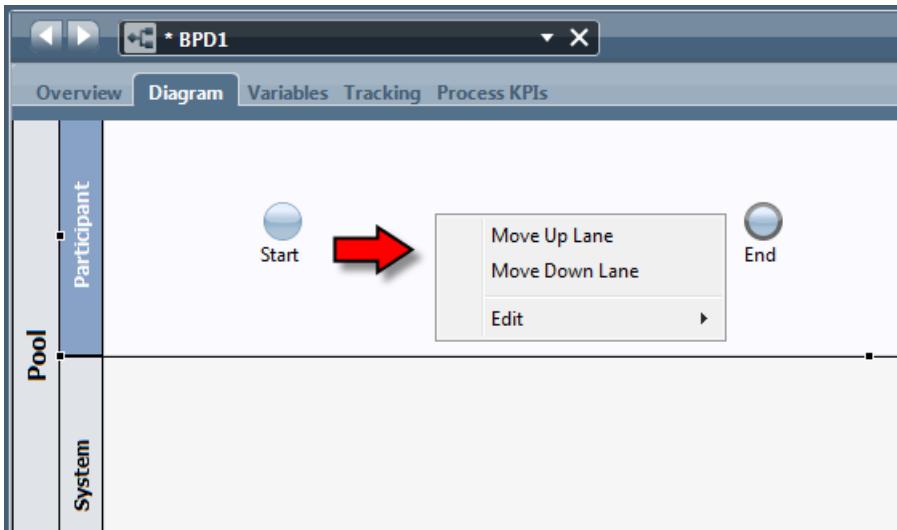
When a lane is added to the diagram we can specify some primary attributes:



Every lane has a Name attribute so we should supply a meaningful name that will remind us of its purpose. A Presentation Color can be assigned which will change the background color of the lane. This can be especially useful if you have many lanes and need to read to the far right in order to see all of the process content. The Name of the lane may not always be immediately visible but if colored correctly, will highlight its intent.

For human services, we can assign the set of users that should be associated with the lane. This set of users is known as a Team. When a human service is added as a step on the lane, if no specific human routing instructions are supplied, then the human service will be routed to the members of the Team associated with the lane.

We can re-order the lanes in a pool from their existing values. By right clicking in a lane, a context menu is shown:



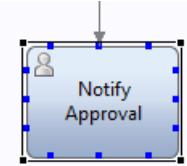
Within that menu we can choose to move the selected lane up or down relative to the other lanes in the pool.

Sequence Lines

The *lines* which are drawn which connect individual steps in a BPD are called *Sequence Lines*. A sequence line describes which activities will follow other activities in sequence as the process executes. Sequence lines have an arrow head point to the next activity to be executed. When using PD, the following icon can be used to place the tool into a mode where Sequence Lines can be drawn:



When in sequence line drawing mode, the cursor shape changes to include a "plug" in its visual representation. This is to represent the idea of wiring (or plugging) one component into another. When the cursor is over an element that can be the source or target of a Sequence Line, the element's representation changes to show attachment points.



These attachment points serve as the anchors for the start and end of a Sequence Line. To wire two elements together, enter wiring mode and hover the mouse over the originating element. Pick one of the attachment points and click and release. Next move the cursor (without pressing any further buttons) over the target element. A line will automatically be drawn from the source to the target. When over the target, attachment points will again be shown. Pick the attachment point that is desired and click and release once more. A Sequence Line will now connect the two elements.

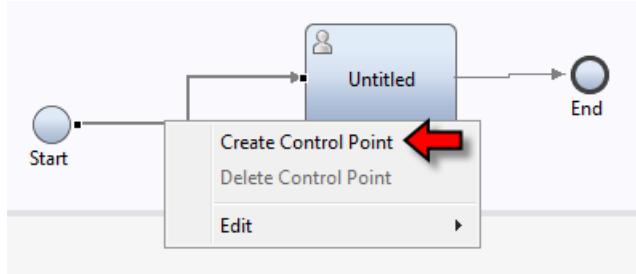
You can leave wiring mode by clicking on the icon that looks like:



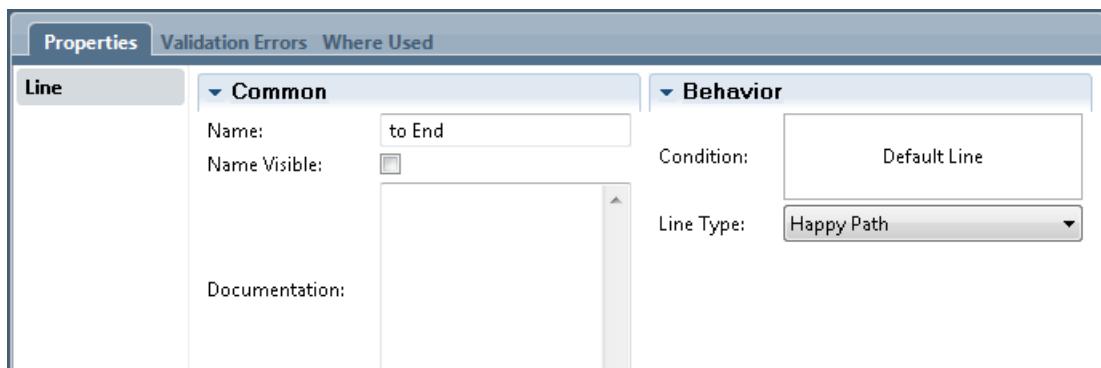
You can tell which editing mode you are in as the current mode icon has a blue background. Alternatively, when in wiring mode, pressing the Escape key on the keyboard will return you to the default editing mode.

A Sequence Line can have its source or target changed. If the line is selected with a mouse click, either endpoint of the line can be dragged to another element.

From the context menu when a Sequence Line is selected, additional "points" on the line called control points can be added. A control point allows the line to have additional junctions in it to further control its visual layout and appearance. This can be used to make the diagram more attractive. The control point can be dragged with the mouse to re-orient the line.

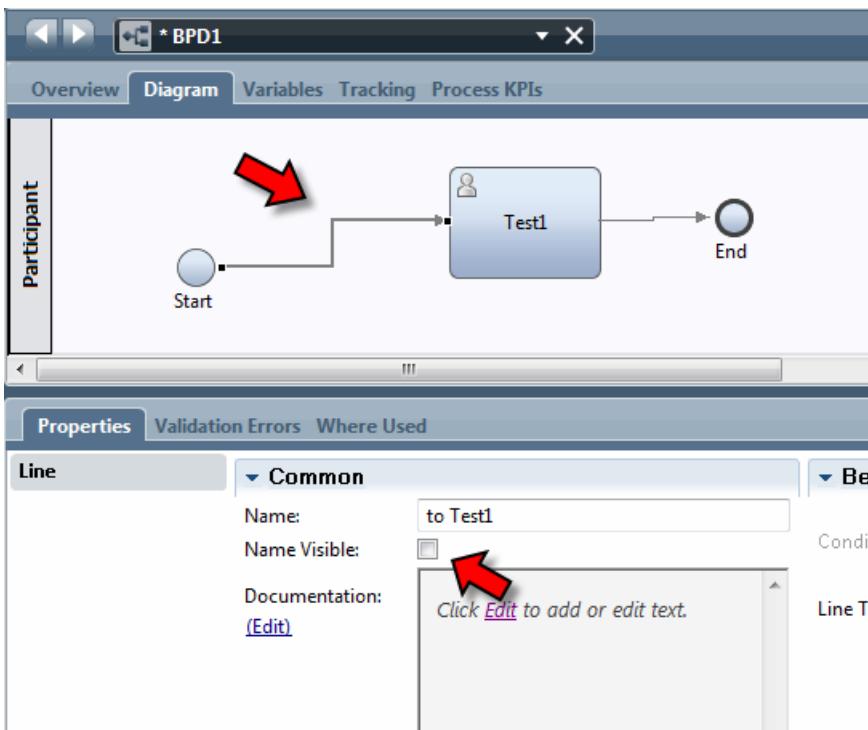


When selected, we can also edit the properties of a Sequence Line:

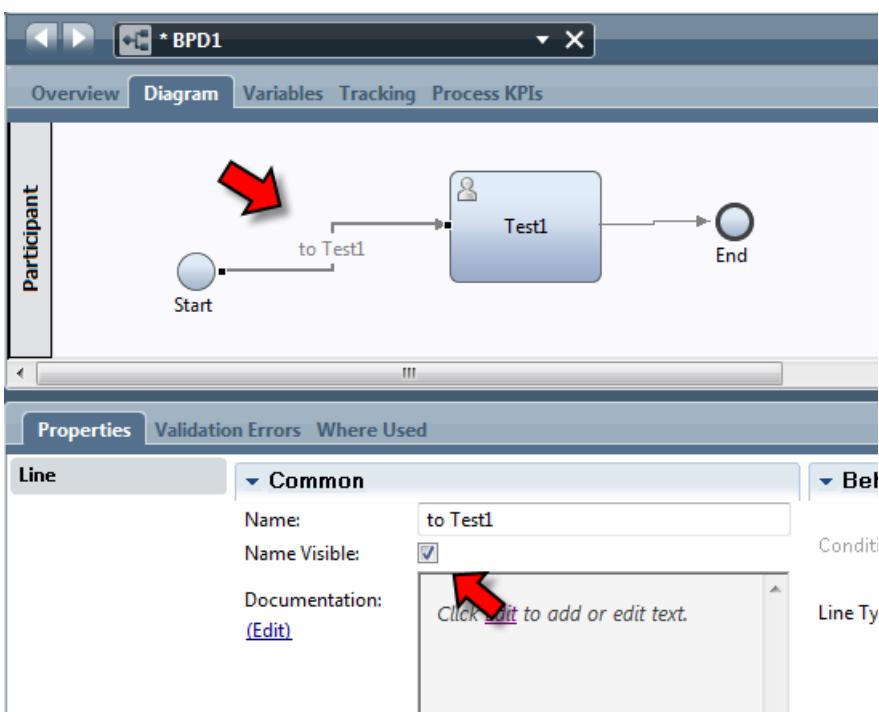


Amongst these properties are the Name that can be given to the line. This name can be made visible and shows on the diagram.

Here we see a diagram with the text of the line not visible:



and here is the same diagram with the name visible:



In addition, if the Sequence Line is used as the output of a condition gateway, the name given to the line is seen in the gateway's description of the conditions that are used to select which of the possible alternative paths are to be taken.

The condition associated with a decision gateway can also be seen on the properties of the line.

Finally there is an attribute called Line Type which can have a value of Happy Path or of Exception Path. These values do not alter the operation of the process but are used exclusively during process analysis. Most of the time they can simply be ignored.

Activities

Without question, the core items in a process are the activities. These are the declarations that some work is going to be performed. From the palette, the activity icon looks as follows:



When added to the canvas, it looks like:



or like depending upon which type of lane the activity is dropped.

The initial properties of an activity look as follows:

Properties Validation Errors Where Used

General

Simulation

Implementation

Assignments

Data Mapping

Pre & Post

KPIs

Condition

Custom

Common

Name: Untitled

Presentation: Color Icon

Presentation Color: Default

Documentation: [Edit](#) Click [Edit](#) to add or edit text.

System ID: bpdid:cd2e6288625ccebf:-7fd

Behavior

Loop Type: None

[Multi Instance Looping](#)

[Simple Looping](#)

Initially, a newly added activity has no implementation. It is merely a "place holder" or a description that something is going to happen here.

Properties Validation Errors Where Used

General

Simulation

Implementation

Assignments

Data Mapping

Pre & Post

KPIs

Condition

Custom

Implementation

User Task

[Default Human Service](#)

[Coaches](#)

[Select...](#)

[New...](#)

Task Header

Clean State:

Subject:

Narrative:

1

Priority Settings

Priority: **js** Normal (default)

Due In: **js** 1 Hours 00:00

Time Schedule: **js** (use default)

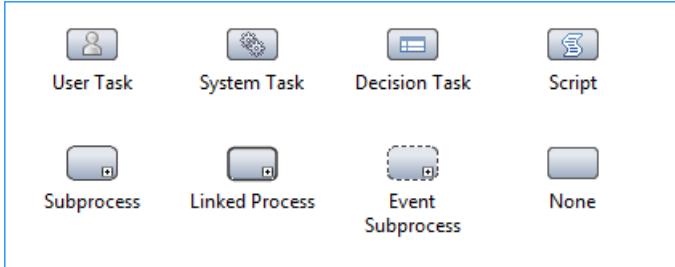
Timezone: **js** (use default)

Holiday Schedule: **js** (use default)

Processing Behavior

Automatically flow to next task:

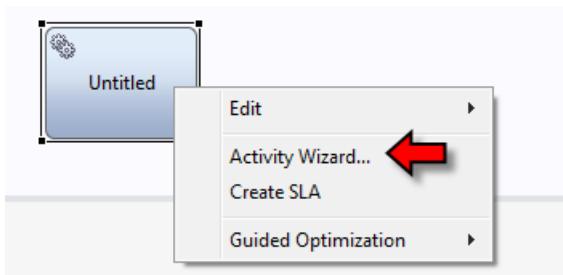
Following the core philosophy of the product that everything about the process is contained within a common or single shared model, information about the implementation of this activity will eventually be entered. IBPM provides a set of defined possibilities for implementing an activity. These are:



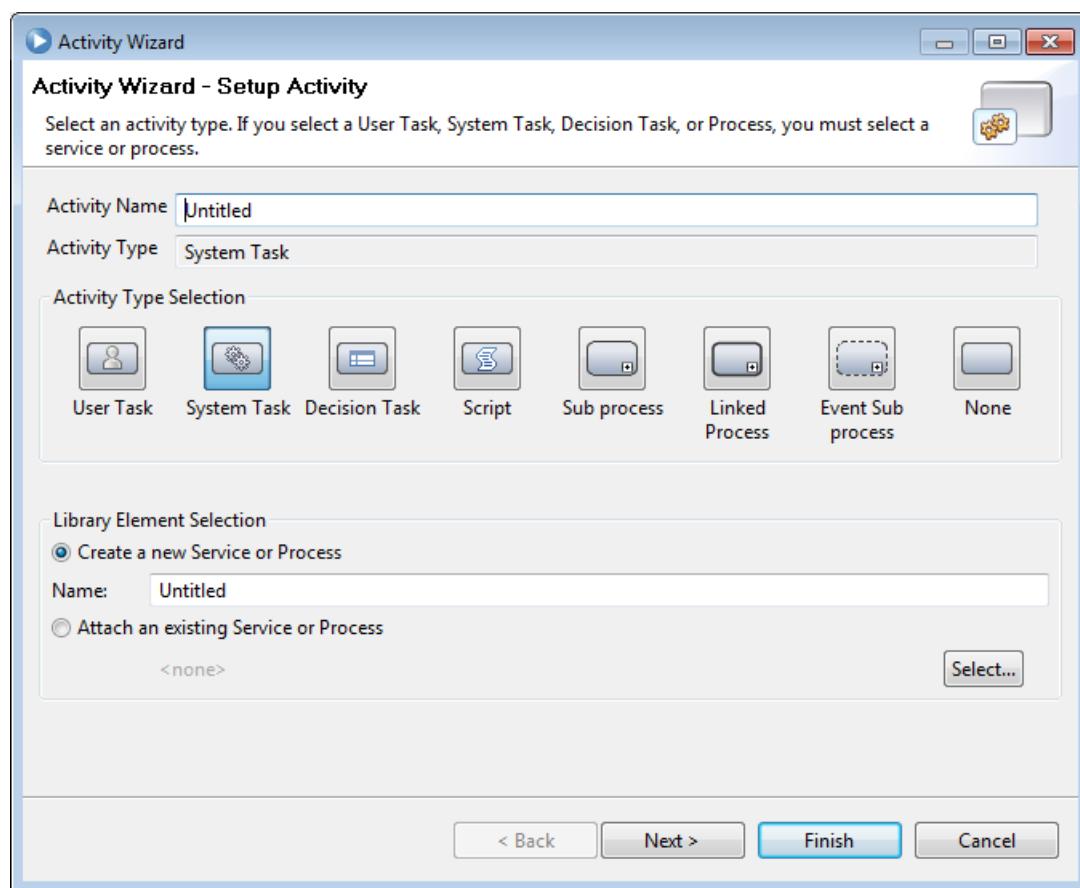
- **None** – No implementation is associated with this activity. This is most useful when the diagram is initially being drawn and we are not yet ready to make a decision on how this step will best be implemented.
- **User Task** – Implemented as a Human Service. This step in the process will be performed by a human being. - - Human Service
- **Decision Task** – Implemented as a decision service where rules will be executed to make a business decision -
- **System Task** – Implemented as a technical or straight through step without human interaction - - A system service
 - Ajax Service
 - Integration Service
 - General System Service
- **Script** – Implemented as a piece or snippet of in-line JavaScript code. This is most appropriate for one-time data manipulation such as initialization or simple copies of data from one variable to another.
- **Sub-process** – Implemented as a sequence of additional activities where those activities are contained within this process. Think of the Sub-process activity like a grouping or container.
- **Linked Process** – Implemented as a call to another process defined separately.
- **Event Sub-process** – Implemented as a sequence of steps that will be performed concurrently with the main process when an external event occurs.

Many of the implementation types of an activity in the BPD require them to be associated with further details in the form of a Service implementation. We can perform this step in either a top-down or bottom-up fashion. Top-down means we start with the notion of an activity and then provide an implementation for that activity. Bottom-up means we start with an activity implementation and then associate that with a BPD activity. Either technique is valid.

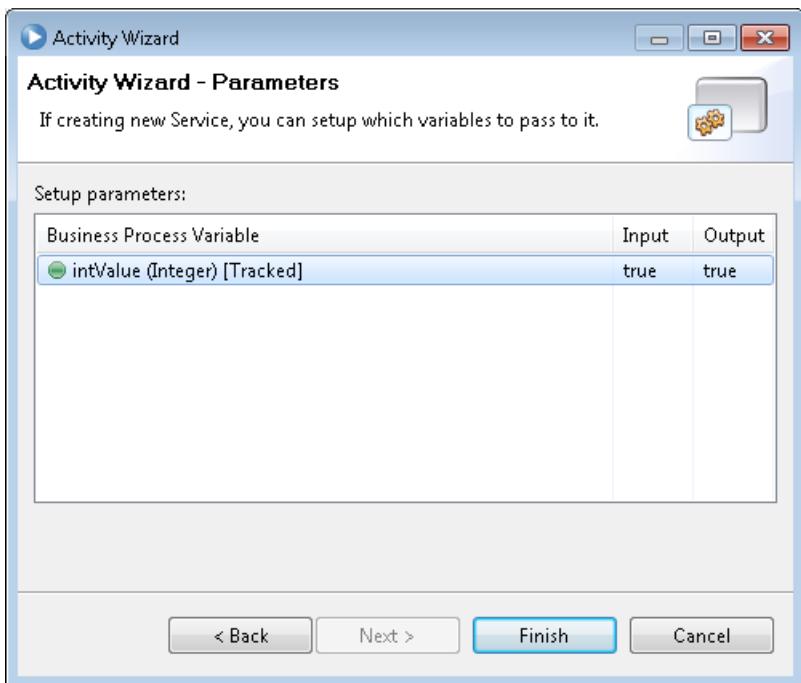
For the most common types of implementation, a wizard is available to assist with the implementation selection/creation. The Activity wizard can be started by right-clicking on an Activity and selecting Activity Wizard...



The wizard allows us to select which type of service to associate with it and, if necessary, create a new service.



If a new service is to be created, the variables in scope in the process are displayed as potential inputs and outputs to the service.

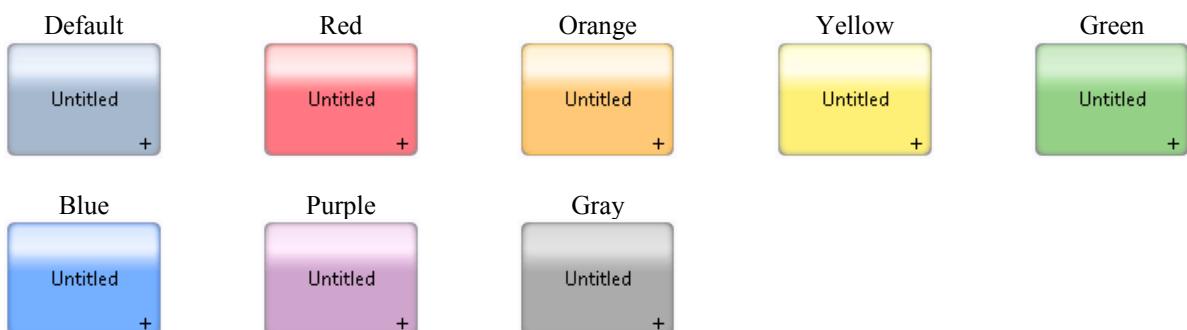


The table cells under the Input and Output columns are click-able and will toggle between true and false. A value of true means that the variable will be an input and/or output parameter to the service while a value of false means that the variable will not be utilized by that service.

Activity Appearance in the BPD diagram

The name of the Activity is used to provide a label for it on the canvas.

The Presentation field allows the activity's appearance to be changed. This does not change its operation but merely changes how it looks to the user who views the BPD diagram. The color entry allows the developer to choose the color used to represent the activity. This could be used to highlight activities based on a convention. For example, red to indicate an activity that interacts with a secure system.



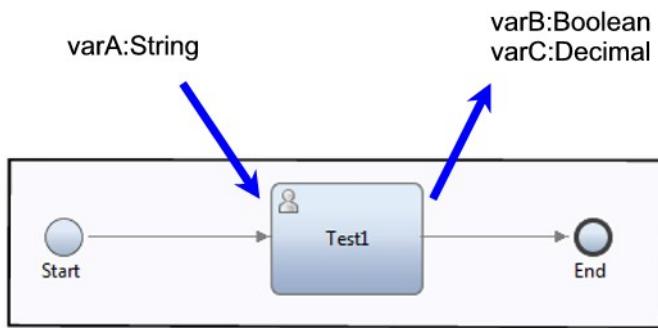
As an alternative to these colored replacements, an Icon representation can be used. Use this with caution/care. One of the key elements of reading a BPD is visual consistency so that a reader can tell very quickly what an diagram means by knowing the set of icons commonly found on such a

diagram. If the icons are changed, the reader may not know as quickly what is intended. By clicking the icon radio button next to Presentation, the file name of a new icon can be supplied.

Data Mapping

Many activities have a Data Mapping section. This is the act of mapping the variables in scope in the BPD to the expected parameters of the service that is to be called by the Activity. This applies to both input parameters and output parameters.

The Data Mapping screen area is split into two columns. One for input mapping (mapping BPD variables to the parameters expected by a service) and the other is for output mapping (mapping returned values from a service to the variables in the BPD). Entry assistance is available to bring up lists of BPD scope variables which can be selected.



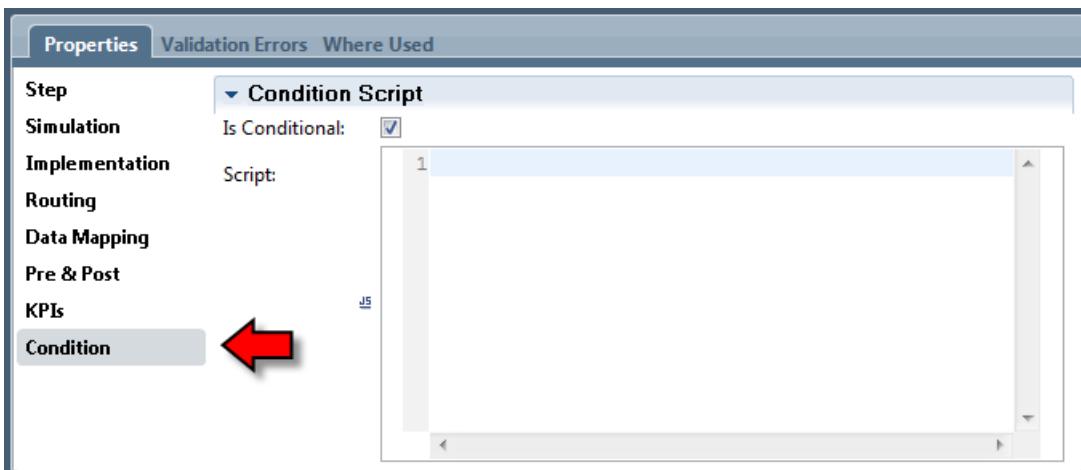
Note that the variables defined in the scope of the BPD do not have to have the same names as the actual parameters of the service implementation. However their data types should be compatible with each other.



Conditional Activities

When an Activity is reached in the BPD, it may not always be applicable to execute this activity but instead it can be skipped over. An attribute of the Activity in Process Designer can be set to achieve this effect.

In the Properties of the Activity a tab called Condition can be selected.



In the details associated with this tab, a check box called `Is Conditional` can be checked to flag the Activity as conditional. Activities that are flagged as conditional have an additional visual decoration added to their representation to indicate them as such. The decoration is a small diamond at bottom/middle of the Activity icon.



A JavaScript fragment can be coded in the `Script` box in the Condition section. If this JavaScript returns a value of `true`, the activity is performed. If it returns `false`, then the activity is skipped.

The Conditional Activities feature has one further overload associate with it and it takes a little bit of patience to get one's mind around it. If an Activity is flagged as conditional and has `no` associated JavaScript then should it be executed or not? The answer to this is dependent on a system variable called `tw.system.process.selectedConditionalActivities`. This variable is a list of `Strings`. If the Activity is to be executed, then its ID value will be found in the list. If the Activity is **not** to be executed, then its ID value will **not** be found in the list.

Initially, this list is empty and hence none of the conditional activities that have no JavaScript associated with them will be executed. This variable can be assigned inside the process to add (or remove) activities to be executed. To aid in the population of this variable, another system variable can be used called `tw.system.process.conditionalActivities`. This variable is a list of **all** the possible activities in the current BPD that are flagged as conditional. The entries in this list are of type `ConditionalActivity`. A `ConditionalActivity` instance contains both the name (as seen in the BPD) and id (GUID for the activity as needed in the `tw.system.process.selectedConditionalActivities`) variable. Using this list, algorithms can be used to select which activities to enable for execution and which to skip.

A common pattern of usage of this feature would be to present selection to the user to determine which activities should be executed and which not. IBPM provides a pre-built Human Service that performs exactly this function. That Human Service is called lsw Conditional Activity Selection Coach.

Note: Although this is a fully supported feature of the product and can be used in any production environment, experience has shown that very few uses of this capability have been seen. This is mentioned not to dissuade you from using it but rather to alert you to the notion that it is likely not critical that you spend too much time learning this function.

See also:

- lsw Conditional Activity Selection Coach

Start Event

The Start Event is a starting node that can be used to indicate the starting point of an instance of a process. When added to the canvas, it looks as follows:



By default, when a new BPD diagram is created, a Start Event node is automatically added to the diagram. The Start Event node is used to model the start of a process. From an implementation perspective, the Start Event node is used when one wishes to:

- Start an instance of a process through the Process Portal
- Define a BPD as a linked process. Every called linked process must have a Start Event.

In either of these circumstances, execution of the process will begin from this node. A BPD can only ever have one Start Event contained within it. Although the PD allows us to mechanically add more than one start event, the run-time detects this and flags it as an error.

A Start Event can **only** have outgoing Sequence Lines from it. It is not permissible to have any incoming links to it.

It is permissible for a process to have no Start Events but instead it must have one or more Start Message Events. These allow the process to be started from an external event such as a UCA.

If a process has no Start Events and is attempted to be started, an exception is thrown which shows a message similar to:

```
[9/2/11 11:16:52:717 CDT] 00000089 wle E CWLLG2229E: An exception occurred in an EJB call. Error: Cannot start BPD "BPD3". No start event of type None is found
```

Author: This message frustrates me ... "No start event of type None is found". Personally, I would have preferred to see "A start event of type 'None' was not found within the BPD named 'BPD3'".

End Event

The End Event is used to model the end of a process. When added to the canvas, it looks as follows:



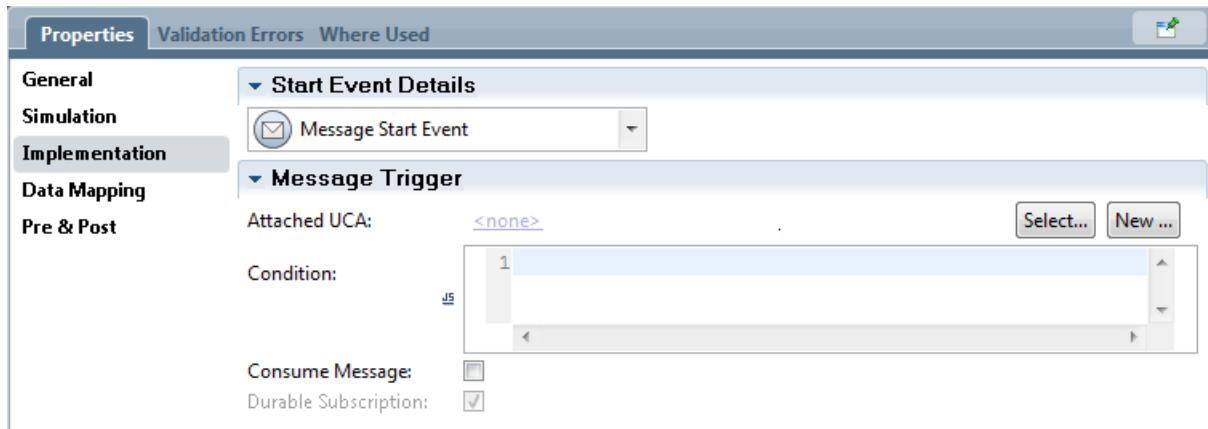
An End Event is added to the default BPD diagram when it is created. There can be multiple End Event nodes in a diagram. The process instance will come to an end only when **all** the concurrently executing branches reach their End Events. Explicitly, the process will not end when any one single branch reaches an End Event.

Message Start Event

The Message Start Event is a starting node that can be used to initiate an instance of a process due to the arrival of an event. Once added to the BPD canvas, it looks as follows:



An instance of the process can be started based on this element when an external event occurs. The association between the process and the mysterious external event happens as a result of an undercover agent (UCA). When the Message Start Event element is added, the properties of that element allow us to define which UCA should be used:

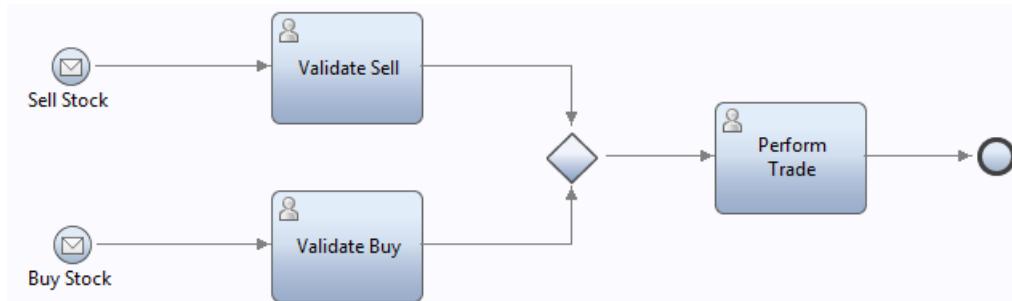


When the UCA fires the data from the UCA is passed as the starting data to the new instance of the Business Process that is to start.

The **Consume Message** check box needs some discussion. Within a BPD instance, a Message Start Event step can consume the message. This means that if there are other Message Start Event steps in the same process that are also listening for this same event, only one of them will be started. If the **Consume Message** check box is not checked, then all the Message Start Event steps for the same event will be started.

The **Condition** area is a JavaScript boolean valued expression which is a guard on whether an arriving event will be utilized by this process. The data passed with the event is available in the JavaScript name-space called `tw.message`.

A process can include multiple Message Start Events signifying multiple entry points into the same process. This is illustrated in the following diagram:



The BPD started when a UCA is fired is always the default process app assuming that multiple process app snapshots exist for the same solution. If a specific snapshot is to be used as the one in which the BPD is to be started, consider using the `startBPDBByName` JavaScript function.

See also:

- Message Intermediate Event
- Modeling Event Sub-processes
- Undercover Agents (UCAs)
- Starting a UCA from REST
- Scenario - Asynchronously starting one process from another

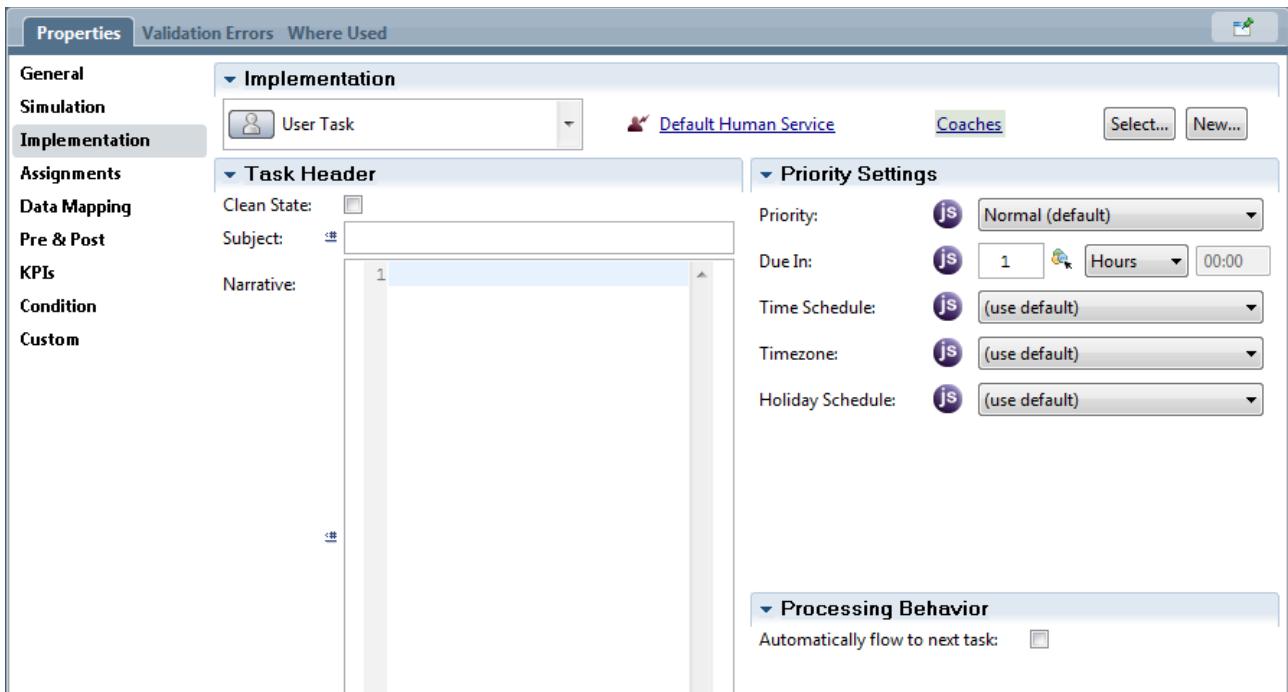
User Task

The User Task is a step in the BPD that is designed to be performed by a person. The implementation of a User Task is always a Human Service. A User Task is indicated by a human icon in the top left of the activity box:



When a User Task is reached within the process, the process pauses its execution until the task created by this step is completed. This could take some time and as such the process state is always written to database and the resources being used by the process released. There can be far more process instances in the system waiting on User Tasks to complete than could be accommodated concurrently in memory. How a user works with these tasks is a separate story told in the Process Portal section.

The User Task has a number of associated properties.



The Subject property of the activity allows us to enter a single line of text to act as a summary of the task. This becomes the title of the Human Task that is created.

The Narrative section is a description of the task. This is especially useful if the activity is a human service as this shows up in the Process Portal. If HTML is added to the narrative, that HTML is displayed in the Process Portal and is interpreted.

The `Priority` attribute associates a priority value with the task. This can be used by a task list interface to sort or otherwise visualize a list of tasks such that those with higher priority are shown first. The value for `Priority` is one of:

- Lowest
- Low
- Normal
- High
- Highest

The system does not act upon these values directly. They are merely indicators to user interfaces such as the IBM BPM Process Portal.

The `Due_In` property defines a due date for when the task is expected to be completed. This due date becomes associated with the task such that user interfaces can guide the user on which tasks to work upon first. It is presumed that those with the same priority but sooner due dates would most likely be the next ones to be handled.

The due date is calculated as the addition of a time interval to the time when the activity was started. Time intervals can be supplied as either hours, minutes or days or from a variable or JavaScript.

The `Clean_State` property is documented in the InfoCenter (8.5) as the following:

Select `Clean State` if you want to clear the runtime execution state of an activity after it is complete. By default, this option is disabled. Enable this option only when you do not want to store execution data (such as variable values) for viewing after the process finished execution.

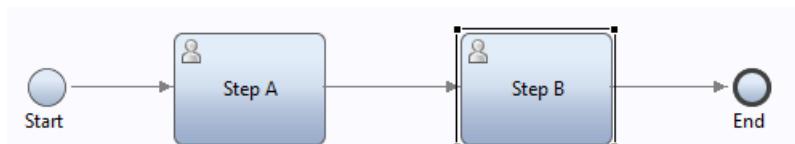
It isn't quite clear when one would use this or what exactly it means.

See also:

- Human Service

Automatic Flow

Consider the following BPD diagram:

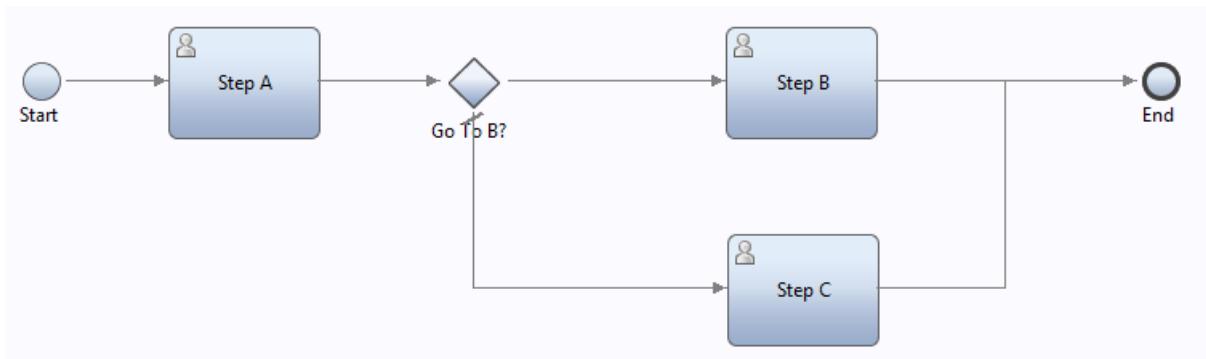


It shows two tasks to be executed one after the other. If we assume that the second task has the assignment set to "Last User in Lane" then what we are requesting is that the user who completed Step A will be the user who works upon Step B. We may now wish the process to automatically proceed to show the user (who completed Step A) the screen for Step B. Explicitly, we don't see a need to have the user go back to the task list and manually open the task. We can achieve this by the use of the "Automatically flow to next task" check box set in Step A.

Screenshot of the 'Properties' dialog box for a User Task implementation. The 'Implementation' tab is selected. Under 'Task Header', the 'Subject' field contains '1'. In the 'Priority Settings' section, the 'Priority' dropdown is set to 'Normal (default)'. Below it, 'Due In' is set to '1 Hours 00:00'. In the 'Processing Behavior' section, there is a checkbox labeled 'Automatically flow to next task:' which is checked, indicated by a red arrow pointing to it.

If checked, when Step A is completed, we now show the next task for that user. There are some constraints to this, specifically, the next task for the user in the same process must be available within a finite period of time. The default is three seconds. This is configured by the "autoflow-timeout" parameter set within the 99Local.xml configuration file. Experimentation has shown that almost any activity between one Human Activity and the next will break the association. It appears that only simple decisions will allow auto flow to work.

An interesting use case for the auto flow capability is to place logic between the tasks such as is shown in the next image:



Here we assume that something entered in Step A is used as a determinant as to whether Step B or Step C should be shown next.

Assigning Activities – Mapping staff to work

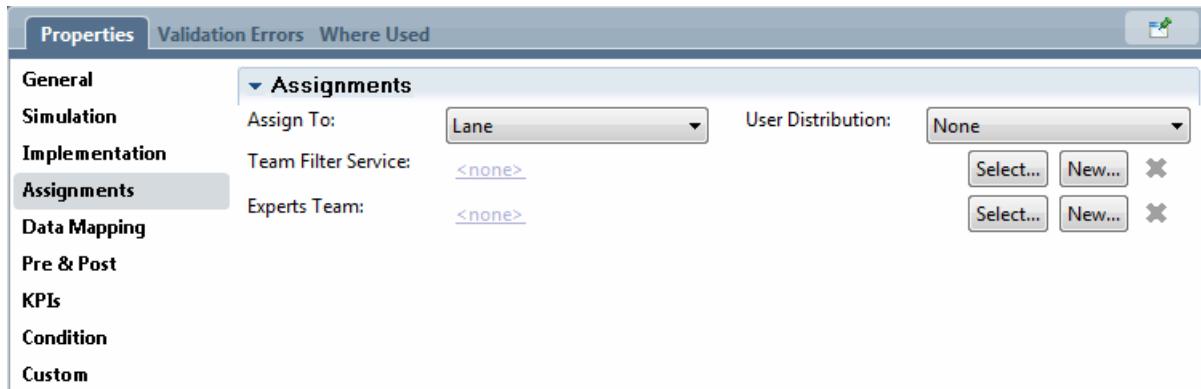
For human activities, a basic question comes to mind ... which users should be eligible to perform the work? Not all employees in an enterprise have the same roles and responsibilities. For example, if I request the purchase of a new laptop within IBM, it should be a member of the procurement department that should handle that request and not, for example, someone in the advertising department.

Assigning work to the correct people not only results in work being responded to in a timely fashion (if at all) but also has important implications with respect to security. For example, if we are bringing a new employee into our company, the request for salary information may be desired to be processed by someone in human resources and the details of that request should not be visible to others (for example staff in the new employee's same team).

Another consideration is that of balancing workload. If a department has three staff members and two are idle while the third has more work requests than he is able to handle, this is also not a good situation.

To provide answers for these considerations, task assignment can be performed through the Assignments tab in the Process Designer section while editing a human step.

By default, an activity is assigned to the set of users defined in the Team of the lane in which the activity is placed. The selection of assignment for an activity may be modified by changing the Assign To definition of the activity as found in the Assignments tab in the properties view:



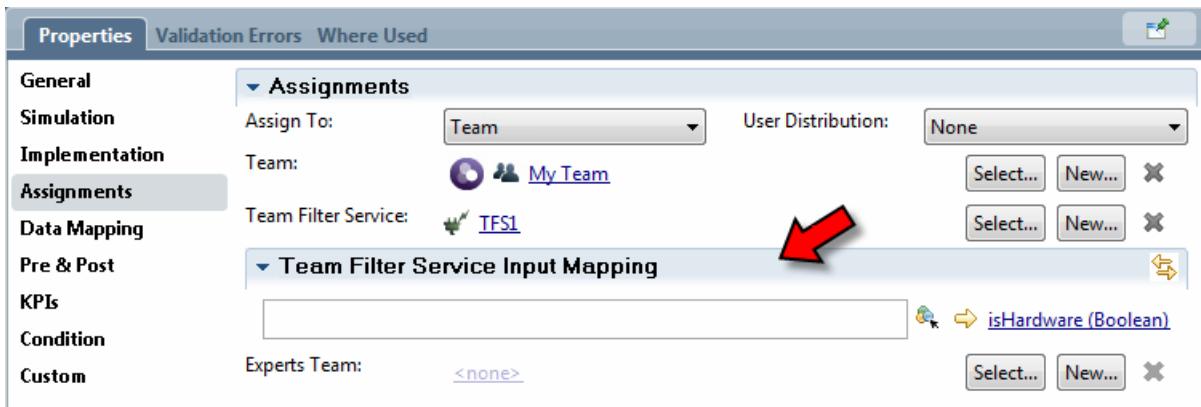
The Assign To property is a pull-down list that defines the selection criteria used to select which people are eligible for the activity. It may have one of the following options:

- Lane – This is the default selection. In this mode, the activity is assigned to the Team associated with the lane in which the activity is placed. The activity can then be owned by one of the members of that Team.
- Team – An explicitly selected Team will be selected to own this task. When this option is selected, a new entry for the Team to be chosen will be shown. A team can then be selected from the list of available defined Teams. An alternative to explicitly naming the team is to reference a String variable that will hold the name of the team.
- Last User in Lane (Deprecated) – Since this option has been deprecated in the current release of the product, no further discussion on it will be made.
- Routing Policy (Deprecated) – Since this option has been deprecated in the current release of the product, no further discussion on it will be made.
- List of Users (Deprecated) – Since this option has been deprecated in the current release of the product, no further discussion on it will be made.
- Custom (Deprecated) – Since this option has been deprecated in the current release of the product, no further discussion on it will be made.

Both the Lane and Team options also have the concept of a "Team Filter Service". This is an optional service which takes as input a Team data structure and returns a new Team data structure. The input is the selected team members that would otherwise be those assigned to the work. The output is presumed to be a subset of those team members dynamically selected to perform the work. This effectively allows us to filter the team members. If no team filter service is supplied, the original selection of team members will be used.

It is anticipated that the filtering of team members may need additional parameters in order to make its decision. For example, the staff members necessary to perform a procurement may be chosen from the procurement department dependent on whether it is hardware or software that is being

ordered. The team filter service can be supplied additional parameters in its definition. If additional parameters are defined, these will be available to be mapped in the Assignments area:



When looking in detail at a Team Filter service, we find that it has an input parameter called "originalTeam" of type Team and it also has an output parameter called "filteredTeam" also of type Team. It is expected that the service will populate the "filteredTeam" structure with the team members to be included in the assignment.

Take care when using the `originalTeam` as input. The "members" property is a list of Strings describing the members of the team. We don't know if a member is a simple user or a group of users. As such, we may need to code defensively if we wish to iterate over each user to see if they meet a specific criteria. Here is an algorithm that will give us all users in a given team:

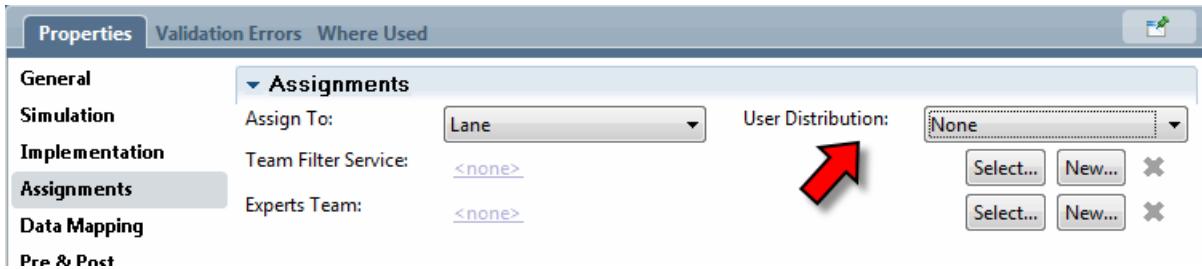
```
// Create the resulting empty list
var users = new Array();

// Iterate over each member of the team. A member may be a user or it may be a
Role (Group)
for (var i=0; i<tw.local.team.members.listLength; i++) {
    var currentMember = tw.local.team.members[i];

    // Try and get the member as a user. If we find one, add it to the user list
    var currentUser = tw.system.org.findUserByName(currentMember);
    if (currentUser != null) {
        users.push(currentUser);
    } else {
        // Try and get the member as a Role. If we find a match, add all the users in
        the role to the user list
        var currentGroup = tw.system.org.findRoleByName(currentMember);
        if (currentGroup == null) {
            log.info("Problem: Unable to find a user or group called " +
currentMember);
        } else {
            var allUsers = currentGroup.allUsers;
            for (var j=0; j<allUsers.listLength; j++) {
                users.push(allUsers[j].name);
            }
        } // We have a good group
    } // Process as a group
}
// Return the list of users we found.
tw.local.users = users;
// End of script
```

A second pull-down on the same preferences area available to all the previous types of assignment

choices is called "User Distribution". This option determines which users from within a set of users should be immediately assigned the task.



The options are:

- None – This is the default. IBPM assigns the task to all members of the Team.
- Last User – In this mode, the user selected is the last user that had a task in the same swim lane.
- Load Balance – In this mode, the task is assigned to the member of the Team who has the fewest number of open tasks.
- Round Robin – In this mode, the task is assigned to a member of the Team in a serial fashion. For example, if the Team has three members, the first time a task is assigned, it will be assigned to the first member, the next task to the second and the third task to the third member. When the fourth task is to be assigned, it will be assigned to the first member again.

See also:

- Security
- Teams
- Vimeo - [IBM BPM 8.5 Teams](#)
- DeveloperWorks - [Common business process modeling situations in WebSphere Lombardi Edition V7.2, Part 1 - Role-based and dynamic routing of activities](#) - 2011-06-29

Re-Assinging Tasks and changing group members

Imagine a BPD instance which includes an activity associated with a team. If the team is associated with an external system group then what will happen if the system group's membership changes after the creation of a task but before it is claimed? The answer appears to be that any tasks already created will honor the new group change.

Unfortunately, there does not appear to be a way to change the makeup of a team after the task is created. So if a task is created with a certain set of potential owners those potential owners are fixed to that task (unless they come from a system group).

Assignment Examples

Here are some examples of some of the more interesting assignment uses.

The Four Eyes Example

There are some processes where two staff members have to review or approve some activity. As an example, let us think about hardware procurement at IBM. If I, as an IBM employee, wish a new laptop, I can submit my request to my procurement department. Their process says that the request must be reviewed by two procurement specialists. We can imagine this being implemented in IBPM. A Human Service would be executed that assigns the request to anyone in the Procurement

group. If the request is approved we now create a second Human Service and assign that also to the Procurement group. Unfortunately, there is a problem in that simple solution. If the first task is assigned to the Procurement group and "Bob" works on it, the second task will also be assigned to the Procurement group and, unfortunately, there is nothing to prevent "Bob" from working upon and claiming that second task as well. The tasks "have" been handled by "someone" in the Procurement group ... it just was the same person each time and that is not the intent of our story.

Fortunately, there is a relatively easy solution.

We build a process that looks like this.



The assignment decisions for both tasks use the "List of Users" feature for owner selection. After the first task has completed, we determine the owner of that task and remove them from the list of potential owners for the second task. A suitable fragment of JavaScript to achieve this may look like:

```

for (var i=0; i<tw.local.users.listLength; i++) {
    if (tw.local.users[i].id == tw.local.userName) {
        tw.local.users.removeIndex(i);
        log.info("Found and removing");
        break;
    }
}
log.info("Final list: " + tw.local.users.toXMLString());
    
```

The key to understanding this fragment is to realize that `tw.local.users` is an array of the users that are in a particular group and `tw.local.userName` is the name of the user that was assigned to the first task.

Assigning a task to a named user

Consider the notion that within a process we know the identity of a specific user to whom we wish to assign a task. How can we achieve that within IBM BPM?

We create a new Team Filter service called "Single User" that has the following signature:

Its implementation is a single Server Script which contains the following logic:

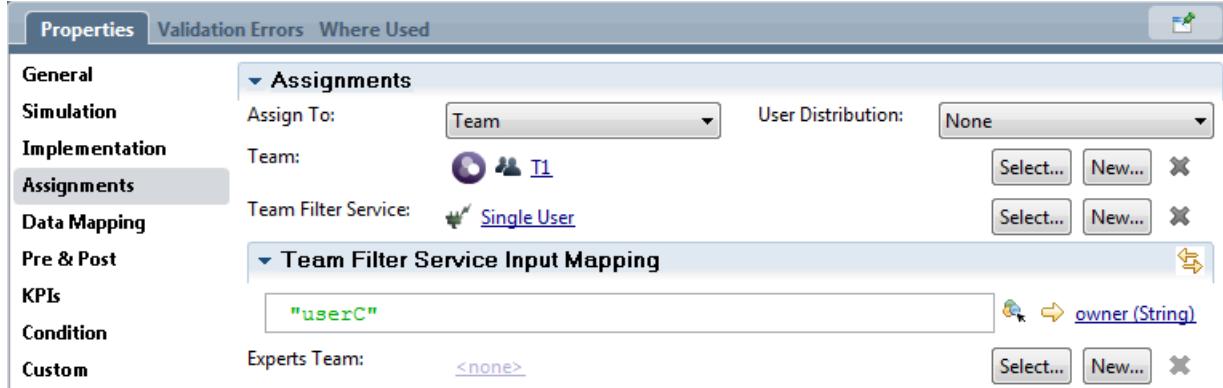
```

tw.local.filteredTeam = new tw.object.Team();
tw.local.filteredTeam.name = tw.local.originalTeam.name;
tw.local.filteredTeam.managerTeam = tw.local.originalTeam.managerTeam;
    
```

```
tw.local.filteredTeam.members = [ tw.local.owner ];
```

What this code does is build the resulting Team object from the original Team object but comprises the new team's membership solely from the identity of the passed in user.

In a User Activity's assignment specification, we can now name the service and the user we wish to assign:



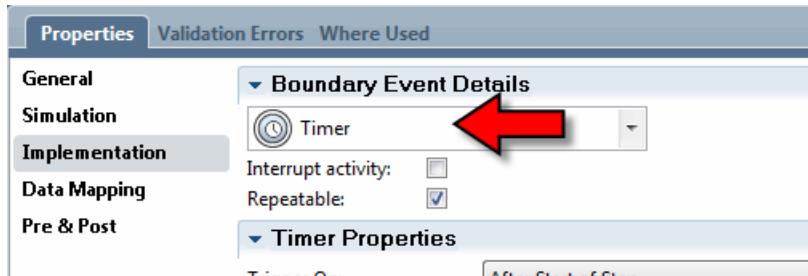
The desired owner can be obtained from a string or a variable.

Timer Events

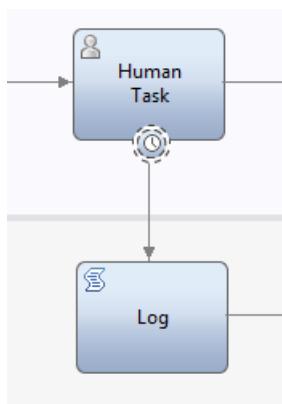
A timer can be added as an attribute on an existing task or can be an element in its own right. The icon for this can be seen in the palette and looks as follows:



Once added, its implementation type should be changed to Timer:



It can be dragged and dropped on an existing activity. This is called "attaching" a timer event. A link from this timer can then be associated with a different activity. If the timer fires, the linked activity will be called. In the following diagram, an activity called Human Task has had a timer added to it and that is wired to the activity called Log.



A timer has an associated **Trigger On** attribute. This is an algorithm used to select when the timer will start ticking. The choices available are:

- After Start of Step – A time after the step is reached
- Before due date – A time before the due date
- After due date – A time after the due date
- Before custom date – A time before a custom date
- After custom date – A time after a custom date

The **Before/After Difference** field allows us to specify the time interval relative to the **Trigger On** algorithm selection. The units for this field can be minutes, hours or days.

The **Tolerance Interval** is an additional time interval associated with the algorithm. If a task has **not** been claimed by a user, then when the primary interval is reached, the timer fires as expected. However, if the task has already been claimed, then the additional **Tolerance Interval** value is used to extend the deadline to complete the task before the timer fires.

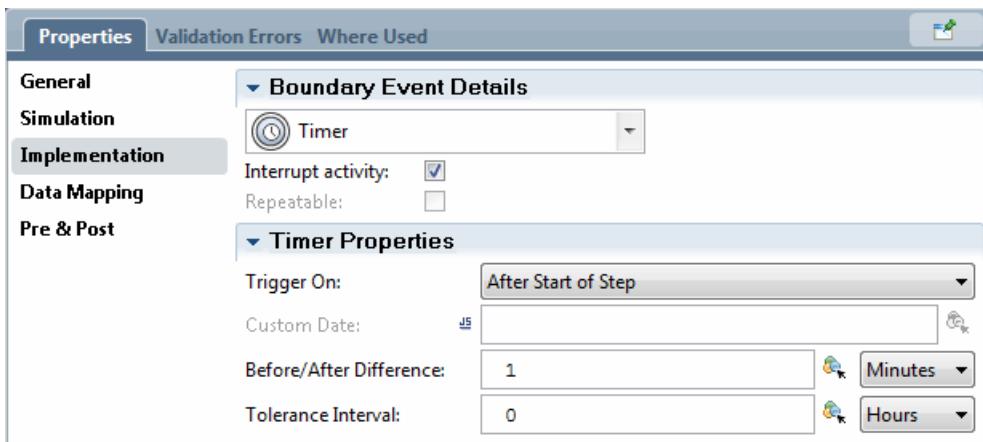
The **Interrupt Activity** option causes the activity on which the timer is attached to end when the timer expires. The icon used to represent the Timer changes depending on the setting of Interrupt Activity:

Interrupt Activity: On	
Interrupt Activity: Off	

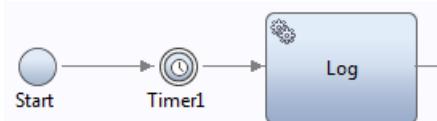
If this option is **not** selected (which of course means the attached activity continues to run), the **Repeatable** option may be selected to cause the timer to start again after it has already fired. The timer is restarted as though it had just been reached. A little care should be taken here. If we flag a timer as repeatable and the date/time on which the timer is to fire is an absolute and that date/time has passed, then the timer will immediately re-fire over and over again without much of a pause. It is unlikely that this is the desired result.

Multiple timers can be attached to a single activity.

For the fields which can be used for time values, either constants or variables may be used.



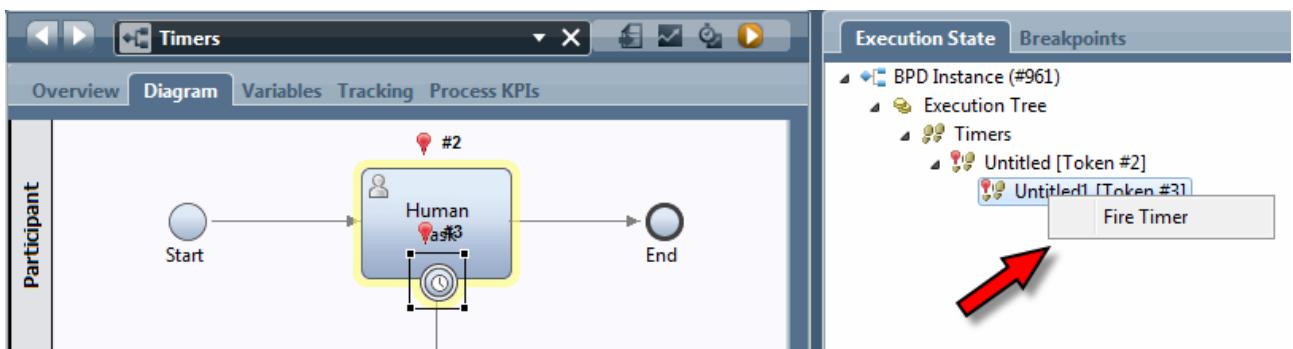
Similar to the attachment of timers to activities, timers can also be added in-line between activities.



This has the same Timer Properties as previously discussed. When this Timer is reached, processing delays until the timer expires. This allows for delaying the execution of a process for a period of time.

When a timer fires, its Post-assignment variables are in the scope of the attached activity. This means that we can copy out the Task instance id that had time out on it.

While working on the development and testing of a BPD that uses Timer Events, the Inspector view can be used to manually fire a timer even though the time interval has not yet expired. This can be useful for testing logic paths. If we right click on the token in the Execution State area, a context menu pops up that provides Fire Timer. If selected, it will fire the timer and the system will behave as though the timer had elapsed.



This is a great solution when one is able to attach a Process Designer but that is only available within a development environment. An alternative which can also be used for testing is to use the Process Admin console and switching to the Process Inspector tab. From here we can view the list of processes and make changes to them. Notice the option at the bottom right of the page. Here we find a list of active timers within the process. We can select a timer and cause it to fire immediately thereby bypassing any waits that may be outstanding.

The screenshot shows the IBM Process Admin Console interface. On the left, there's a sidebar with a search bar, filter options (Status: Active, Completed, Executing, Failed, Late, Suspended, Terminated), and a date range selector (From: 2013 May 16, To: [empty]). The main area displays a list of process instances. Each instance has a status icon, a title (e.g., CEM-T Survey Preparation), a brief description of its state, and a timestamp for the last action. On the right, there's a detailed view of the first instance, showing its status (Active), start time (2013-05-16 06:54:03 PM), and end date (2013-05-24 12:00:00 AM). It also lists actions (Suspend, Terminate) and tasks (Prepare Process, Send Initial Notifications, Log Weekly Call Attendance, Send Second Notice). Below these, a 'Timers' section is shown, with a red arrow pointing to the 'Final Notice Timer' entry.

On occasion, the time when the timer is to fire is not a simple fixed value but must instead be calculated. For example, we might want the timer to fire after four *business* hours instead of four wall-clock hours. To achieve this, we need to explicitly calculate the date/time when the timer is to fire and supply that as a custom time value.

See also:

- Error: Reference source not found
- Data Type – TWTimerInstance

Tracking Intermediate Event

A tracking intermediate event is used to generate a data point used in performance reporting on process instances. When added to the diagram it looks as follows:



When encountered, a record that this point was reached in this instance of the process is made. In the implementation section of the element, a reference is made to a Tracking Group:

Properties Validation Errors Where Used

Implementation

Tracking Properties

Tracking Group: **OrdersTG**

Performance Data Warehouse ID: b5f7a14f8ba531

<input type="checkbox"/> dateShipped (datetime)	
<input type="checkbox"/> price (number)	
<input type="checkbox"/> dateOrdered (datetime)	
<input type="checkbox"/> widgetType (string)	
<input type="checkbox"/> orderID (string)	
<input type="checkbox"/> salesRep (string)	
<input type="checkbox"/> quantity (number)	

We can define which of the tracking group properties will be populated and with what values. Checking the box next to the tracking group property causes that property to be included in the output record. We can also define which variables or values within the process should be mapped to the tracking group property.

See also:

- Tracking Groups Overview
- Monitor Tracking point data

Ad-hoc Start Event

The Ad-hoc Start Event is a mechanism that allows us to start a chain of activities in an already existing process instance. When added to the process, a new entry appears in the options for an instance of this process or in the Process Portal. When the steps associated with the Ad-hoc Start Event are invoked, these steps have visibility to the data contained within the process as a whole.

When added into the BPD diagram, an Ad-hoc Start Event step looks as follows:



The name of this concept ("Ad-hoc event") is an odd one and one I find not representative of what the function actually does. What we really have here is the ability to have an optional "event" passed into the process which causes a new flow of control to execute in the same process context.

The implementation properties of this entity are shown below:

Properties Validation Errors Where Used

Implementation

Start Event Details

Ad Hoc Start Event

Event Visibility

Restrict the visibility of this event by:

Swimlane:

Milestone:

By default, the event can arrive at any time during the life span of the process. Further options are also provided:

- Swimlane – TBD
- Milestone – The Ad-hoc start may only be initiated when a process token is within the same milestone as that of the ad-hoc activity

When added into the BPD, you can drop the Ad-Hoc start event in any area. We have the option to associate an Ad-Hoc Start Event with a “milestone”. If added to a milestone, the event is only available when the process is within the milestone area. You switch this on with the check marks in the implementation area of the Ad-Hoc event. By default, the Ad-Hoc event is available in all milestones (and all swimlanes).

See also:

- Error: Reference source not found
- Starting Ad-Hoc entry points
- Securing Process Portal capabilities
- [IBM Education Assistant – Ad-hoc tasks](#)

Message Intermediate Event

The Message Intermediate Event is an Event listener that is used to listen for incoming events within a process instance that is already running. Contrast this with a Message Start Event which will initiate a new instance of a process when a message arrives. The Message Intermediate Event has two flavors.

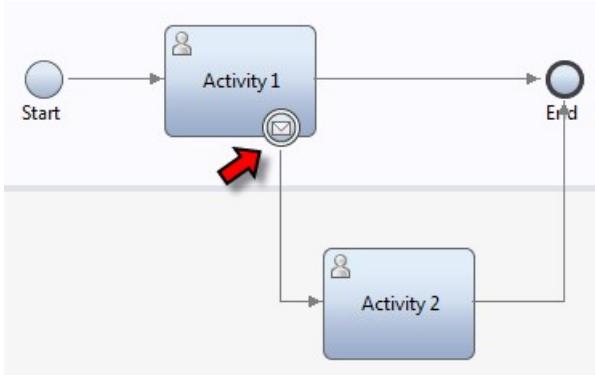
These are called:

- Stand alone listener
- Attached listener

The Stand alone listener is seen within a BPD as its own diagram element. When it is reached, it suspends further execution of the process until an event is received or the process is terminated elsewhere.



The other type of Message Intermediate Event is called the Attached listener. It is called this because it *attaches* to an activity. While the activity is running, if an event arrives, a link coming from the Message Event will be followed. An option called "interrupt activity" available on an attached listener determines whether or not the outgoing link from the activity will also be followed when the activity completes. If "interrupt activity" is checked, then if a message arrives, the activity is cancelled.



Because a Message Intermediate Event listener lives within an existing process instance, when an event occurs that event must be matched against the correct instance of the process for which the event is destined. The ability to match the event against this correct process instance is called "correlation".

The Message Intermediate Event is actually watching for the arrival of a UCA. When that UCA arrives, the event is said to have happened. This is the mapping between the logical concept of an event and the actual implementation of an event within the IBPM product.

The settings for an Attached listener are as follows:

Properties		Validation Errors	Where Used
General	Boundary Event Details		
Simulation	<input type="button" value="Message Intermediate Event"/>		
Implementation	<p>Interrupt activity: <input checked="" type="checkbox"/></p> <p>Repeatable: <input type="checkbox"/></p>		
Data Mapping			
Pre & Post	Message Trigger <p>Attached UCA: <none> <input type="button" value="Select..."/> <input type="button" value="New ..."/></p> <p>Condition: <input type="listbox" value="1"/></p> <p>Consume Message: <input type="checkbox"/></p> <p>Durable Subscription: <input checked="" type="checkbox"/></p>		

The settings for a Stand alone listener look like:

Properties		Validation Errors	Where Used
General	Intermediate Event Details		
Simulation	<input type="button" value="Message Intermediate Event"/> <input type="button" value="Receiving"/>		
Implementation	Message Trigger <p>Attached UCA: <input type="button" value="UCA1"/> <input type="button" value="Select..."/> <input type="button" value="New ..."/></p> <p>Condition: <input type="listbox" value="1"/></p> <p>Consume Message: <input type="checkbox"/></p> <p>Durable Subscription: <input checked="" type="checkbox"/></p>		
Data Mapping			
Pre & Post			

Now, what if the event that would trigger the listener occurs **before** the listener is reached? This is where the concept of the Durable Subscription comes into play. If this flag is checked, then the arrival of an event before the listener is reached is remembered and when the process

reaches the Message Intermediate Event, it is immediately notified that the message is available and the solution can continue. If this flag is not checked, then the event that arrived previously is discarded and the process will wait for a further event.

The **Consume Message** option is used to indicate that if this event listener is triggered by an event, should a subsequent event listener also be triggered by this same event? If **Consume Message** is checked then the next occurrence of the listener will not be triggered by this event occurrence.

Consider a message arriving that says "Payment has been received". This is great news and now the process can continue. But wait a moment ... we seem to have a problem. Assume that I am a car salesman and I have two separate customers waiting in my office. My finance department calls me and says that the payment has been received and the customer can leave with the car. This by itself is not sufficient. I have **two** potential recipients of keys ... for *which* customer has the payment cleared?

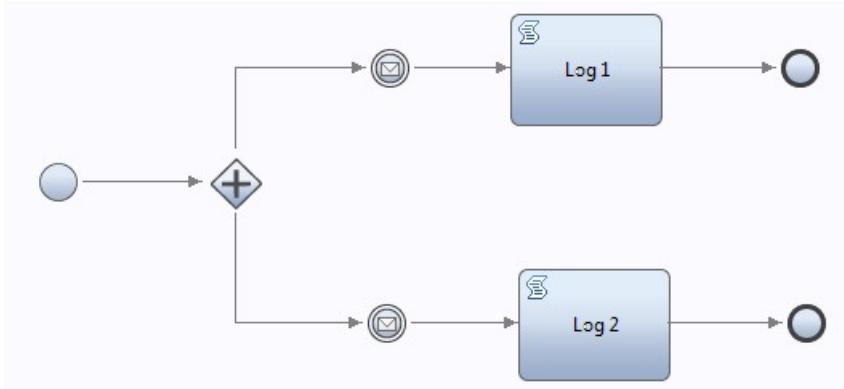
In IBPM, the challenge is that when an event is to be delivered to a process we must know to **which** instance of the process should the event be delivered. To resolve this problem, an event must carry with it sufficient information to allow IBPM to uniquely identify to which instance of a process the message should be delivered. This information is called *correlation*. We need to correlate an incoming message with the process instance to which it is to be sent.

A Message Intermediate Event is always associated with a UCA and a UCA can carry data with it. In the IBPM PD, we can specify a variable who's value must match that of an incoming event's UCA payload in order to be considered a match.

The screenshot shows the 'Properties' tab of the IBPM interface. Under the 'Data Mapping' section, the 'Correlation and Output Mapping' tab is selected. A note says: 'Select one variable to be used for correlation. Specify data mappings from the UCA output data to the BPD variables.' Two mappings are listed: 'correl1 (String)' mapped to 'tw.local.correl1' and 'data1 (String)' mapped to 'tw.local.data1'. Both mappings have small edit icons to their right.

Note that it is **vital** that each of the UCA Output Correlation fields name a local variable even if they are not to be used for the correlation mapping. Currently, the product only allows the selection of a single variable for correlation matching. If the match requires additional comparison, then a new variable must be defined which will be the concatenation of multiple values. In the UCA associated General Service, the incoming data will also need to be concatenated together to build the matching data.

With this background information in place, we can now look at some potential usage scenarios. The first one we will choose is the idea of the Multi Wake-up. In this scenario, we will have a single process with two parallel Intermediate Message Events **both** waiting on the same event:



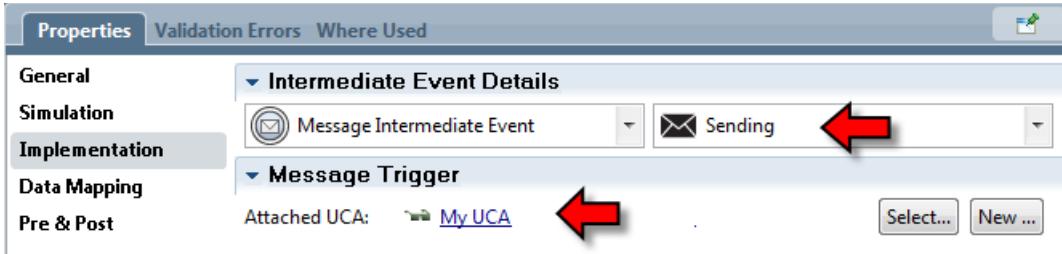
What we desire is for both of these to block and wait for a new event to arrive. We define the Intermediate Message Events to:

- Be associated with the same UCA
- Use the same correlation value
- To **not** consume the message. If we consumed the message, one would wake and other other wouldn't.
- To **not** have a durable subscription.

In addition to blocking waiting for a message to arrive, the Message Intermediate Event can also be used to invoke a UCA which is the logical equivalent of generating a new message. A Message Intermediate Event can be visually seen as sending a message rather than waiting for a message because it has a different representation in the BPD diagram. Specifically, it has a black envelope icon:



In the Implementation tab of the activity there is a selection option in which we can define it to be either receiving or sending. Changing it to Sending results in the invoke of a UCA when reached. The name of the UCA is also supplied on this section. Once a UCA has been chosen, any data passed as input into a UCA of that type can be defined in the Data Mapping section.



See also

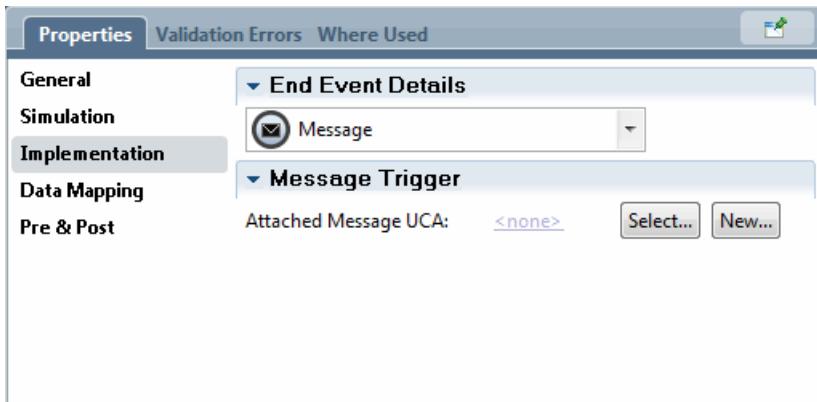
- Message Start Event
- Undercover Agents (UCAs)
- Starting a UCA from REST

Message End Event

The Message End Event ends the current path of the process and fires an event. When added onto the canvas, the Message End Event looks as follows:



Its implementation properties are:



When reached, it causes the Event associated with the configured UCA to be fired and then ends this path in the process. This activity is useful for modeling a final outcome of a process flow.

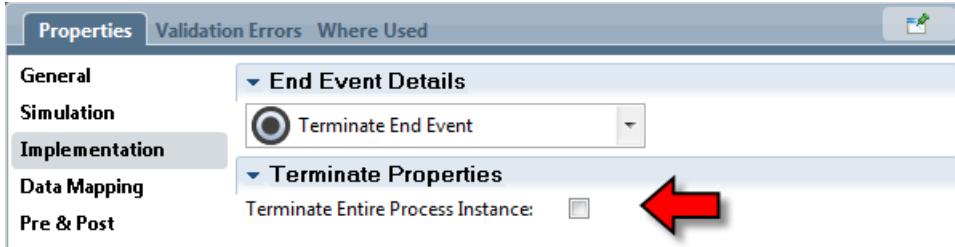
Terminate Event

The terminate event activity looks as follows when added to the canvas. It is added from the End Event palette entry and then changed in its implementation settings.



This activity will cause the immediate termination of a process instance. If there are multiple branches of control within the process, the first branch that reaches a terminate event will cause the immediate termination of all branches. Putting it another way, it will close all outstanding tasks and cancel any timers.

A check box on the implementation section of this event allows us to flag it as terminating the entire process instance or just the process part in which the terminate end event is found. This comes into play when we think about linked or sub-processes.

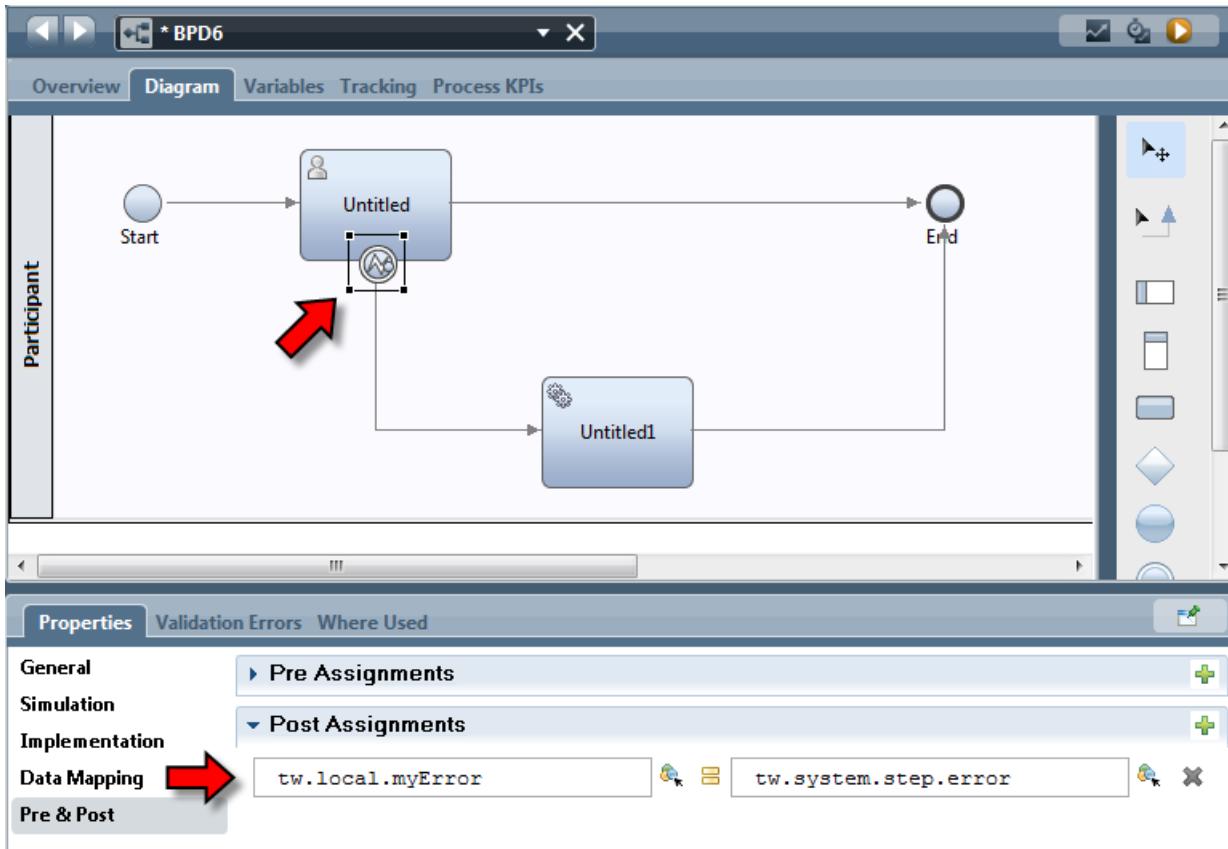


Error Intermediate Event

An Error Intermediate Event may be attached to an activity. Control is passed to this event handler when an exception is caught within the implementation of that activity. In a typical diagram it may look as follows:



The exception occurrence details are available within the variable called `tw.system.step.error`. Note that this variable only has a value within the activity that populates it. This means that downstream from the Exception handler, it will have no value. The details of the exception should be copied out of the Intermediate Exception Event to a local variable of type `XMLElement` in a Post-Assignment expression if they are to be subsequently used.

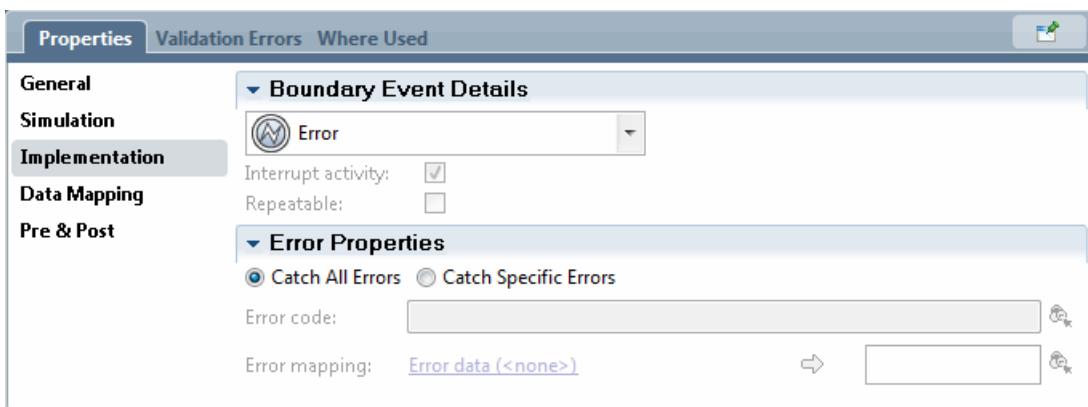


The exception details can then be retrieved from the XML element variable.

In general, the format of the XML returned is as follows:

```
<error type="..." description="...">
  <cause type="..." description="..." />
  <localizedMessage type="..." description="...">text</localizedMessage>
  <message type="..." description="...">text</message>
  <messageKey type="..." description="..." />
  <stackTrace type="..." description="...">
    <element type="..." description="...">text</element>
    ...
  </stackTrace>
  <toString>text</toString>
  <stackTrace>text</stackTrace>
</error>
```

The Implementation section of this activity has additional properties.



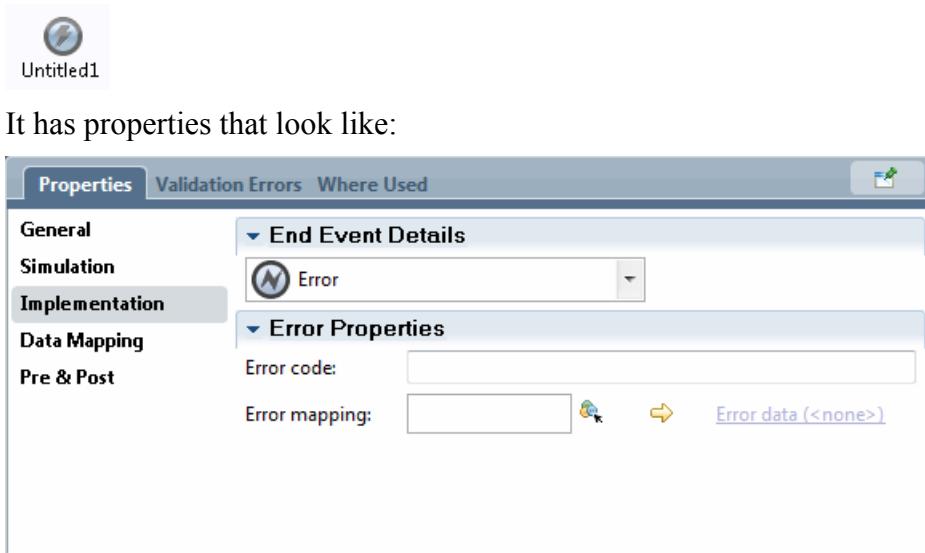
See also:

- DeveloperWorks - [WebSphere Lombardi exception handling and logging](#) - 2011-05-25

Error End Event

If a process detects that an error condition has occurred, it can terminate the remaining and current steps of the process and throw an Exception. This exception can be caught by higher level exception handlers or passed on to the user.

To add this item, drag an End Event activity from the palette and change the implementation type. The resulting icon in the canvas looks as follows:



It has properties that look like:

Here an error code and error data can be supplied that are passed onwards with the exception details.

Gateways, Conditionals and Joins

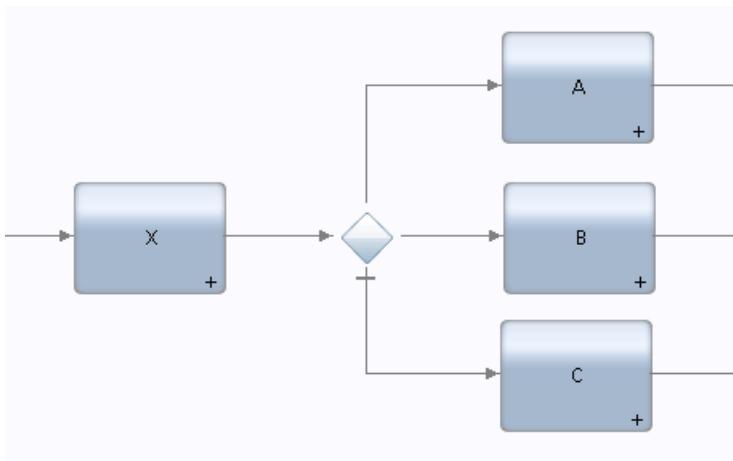
Gateways are decision points in the BPD. When reached, an expression is evaluated and based on the outcome of that expression, control flows in different paths. In IBPM, the JavaScript language is used to evaluate the decision's outcome.

The simplest gateway is called the Exclusive Gateway. The outcome of the expression is a boolean (true or false) outcome. The output of the decision is always that a single path is taken. On the

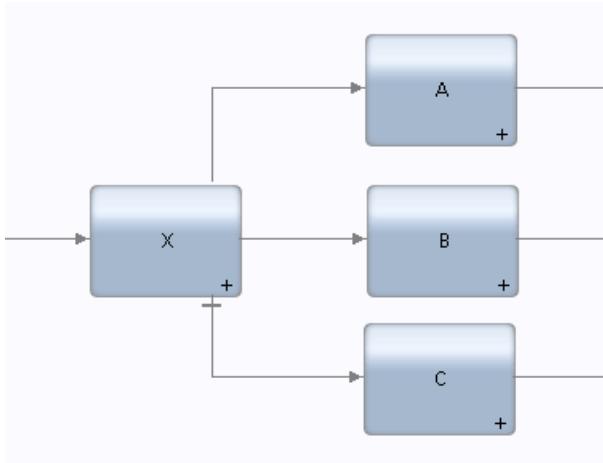
diagram, it looks as follows:



In an actual BPD usage situation, it may look like:



The inclusion of the decision gateway in this fashion is called an explicit decision gateway because the decision icon is explicitly included in the diagram. An alternative construct called the implicit decision gateway looks as follows:

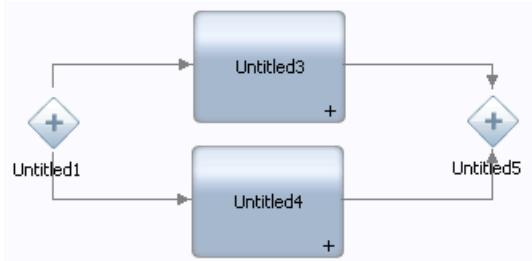


Here, the decision links come straight from the preceding activity. Although semantically equivalent and hence the visual style is all that separates them, the explicit decision gateway is much more commonly seen than the implicit style.

The second type of gateway is the Parallel Gateway. It does **not** have an expression associated with it. When reached, **all** the paths associated are followed in parallel. Each followed path is termed a branch.



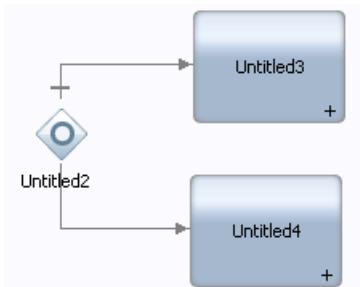
To join the paths again, the same primitive can be used as a join



The third type of gateway is called the Inclusive Gateway. Like the simple decision, there are expressions involved but in this case, control can flow from this primitive in one **or** more directions assuming multiple conditions evaluate to true.



For an Inclusive Gateway and Simple Decision, one path is defined as the *default* and is taken if none of the expressions associated with the other paths evaluates to true. The default path is visually marked as the default by having a “bar” through it.

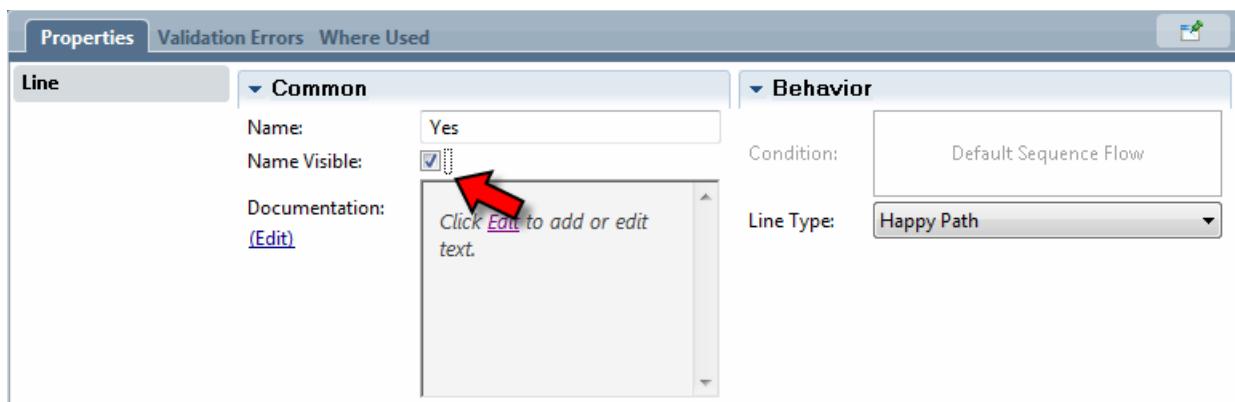


When a link is made from a decision type gateway, it is a good practice to give the link a name and make the name associated with the link visible.

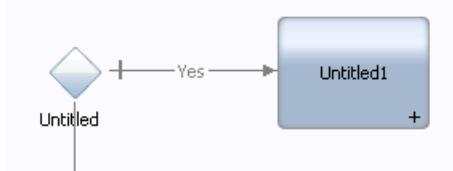
Here we see a decision with no visible label on the link connection:



By selecting the link between the decision and the next component, in the properties view, a tab called **Line** is shown. Now we can give it a name (“Yes” in this case) and check the box that the name should be visible.

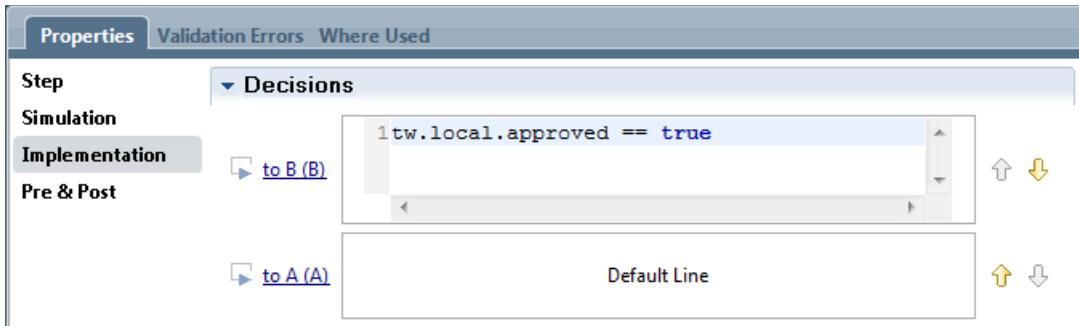


The result is that the name is now shown on the line:



This makes the overall diagram much more readable as now one can tell just by looking at a decision box in a diagram the logic associated with a decision outcome.

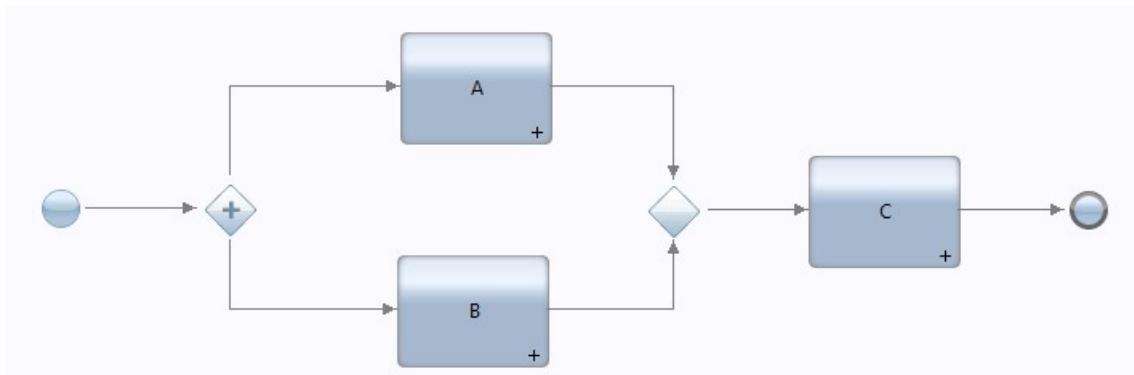
For gateways that have expressions associated with them, the expression can be entered in the Implementation section in the Properties view:



The expressions are supplied as JavaScript expressions. When expressions are defined, there is always one item that is defined as the default line. This link will be followed if none of the expressions evaluate to true. The order of the expressions is important. For an Exclusive Gateway, only one path is followed. If multiple expressions could evaluate to true, the first expression that is true is the one followed. This means that the order of evaluation has a bearing on the outcome of the logic flow. The expressions can be re-ordered up or down using the arrows to the right of the expressions. The default line is always the last in the list.

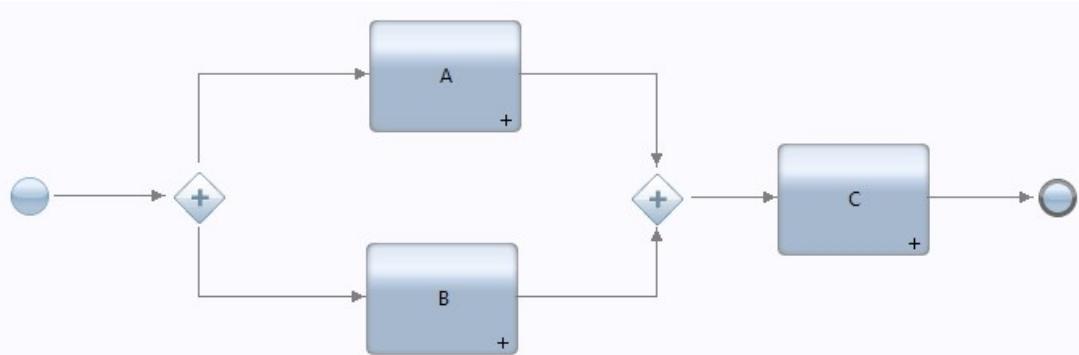
The Joins in a process diagram also need some explanation. If we look at the BPD palette we see three possible different types of joins so let us look at how they differ and how they might be used.

The first one we will look at is the exclusive join.



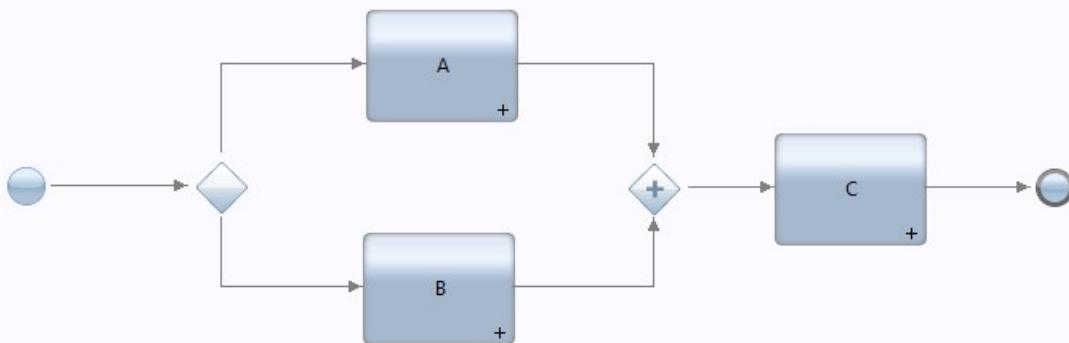
In this case, if **either** A or B complete, then C will start. There is no synchronization. If one thinks about this, there is a new question to be asked. Obviously, **one** of A or B will complete first and C will be started ... but what happens when the other of A or B then completes? The answer is that a second instance of C will start. From a token perspective, the token from A will pass to C and **also** the token from B will also pass to C.

The next join we look at looks like this. This **is** a synchronizing join. In this instance **both** A and B must complete before C is allowed to start.



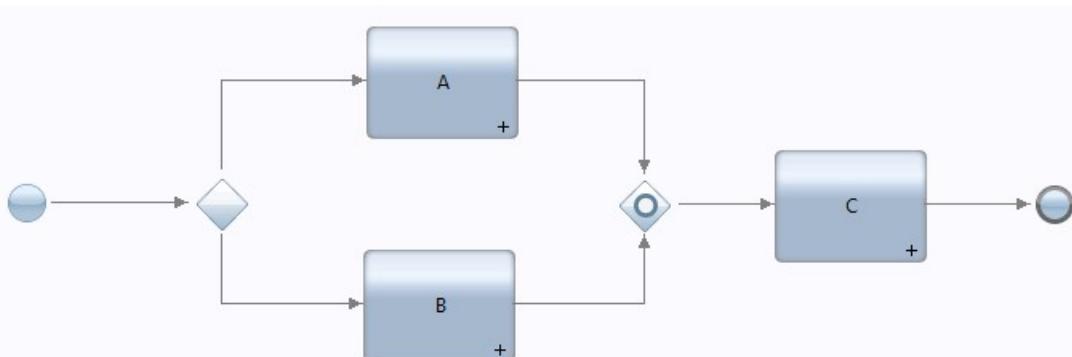
From a token perspective, the tokens from A and B are merged at the join to form a single token which is propagated onwards. With this join, **all** inputs must be satisfied before C can progress.

Still looking at the same join, give some thought to this pattern:



Notice that only **one** of A or B will execute as the condition gateway is an either/or. The join expects **all** inputs to be satisfied before continuing onwards towards C. Since only one of A or B will execute, this is an invalid diagram as the join will **never** be satisfied.

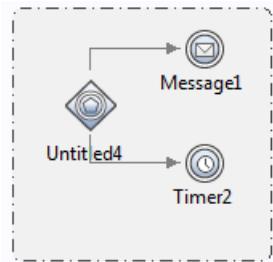
This brings us to the last type of join.



This join has somewhat "magical" properties. Again, it is a synchronizing join and will wait for all possible inputs to arrive before it continues but the magic here is my use of the word "possible" inputs. In this diagram, it will know that a token will arrive **only** from one of A or B and when one arrives, it will pass on to "C". However, if there were other diagram structures where other tokens could arrive, it will wait for those. Think about this join as being able to do work to see the bigger picture and know what possible tokens *could* arrive and for the ones that will come in the future, will wait but if there are no further tokens that will arrive, it will allow control to pass onwards.

There is one final gateway type that is called the Event Gateway. Strictly speaking, this should really be considered an intermediate message event but it is grouped in the gateway section in the

PD tooling.



The event gateway waits for a message to arrive but, if a message has not been delivered within a time period, the gateway times out and follows the path associated with the timer. This allows us to wait for a message and, if none has arrived, we can timeout.

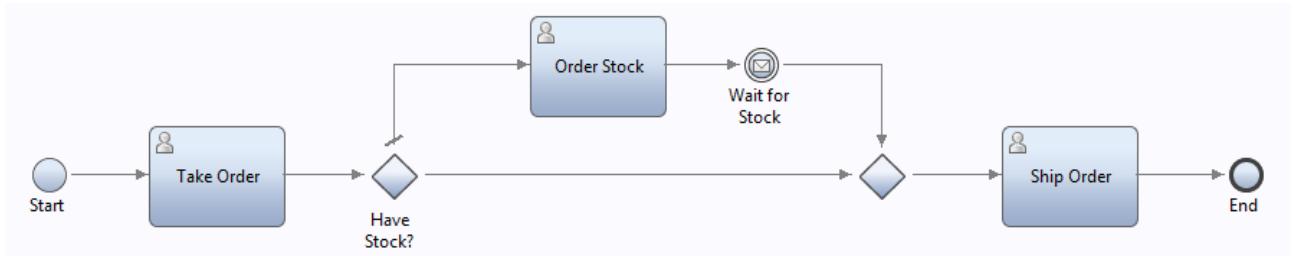
In addition to using the expression mechanism described above, one can also use the Decision selection:



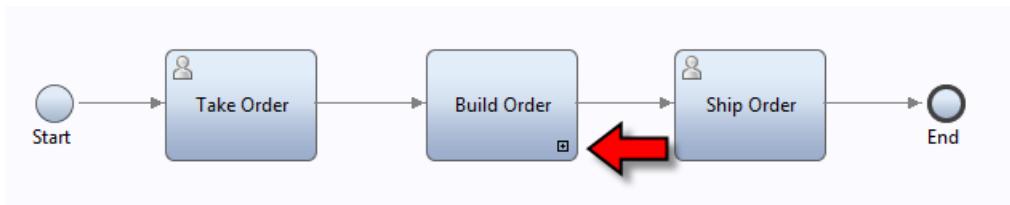
In the Decision section, a Decision Service can be mapped to be called. It can be passed any mandatory parameters needed for its execution. This decision service is invoked when the gateway is reached. The resulting output parameters from this decision are available in the variable name-space that starts "tw.decision.*". These values can then be used in the expressions to determine which path the process should follow.

Modeling sub-processes

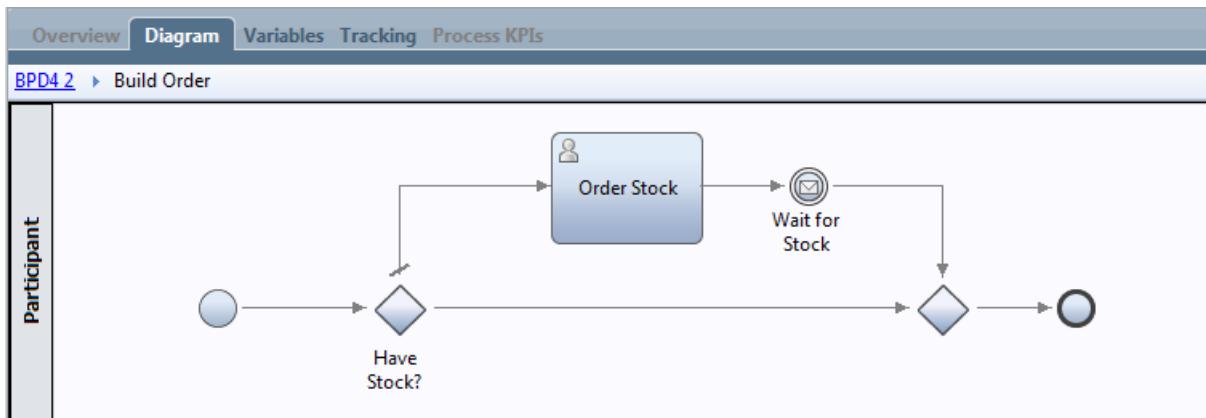
Within a BPD, we can create a sub-process. A sub-process is simply a grouping together of a set of activities into an aggregate area. For example, consider the following process diagram:



Here we see us take an order, package the order (getting more stock if we are out) and finally shipping the order. What we need to pay attention to is the center part of our diagram. The details of ordering stock (if needed) is perhaps not necessary to understanding the overall flow of our process. Instead, we might want the diagram to look as follows:



Notice the marker on the `Build Order` step. This indicates that this is a sub-process. If we drill down into this step, we find:



What we have done is nested the steps for the `Build Order` activity as a grouped/hidden sequence of steps. This sub-process shares the same variables as the parent. In addition, the names of the steps defined in the sub-process must be distinct from those of the parent. It is as though the steps were placed "in-line" in the parent process and have simply been hidden for readability. An important aspect to note is that there is **no** re-usability of these sub-process steps. Their hiding in a sub-process is for readability only.

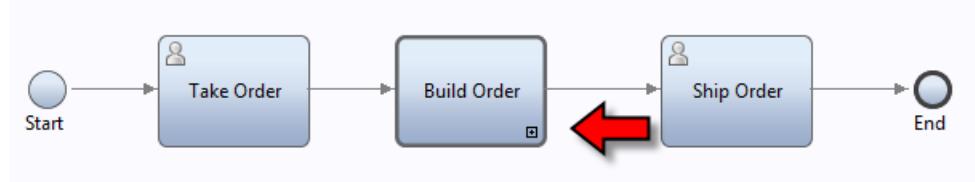
Experimentation has shown that if a Terminate Event is met within a sub-process then the sub-process is terminated and control is returned to the step following the sub-process definition.

See also:

- Terminate Event

Modeling Linked Processes

An alternative style of creating sub-processes is the notion of the Linked Process. In this story, a separate BPD is created that contains re-usable BPD activities. Input and Output variables are defined which describe the expected inputs and outputs from the linked process. In the calling process, a Linked Process activity is defined and a reference added to the target BPD that should be invoked when the parent reaches it.



Again we can see the marker that indicates that it contains additional steps. Notice the heavy border around the activity which marks it as distinct from a sub-process.

A step in the process can dynamically choose which linked-process to invoke without explicitly having to define the name of the BPD to be called. To achieve this, create a variable of type String and populate its value with the name of the linked-process to invoke. In the Advanced section of the Implementation area, select that variable as the source of the name of the process. At run-time, the variable will be consulted and a dynamic call to the process with that name will be made.

If parameters are to be passed to a dynamically called process then each process that may be potentially selected to be called must have the same set of parameters. Think of this as the dynamically invoked process having a template.

The BPD that is named as the process to be started **must** have a `Start Event` contained within it and this will be the starting point for the new sub-process.

When a parent BPD invokes a child BPD any variables passed in as parameters are passed by reference. What this means is that if a child process changes the values of these passed in variables then the changes will also occur in the parent process. Care must be taken here to watch out for unexpected side effects.

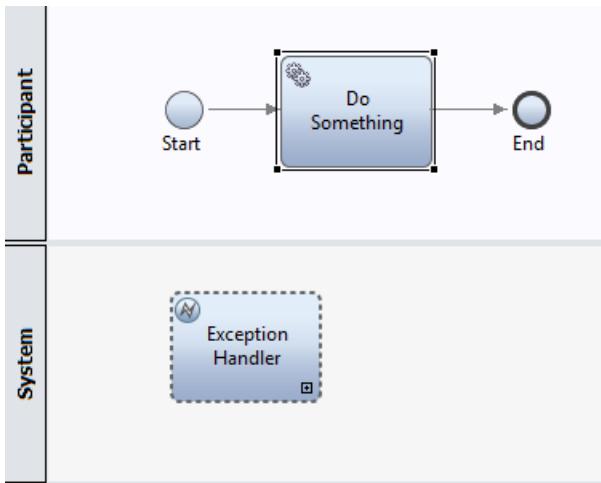
The BPD Process ID of the parent is the same Process ID used for the child.

Modeling Event Sub-processes

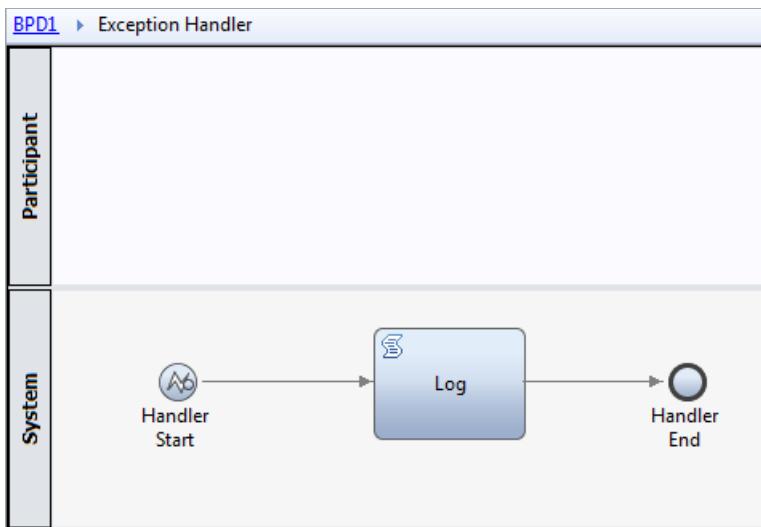
For certain types of events, we can create a "sub-process" which will be invoked if such an event is detected but is not otherwise handled elsewhere. For example, if we have a set of steps that we wish to be executed whenever an exception is thrown we don't want to have to wire this code to every activity in our process that may throw an exception. Instead, we create an Exception Event Sub-process that will be invoked when **any** uncaught exception is thrown.



For example, looking at the following BPD fragment, we see an activity called "Do Something" that presumably does something. Under normal circumstances, "Do Something" will end and that will indicate the end of the process as a whole. But, what if "Do Something" throws an exception? That is where the event sub-process that we called "Exception Handler" comes into play. It contains a set of activities that will be executed whenever an exception event is thrown.



If we double click to expand the Exception Handler, we will see that it itself contains steps:

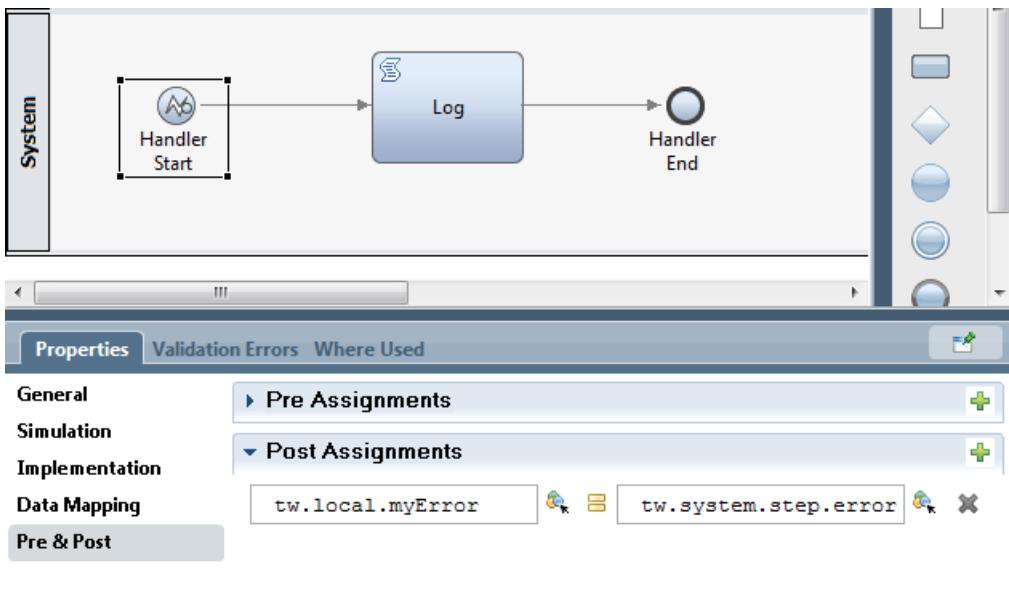


In this case, it is simply a Script fragment that logs to the console. So, if the parent activity called "Do Something" throws an exception, control will be given to the steps contained within the "Exception Handler" which will log data.

Note that although Process Designer allows us to create multiple Sub-process handlers for the same type of event, it is an error to do so. It isn't clear which of the two will be executed.

Because the Sub-process event handler doesn't have any inputs or outputs, there is no follow-on work from this step.

To gain access to the exception details in an Exception Sub-Process handler, a variable of type `XMLElement` must be created and assigned from the `tw.system.step.error` variable in the Post Assignment of the Start node in the diagram:



The Event Sub-process provides handling for event types other than exceptions. Message and Timer events can also be modeled in a similar fashion.

For Event Sub-processes which have starts defined for Message, we have the opportunity to supply a correlation id value to ensure that the correct instance of the process is woken by a corresponding event.

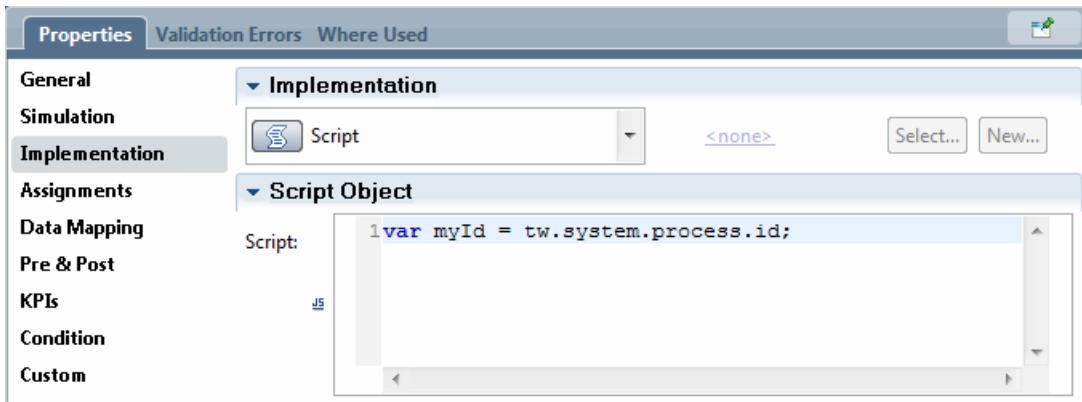
Take extra special care when defining an Event Sub-process which is triggered by the arrival of a message. The parameters called "Interrupt Parent Process?" and "Repeatable" come into play. If "Interrupt Parent Process" is checked, the arrival of the message causes the container of the Sub-process to be terminated. If the "Repeatable" option is not checked, then messages after the first one will be ignored.

See also

- Message Start Event
- DeveloperWorks - [Designing event-driven business processes in IBM Business Process Manager](#) - 2012-06-27

Script Activities

An activity can have an implementation type of Script.



Script allows the programmer to include JavaScript code in-line within the BPD process. The JavaScript can utilize the IBPM supplied JavaScript classes. Although this option is exposed to be used it is unlikely to be a good long term strategy. Code entered here is not re-usable by other activities or services. A better idea would be to build a General Service which includes a Server Script element and invoke the General Service.

External Implementation

When an activity in a BPD is reached, it is associated with and implemented by one of the implementation Service types associated with the IBPM product. These include the common General Service, Integration Service and Human Service types. IBPM provides another type of implementation that is called an "*External Implementation*". This can be subtle to understand so we will take it slowly. Also note that in previous releases, the External Implementation used to be called an External Activity.

The overall goal of the concept of the External Implementation is that some application or code **outside** of the IBPM environment is going to perform some work on behalf of the overall execution of the process. This is not an uncommon situation and IBPM provides a variety of ways in which external applications can be **called** to perform work and return their results. This includes Web Services, REST and other Integration mechanisms. The External Implementation concept though is something different. It is much closer to the concept of a Human Service than it is to an Integration Service.

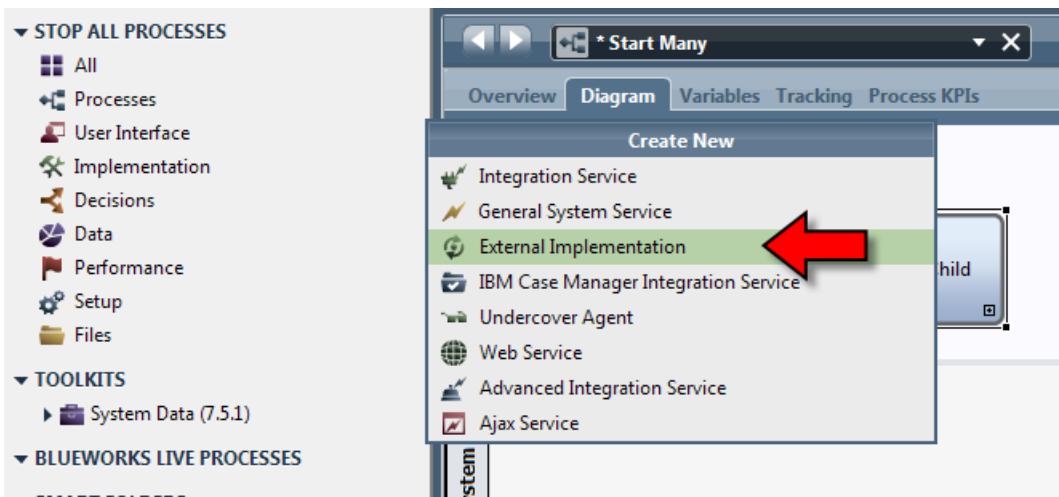
When an External Implementation is reached in the BPD, a new IBPM Task is created and the BPD process suspends itself until the task completes. Unlike a Human Service task, there are no sets of Coaches or other UI components provided by IBPM associated with this Task. However, the Task still exists and can be queried by the REST API or Web API. These APIs can be used by an arbitrary external application to:

- Query for the existence of External Implementation tasks
- Obtain the parameters passed as input to an External Implementation
- Set the parameters to be returned from an External Implementation
- Utilize custom properties set on the External Implementation implementation
- Complete the External Implementation associated task

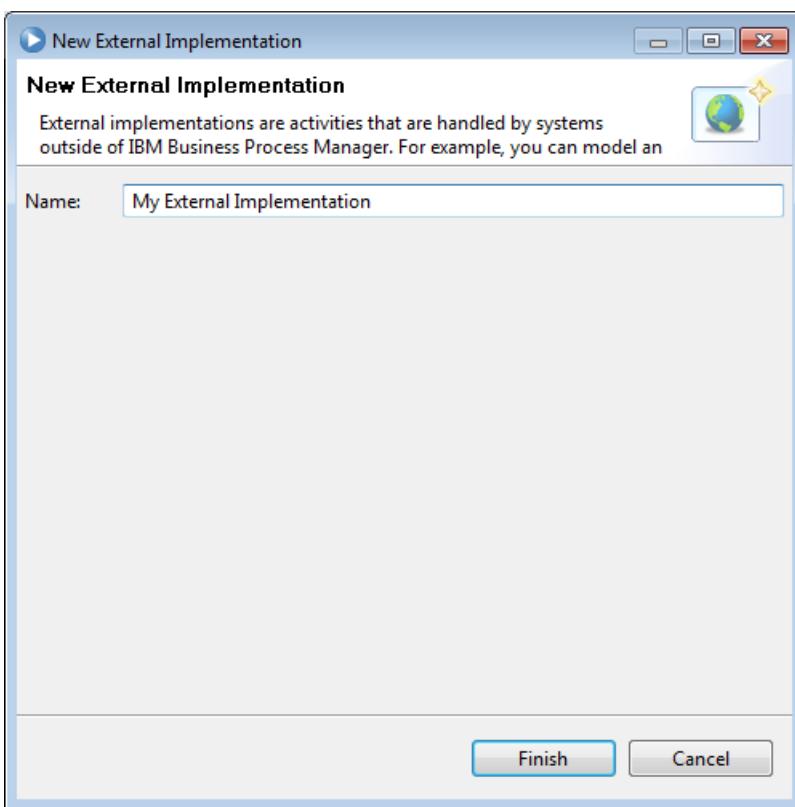
If we think about this for a while, we see that a BPD activity calling an external application through an Integration Service is an explicit invocation of that application while an External Implementation is much more focused on the External Implementation associated application polling and working with the data.

To use external implementations, this capability must first be enabled in IBPM PD.

An External Implementation can be created from the [Implementation Category](#):



A dialog is presented that allows us to enter the name of the activity to be created:



Once done, the properties of the External Implementation can be entered:

To use an External Implementation in a BPD, create an activity and change its implementation type to be User Task. Next, the identity of the external implementation previously defined can be used.

Completing an External Implementation – REST API

When we obtain the data for a `TaskInstance` object of a task, we can get a property from that object called the `ExternalActivityID`. If that value is null, then the task does **not** represent an external activity. However, if it is not null, then its value represents an External Activity identifier. A REST request called

```
GET /rest/bpm/wle/v1/externalActivity/{externalActivityID}/model[?parts={string}]
```

can be used to return the model of the external activity. This model describes the possible input and output data types for this activity. Here is an example of what is returned:

```
{
  "status": "200",
  "data": {
    "name": "EA1",
    "customProperties": {},
    "inputs": [
      {
        "in1": {
          "isList": false,
          "type": "String"
        }
      }
    ],
    "outputs": [
      {
        "out1": {
          "isList": false,
          "type": "String"
        }
      }
    ]
  }
}
```

```

        "type":"BO1"
    }
},
"outputs":
{
    "out1":
    {
        "isList":false,
        "type":"BO1"
    }
},
"validations":
{
    "BO1":
    {
        "properties":
        {
            "f1":
            {
                "isList":false,
                "type":"String"
            },
            "f2":
            {
                "isList":false,
                "type":"String"
            },
            "f3":
            {
                "isList":false,
                "type":"String"
            }
        },
        "type":"object"
    }
},
"ID":"60.0b902a52-8df2-44ed-85e2-04bbd980b01d"
}
}
}

```

Let us look at the inputs and outputs structures to begin with. These correspond to a list of variables where each variable has a "type" attribute. The type attribute matches an entry in the "validations" section.

When it comes time to obtain the input data for the External Activity, that data can be found in the Task Instance details object. To return data as output to an External Activity, use the REST Task Finish method (see: Finishing a Task).

When working with an External Activity, it is a good idea to build a notepad document that looks as follows:

EA1
Inputs:
order
customerName String
orderType String
amount Decimal
Outputs:

```

approval
  approved Boolean
  comments String

For a task input:
{
  data:
  {
    data:
    {
      variables:
      {
        order:
        {
          customerName:
          orderType:
          amount:
        }
      }
    }
  }
}

For task completion:
{
  approval:
  {
    approved:
    comments:
  }
}

```

This document starts with the name of the External Activity, notes its inputs and outputs and then shows the JSON for a GET request of a task. Finally it shows the response JSON to be passed when the task is completed. Building this notepad document and keeping it handy while designing and coding can save much time and effort.

Completing an External Implementation – Web Service API

One of the most common use cases for an External Activity will be completing it. To understand this part of the story, imagine that a BPD has reached a BPD activity that is defined as an External Activity. This will create an instance of a Task. Assume that by some means, an external application knows the identity (ID) of this task. The external application now wishes to complete the task. This will involve executing a Web API call called `completeTask()`. The input to this operation are two parameters:

- The identity of the task to be completed (a variable of type `long`)
- The output variables to be associated with its completion (an array of `Variable`)

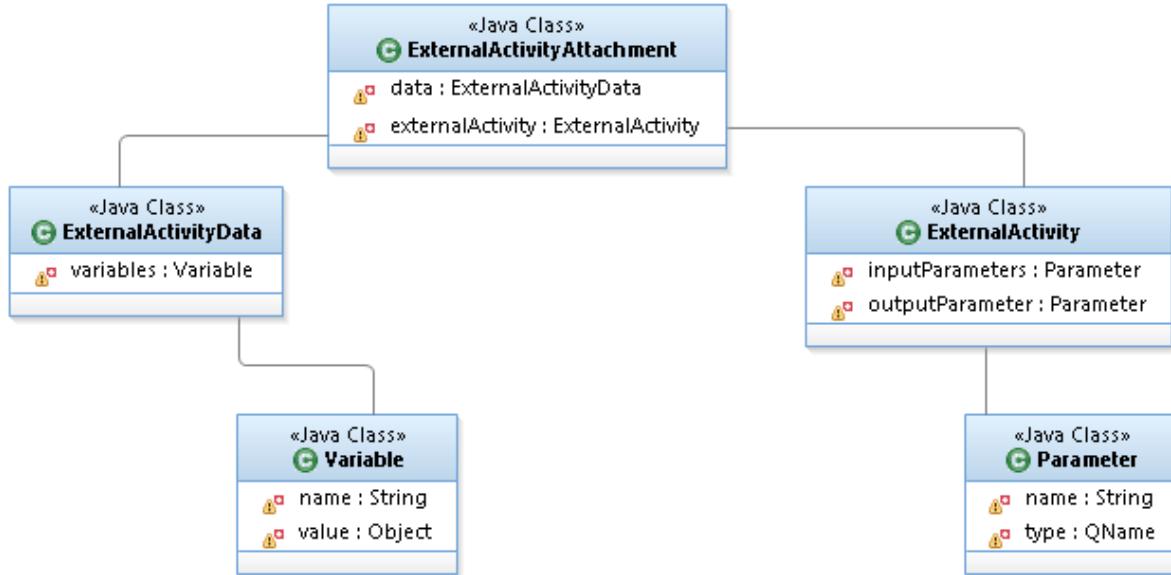
Since in this story, we are already assuming that we know the identity of the task, the missing component is knowledge about what the output variables should be.

When we query and retrieve the Task object for a task, we find it has an attribute called `attachedExternalActivity` of type `ExternalActivityAttachment`.

(Note: It is the authors belief that the concept of two names here is confusing. He believes that

"attached external activity" is the correct name as the notion is that of external activity being attached (or associated) with a Task.)

This structure looks as follows:



The key to understanding this is that there is an array of `Variable` objects called `data`. Each variable consists of a `name` and a `value`. If we next look at the input parameters to the Task, we see that those can be found at `externalActivity.inputParameters` which is an array of `Parameter` objects. Each parameter consists of a `name` and `type`. And here is the tricky/interesting part. For each parameter which has name "X", there is a variable which also has name "X". Thus for each input parameter we can now find its name, type and value. The same is also true for output parameters.

Getting mathematical for a moment if we think of the set of names of input parameters and the set of names of output parameters ... the union of these two sets will be the set of names of variables.

For parameters of "simple" types, these are self explanatory consisting of the usual suspects. However, if the parameters are complex types, more discussion is needed.

The data type is defined as a `QName` which is a qualified name consisting of the namespace of the data type and the name of the data type itself. For example:

```
{http://www.kolban.com}MyDataType
```

In Java, the value in the `Variables` of such a field is a "ComplexValue". This Object is a wrapper for DOM/XML data. It has a property in it called `_any` which can be retrieved with `getAny()` and set with `setAny()`. The data type of this property is a W3 DOM Element.

Here are some useful tips with working with one of these:

To convert to an XML String (in Java), use the following:

```

ComplexValue complexValue = (ComplexValue)value;
Element e = (Element)complexValue.getAny();
DOMImplementation domImplementation = e.getOwnerDocument().getImplementation();
DOMImplementationLS domImplementationLS =
(DOMImplementationLS)domImplementation.getFeature("LS", "3.0");
LSSerializer lsSerializer = domImplementationLS.createLSSerializer();
  
```

```
System.out.println(lsSerializer.writeObject(e));
```

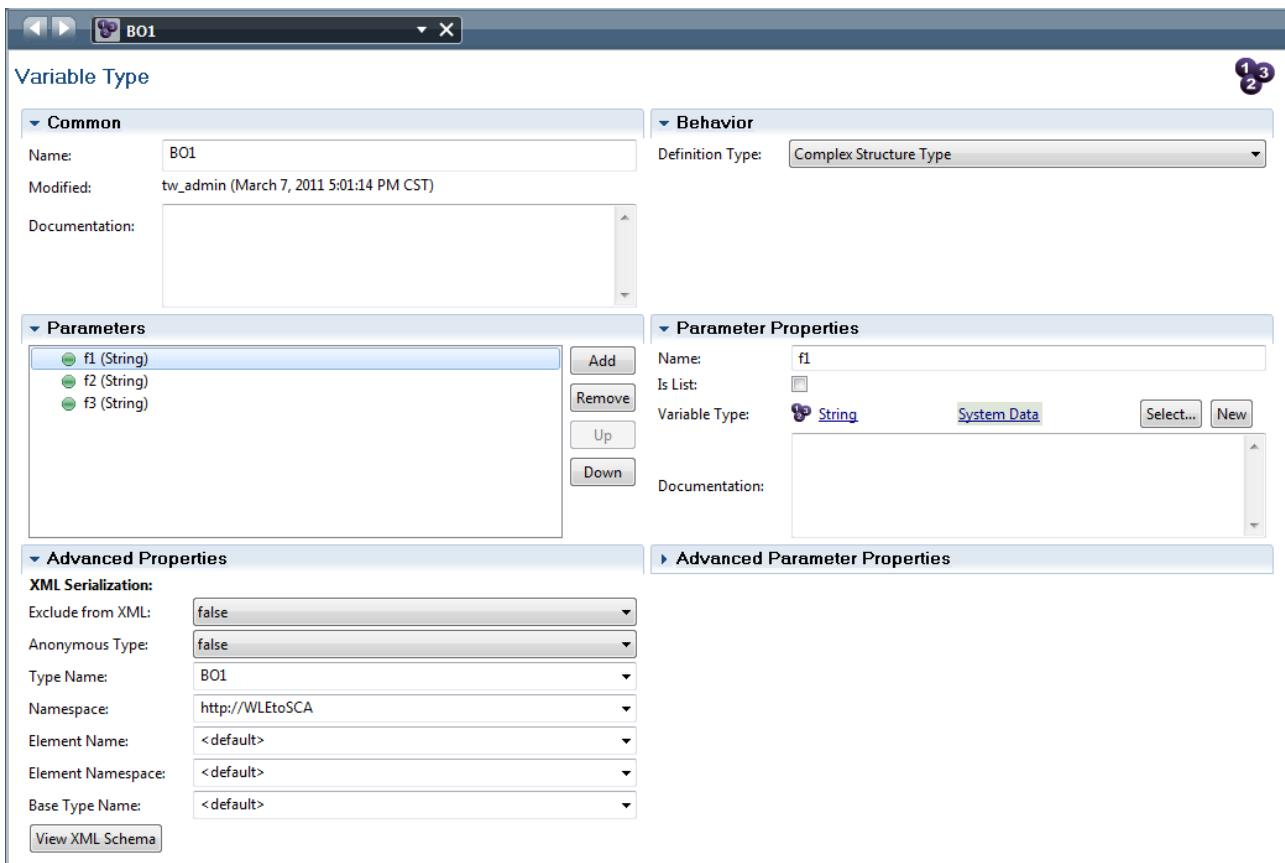
(Note: The DOM "LS" functions are "Load and Save" and are documented at W3 here:

<http://www.w3.org/TR/DOM-Level-3-LS/load-save.html>

)

To create a new output variable is a little trickier. Remember that a Variable consists of two fields, the name and the value. The value field is an Object. If the output type is a complex type then we need to create a ComplexValue object instance and set the Variable's value field to be the ComplexValue. We also need to set the data into the ComplexValue. Remember that this has a set_any() method which expects a W3 DOM Element.

Let us imagine that we have a data type that looks as follows:



Notice that we have expanded the Advanced Properties so that we can see them in more detail. By clicking on the View XML Schema button we can see the XML Schema for this data type. Now we have all the pieces we need. An example of a good value for BO1 encoded as XML would be:

```
<?xml version="1.0" encoding="UTF-16"?>
<ns4:BO1 xmlns:ns4="http://WLEtoSCA">
    <ns4:f1>f1Value</ns4:f1>
    <ns4:f2>f2Value</ns4:f2>
    <ns4:f3>f3Value</ns4:f3>
</ns4:BO1>
```

When we set the ComplexValue's any field, we need to set this to be a W3 DOM Element that

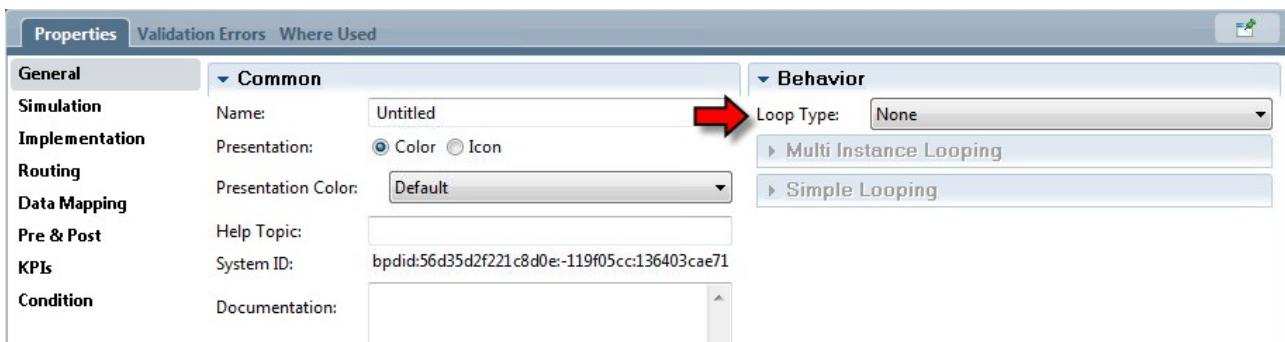
represents this piece of XML.

Notes:

- The author has not tested this scenario with code, so this is pure theory
- There may be an IBPM utility function supplied called `ClientUtilities.toComplexValue()` which may be of assistance in building this out.

Application Loops

Sometimes it is desirable for an activity to be executed a number of times in a loop. In the General properties of an Activity definition, there is a Loop Type indicator in the Behavior section of the panel:



The choices available are:

- None – No looping is to be performed
- Simple Loop – Looping is to be performed one item at a time
- Multi Instance Loop – Looping is to be performed in parallel

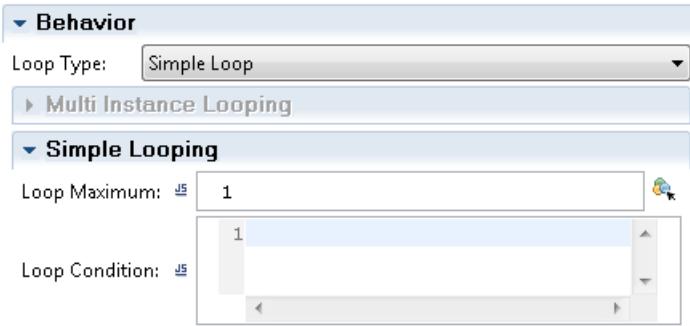
The default is None which simply means that there will be no looping. There is nothing further to say on that topic.

See also:

- developerWorks - [Common business process modeling situations in WebSphere Lombardi Edition V7.2, Part 3: Executing an activity multiple times in parallel in a business process](#) - 2011-08-03

Simple Looping

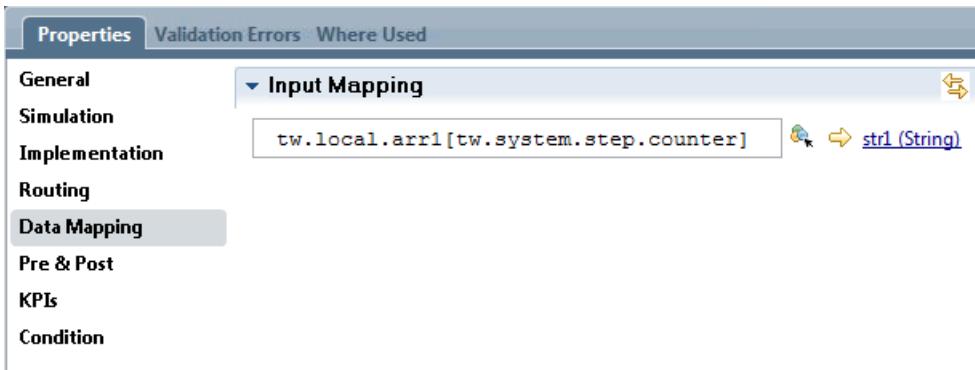
With simple looping, the activity is executed a number of times in succession. The number of times the activity is executed is based on the value of the Loop Maximum field which may be an expression or variable resulting in an Integer value. The optional Loop Condition is a boolean expression that is evaluated prior to the next occurrence of the loop. If the condition evaluates to false, the looping terminates early.



When a simple loop has been added to an activity, the icon representing the activity is updated at the bottom left to show that it contains a simple loop:



To pass data into the implementation of the activity, the local variable called `tw.system.step.counter` contains the current loop index which can be used (if needed) to index into a list. For example, in the Data Mapping section of the properties for the activity, we could code:

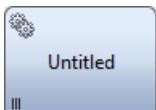


to access the current index of a variable array.

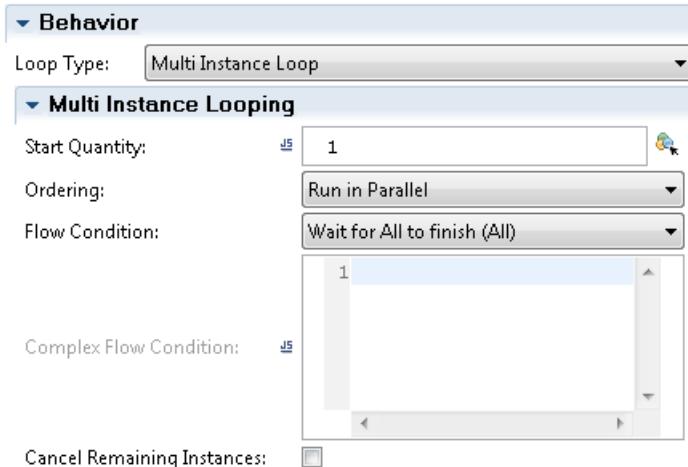
Multi-Instance Looping

Multi-instance looping allows for the activities in the loop to execute in parallel with each other. Another way of saying this is that the activities will execute concurrently.

When a Multi Instance loop has been added to an activity, it has an icon that indicates this in its lower left:



The properties of a Multi Instance Loop are shown as follows:



The Start Quantity field defines an expression of how many iterations to execute. The use of this phrase has caused puzzlement to some users. When we think "Start Quantity" we are sometimes drawn to the notion of "How many things do we start with?". What is really meant here is "What is the count (quantity) of things to start in total".

Each iteration of the loop can find its own iteration index through the variable called:

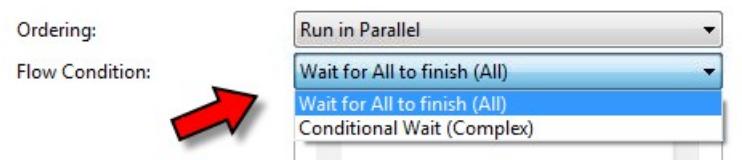
```
tw.system.step.counter
```

The ordering mode can be Run in Parallel or Run Sequential. When Run in Parallel is selected, the activities execute concurrently. When Run Sequential, the activities execute one after the other and the loop becomes a simple loop.

When Run in Parallel is selected, the Flow Condition option becomes available. This determines when the activity as a whole is considered completed. The options are:

- Wait for All to finish (All)
- Conditional Wait (Complex)

These options are shown here:



The Wait for All to finish means that all the activities must complete before the activity as a whole is considered complete.

The Conditional Wait requires an expression to be supplied. When an activity completes, the expression is evaluated and the activity as a whole can be determined to have completed or not based on the value of the expression. If the expression is "true" then we end. If the expression evaluates to false, then the loop continues. The way that I like to think of the expression is as the answer to the question "**Should we end the loop?**".

If all the possible instances have completed, the loop as a whole will also complete irrespective of the outcome of the last condition.

If a completion occurs as a result of the conditional wait expression, the Cancel Remaining Instances check box can be used to terminate any unfinished activities.

When working with looping constructs, the variable called `tw.system.step.counter` contains the current loop indicator.

The implementation types of a looping activity are as for any activity but not all of them make sense in this context. For example, the Sub-process is no more than a diagramming construct to replace a set of one or more BPD activities with a single BPD activity. The Sub-process executes in the BPD data context/scope and as such, has no differentiated variables associated with it. This means that each instance of a Subprocess started in parallel would have no distinction from others.

Looping through a map data type

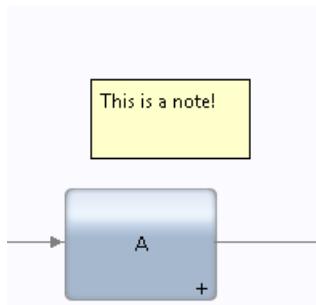
We can use a Multi Instance loop to iterate through a Map object (see: Data Type – Map). Assuming that the variable `tw.local.myMap` is a Map ... set the Start Quantity field to be `tw.local.myMap.size()`. In the Data Mapping section, we pass in the input to each of the service instances or sub-process instances. To supply data to those, use:

```
tw.local.myMap.valueArray() [tw.system.step.counter]
```

This line retrieves all of the values of the map and, for each value, starts a service/sub-process to handle the processing of the data item.

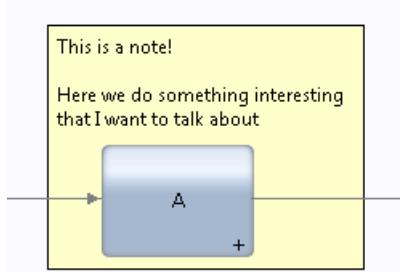
BPD Diagram Notes

Notes are documentation items added to the BPD diagram. They appear as yellow rectangles into which text can be entered. They have no operational semantics and do not affect the execution of the process.



Notes can be positioned anywhere on the diagram canvas using the mouse to drag and drop them. They can be re-sized to any desired size. Unfortunately, there are no options to change the background color nor to have the text contained within wrap to the note size.

One interesting way that notes have been seen to have been used is to place them over existing activities. Notes are always shown beneath activities so a note will never obscure an activity or sequence line. If the note is made larger than the activity, the effect can be useful and pleasing:



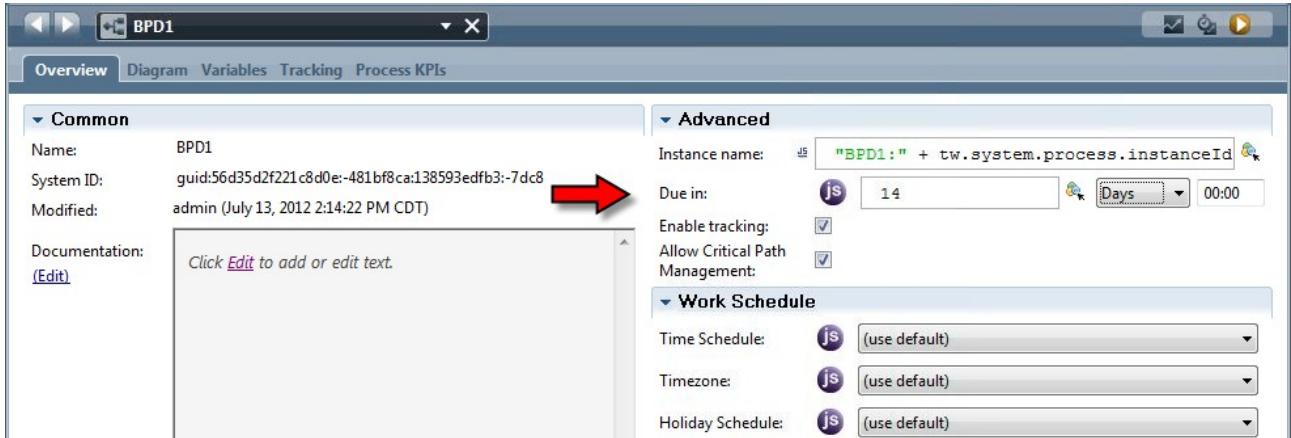
Don't forget that every activity has a Documentation section in which more details of that activity can be recorded. The difference between notes and documentation settings is that with notes, they are always visible while with documentation settings, the user has to explicitly examine the details

of an activity to see more information.

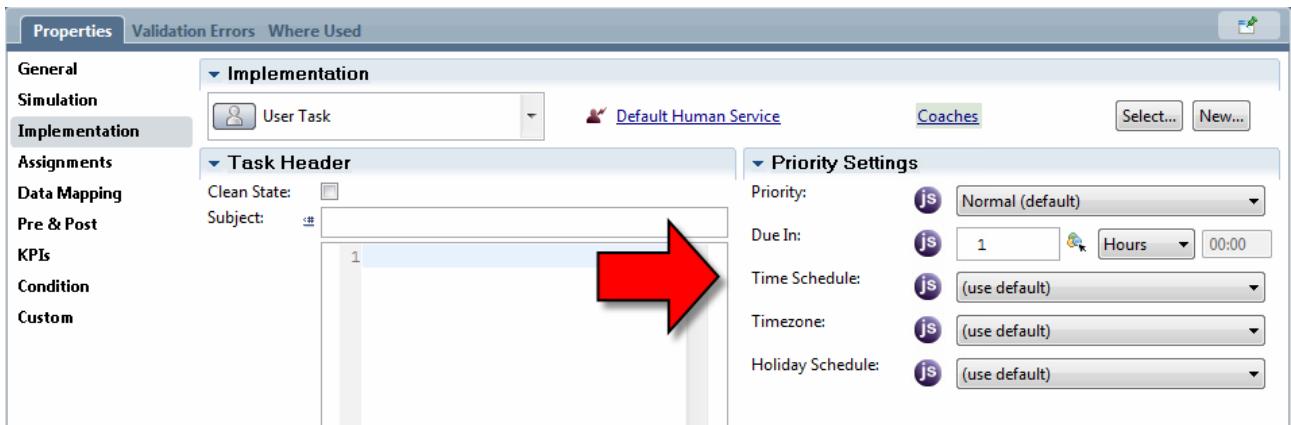
Due dates for process and activities

Each process instance has the concept of a due date. This is the date and time when the process instance is desired to be finished. By having this forward date from when the process instance was created, we now know how close we are to risk of it not meeting some desired target.

If the BPD Overview tab, we can set the "Due in" settings as shown next:



Similarly, we also have a corresponding set of details for each User Task in the the process:



The default values for the work schedules are defined in the 99Local configuration file. In 8.5 they are:

```
<default-work-schedule>
  <time-schedule>24X7</time-schedule>
  <time-zone>CST</time-zone>
  <holiday-schedule>empty holiday</holiday-schedule>
</default-work-schedule>
```

See also:

- Data Type – TWTimeSchedule

Time and Holiday Schedules

Consider a task being assigned to a user. Assume we say that it takes three hours of work to complete that task. If its 10:00am just now, then we can say that the time we expect the task to be completed will be 1:00pm which is three hours later. This means that the due date will be 1:00pm.

But what if the same task is assigned at 4:00pm? Using simple arithmetic, we might say that the

task is due to be complete at 7:00pm. However this does not take into account the business hours of the company. If the office hours are 8:00am to 5:00pm then it would be wrong to ask the staff to complete the task by 7:00pm. To solve this puzzle, we look at an additional BPM concept called "Time Schedule". The Time Schedule defines the working hours of a business. When the Time Schedule is applied to the due date calculation of a task, that due date is no longer a simple arithmetic calculation.

For example, if the Time Schedule declares that our working hours are 8:00am to 5:00pm and a three hour task is started at 4:00pm, we would calculate a due date of 10:00am the next day.

Not only does the Time Schedule contain working hours, but it also describes the working days. For example, a task started at 4:00pm on a Friday may not be due until 10:00am on the following Monday.

See also:

- Data Type – TWTTimeSchedule

Generating Reports

We can generate reports (documentation) based on the content of our Process Applications. From within Process Center we can select a Process Application and navigate to its Snapshots settings. Clicking on the name of the Snapshot takes us to the reports section:

The screenshot shows the SAP Process Center interface. At the top, there are tabs for 'Process Apps', 'Toolkits', 'Servers', and 'Admin'. On the right, it says 'Logged in as admin'. Below this, a navigation bar has 'TEST1 (A111121)' with a star icon, 'Snapshots' (which is highlighted in green), 'History', and 'Manage'. Underneath is a search bar with 'Sort Snapshots By: Date' dropdown set to 'All', and filters for 'Installed', 'Deployed', and 'Archived'. A 'filter' button is also present. The main area lists snapshots: 'Current' (highlighted with a red arrow), 'sn2', and 'sn1'. Each snapshot entry includes a small icon, the name, creation date ('Created on 11/21/11 by admin'), status ('Not Yet Installed to Process Server'), and 'Export | Install' buttons.

From there we get to see a summary of the major sections of our Process Application:

The screenshot shows the 'Process Apps' interface with the 'TEST1 > Current' tab selected. The main area is titled 'Process App Settings'. It contains several sections:

- Toolkits**: A list containing 'System Data (7.5.1)'.
- Coach Overrides**: A list containing 'Coach Transformation Override' and 'Coach CSS Override'.
- Business Monitor Tracking**: A section with the option 'Enable tracking using IBM Business Monitor'.
- Business Process Definitions**: A list showing 4 items.
- Human Services**: A list showing 1 item.
- Business Objects**: A list showing 2 items.

For each of these we can drill down and see more details and, in certain cases, generate reports for printing or transforming into PDFs for recording.

BPD Settings

A BPD has a set of attributes that govern the BPD as a whole. These attributes can be found in the Overview tab of the BPD editor in IBPM PD.

The screenshot shows the 'MyProcess' BPD editor in IBPM PD. The 'Overview' tab is selected. The interface is divided into several sections:

- Common**: Contains fields for Name ('MyProcess'), Modified ('tw_admin (September 15, 2010 3:39:16 PM CDT)'), and Documentation.
- Advanced**: Contains fields for Instance name ('"MyProcess:" + tw.system.process.instanceId'), Due in ('14 Days'), and Enable tracking (checkbox checked).
- Work Schedule**: Contains fields for Time Schedule ('(use default)'), Timezone ('(use default)'), and Holiday Schedule ('(use default)').
- Exposing**: Contains fields for Expose to start ('<none>'), Expose business data ('<none>'), and Expose performance metrics ('<none>').

Exposing the Process

The details of who can start an instance of the process and who has authority to examine the details

of an instance is defined in the Exposing section. There are three types of exposable information. Those are:

- **Expose to start** – Who can start a new instance of a process from the Process Portal web pages
- **Expose Business Data** – Who can see the data associated with a process in an ad-hoc report
- **Expose Performance Metrics** – Who can see the performance data collected for a process

See also:

- Error: Reference source not found

Process Instance Name

Each instance of the process has an optional name. The field called "Instance name" supplies a Java Script expression that is used to build the identity of the process. The default value is the name of the process with the numeric instance ID appended. For example:

```
"MyProcess:" + tw.system.process.instanceId
```

Take care when renaming a BPD. Experience has found that the default identity expression that is created when a BPD is created for the first time is not changed when the BPD is renamed. What this means is that if the BPD have the original name "XYZ", then the identity of a process instance may be "XYZ:1234". But if the BPD is renamed to "ABC" and the instance name expression not changed, an instance of "ABC" may be given an instance name that continues to refer to "XYZ".

When ever the identity of the process is obtained, the expression defined here is re-evaluated. What this means is that the name of the process can be dynamic and the value of the variables in the process at the time that the expression is calculated will be used. This can be used to good effect in conjunction with the IBM Process Portal where the names of the process instances surface.

BPD and BPMN

Originally called the Business Process Modeling Notation, the latest BPMN 2.0 specification has renamed itself to the Business Process Model **and** Notation. The purpose of this section is to contrast what is contained in the BPMN specification against what is found in the BPD diagrams of the IBPM product. We will not describe BPMN in depth as that would be the subject of its own book (the spec itself is 550 pages) and the reader is referred to one of the many references including the BPMN specification itself.

See also:

- [BPMN Specification page](#)

Concept of a Token

As a process runs from its Start to its End, the process executes a sequence of activities. The sequence of executed activities is governed by the Sequence Flow (lines) between the activities. Assuming that there are no parallel branches in the process, at any one time, a single Activity is being performed. To aid in understanding this idea and to be able to discuss in more depth, we introduce the logical concept known as the *token*. A token is an imaginary item and is not formally

part of either the BPMN specification or the IBPM product. A token can be considered to be at exactly one activity in a process and the activity which has the token is available to run or is actually running. When an Activity completes, the token can be thought of as moving along an outgoing Sequence Flow to the next Activity. To help your understanding, you can think of the token as moving **instantly** from one activity to the next ... another way of saying this is that it spend no time traversing the Sequence Flow itself.

A dictionary definition of a token is very appropriate:

1. something serving to represent or indicate some fact, event, feeling, etc.; sign: *Black is a token of mourning.*

See also:

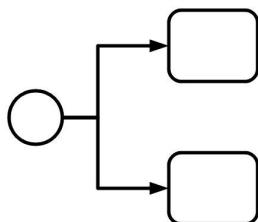
- Token Management
- Moving a token within a process using REST

Deviations from BPMN

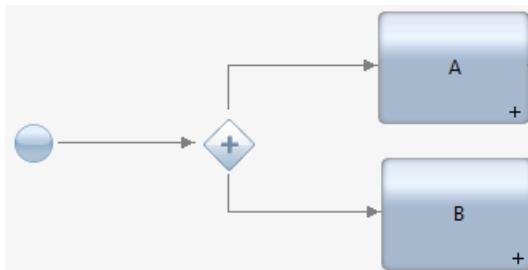
IBPM supports much of the BPMN story but there are some deviations in the implementation of the IBM product that are worthy of note.

Parallel Starts

Here we have a BPMN fragment that shows that when a process starts, in BPMN it will execute two activities in parallel.



If we try and code this in IBPM, it won't work. Only one path or the other will be taken but not both. To achieve the parallelism we desire, we need to add an explicit Parallel Gateway.



Other than stylistic differences, the two are semantically identical.

Looping

BPMN provides the concept of a loop which looks like an activity. If the activity is expanded, we can see the contained activities which are to be executed repeatedly. One common way of using looping is to repeat an activity until something has been "satisfied". Consider a loop that periodically checks that an oven has reached a specific temperature. If it has, we end the loop if

not, we check again some time later. This is a loop concept that in Java might have been called a while-do (or do-while) loop. A predicate (boolean valued expression) determines when the loop is to terminate.

Another (but distinct) looping construct in BPMN is the for-each loop. This is a loop which is executed some finite number of times with the activities contained within executed sequentially or in parallel. This looping construct has a distinct appearance from the other loop.

IBPM provides a looping construct that is based on the for-each looping style. However, the visual representation of the IBPM activity is that of the while-do loop style. IBPM does not provide a separate while-do activity style.

XPDL Support

XPDL is an Open Standard describing the exchange of BPMN diagrams as XML documents. IBPM does not support either import or export of XPDL format documents.

Boundary exposes Start Events

A process diagram can have more than one Start Event defined within it if the Start Events are placed on the boundary of the diagram. This allows a single Process to be started at multiple potential start places by its caller.

IBPM does not support this pattern.

Nested terminates

When a parent process calls a child process and the child process executes a Terminate Event, the semantics say that the child process should immediately terminate and the parent process then continue to its next steps. Unfortunately in IBPM, we find that if a child executes a Terminate Event, both the child and its parent are terminated.

Other omitted BPMN items

Other omitted items from the BPMN specification include:

- No Signal Start Event
- No Conditional Start Event
- No "None" Intermediate Event
- No throw type Message Intermediate Event
- No Signal Intermediate Event (throw or catch)
- No Cancel Intermediate Event
- No Compensation Intermediate Event (throw or catch)
- No Conditional Intermediate Event
- No Link Intermediate Event (throw or catch)
- No Multiple Intermediate Event (throw or catch)
- No Multiple Start Event – Although omitted, the same effect can be achieved by simply having

- multiple other start events of different types in the process.
- BPMN permits process models to omit start and end event symbols. This is not permitted within IBPM.

BPD Flow Control

From time to time we may want the path of the process to flow in different directions based on a variety of factors. This section describes some illustrations of this and possible ways to achieve the effect.

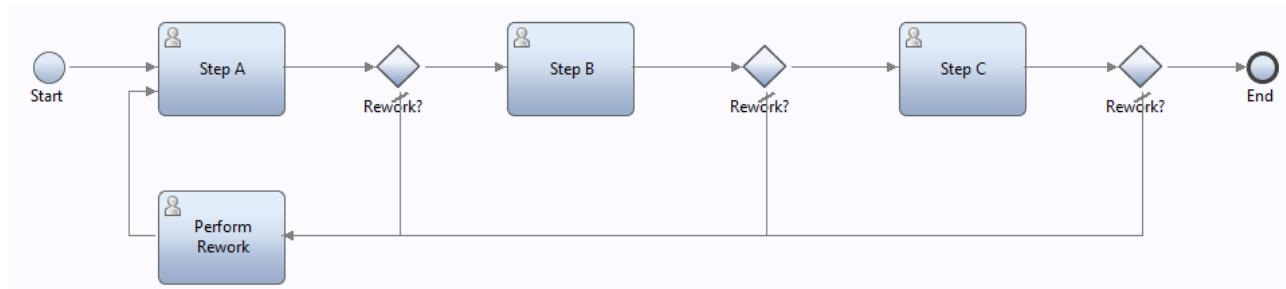
Rework a set of steps

Consider a sequence of steps as follows:



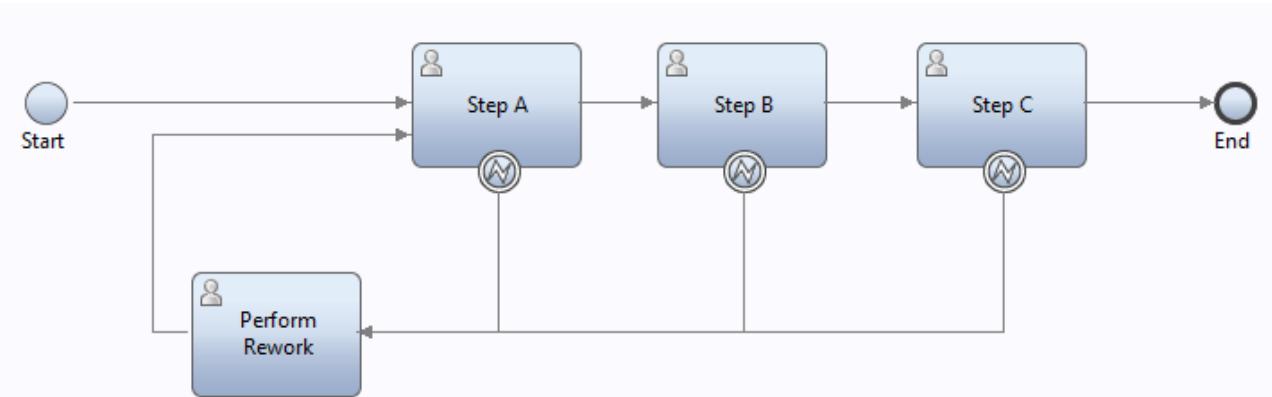
While working on a step, a user can say that something is "wrong" that will result in the process as a whole being restarted after some data changes. For example, in "Step B", a user could see something that invalidates the whole process. The user should then be able to press a button that would cause a task to be created that allows someone to modify the process instance data and then we begin again from "Step A".

One solution to this puzzle would look as follows:

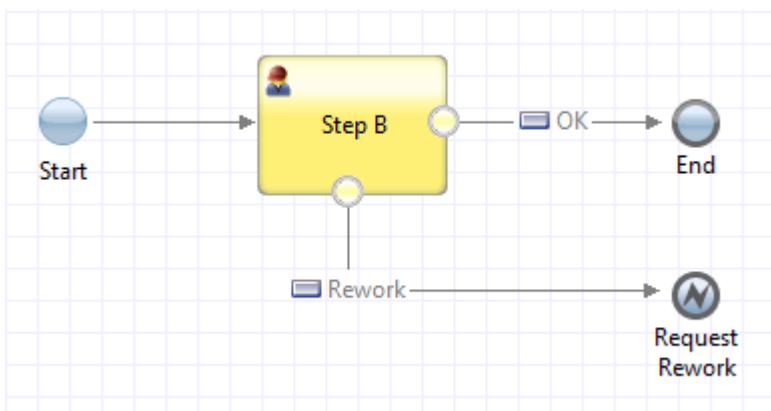


Here, after a Step completes, we now realize that it could complete because a rework was requested. After each step we then check if a rework was indeed requested and if it was, we then explicitly go and perform that rework. This is a good solution. It is very visible on the diagram.

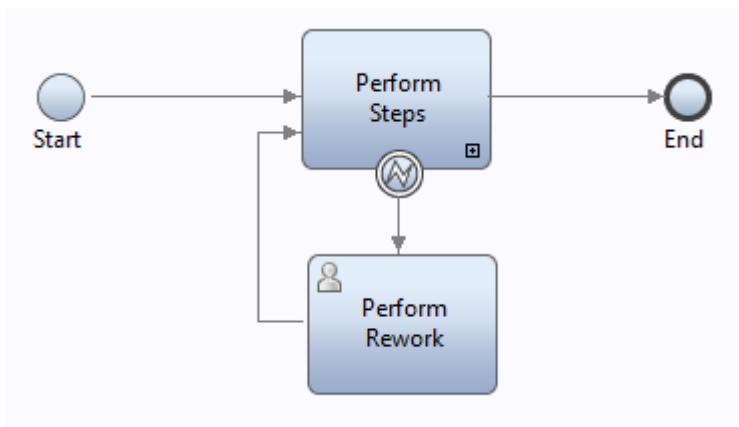
Another solution would be to use the concept of exceptions. Now, when a rework is requested, the Step itself throws an exception which might be called "Rework". The diagram may now look like:



Here we are saying that we have an "expected" path and a rework request, although very valid, we call an exception. Within the logic of the Human Service that implements the step, we would now raise the exception:



A third choice would be the use of the sub-process. We would have a top-level process that looks as follows:



The Perform Steps activity is a sub-process that contains:



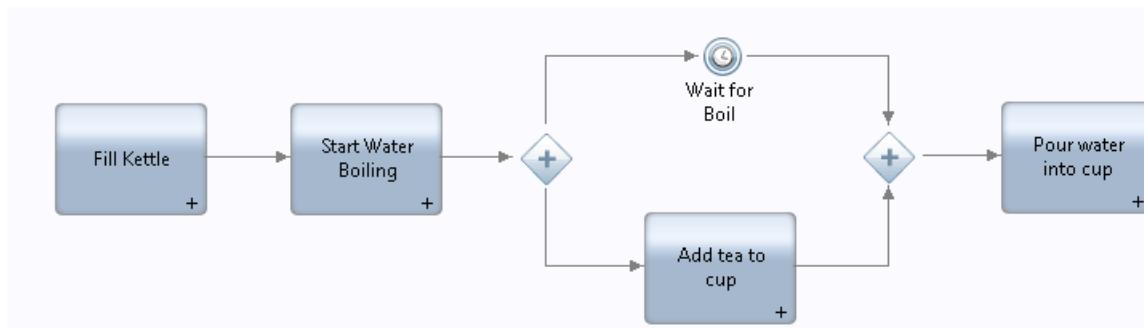
Each step can throw a Rework exception which is then caught by the container process.

All three of the techniques described here are equally valid. Which one you choose is as much a

matter of taste. The results should be identical. The choice used would be to the taste of the designer as to how best they comprehend the resulting diagrams so that they can be read and understood by others.

Services

The activities contained in the BPD describe the overall flow of the process and the diagram as a whole provides a readable description of the process. Although the BPD provides a great deal of information, it doesn't capture *how* the process is implemented. Consider the simple process of making a cup of tea in the morning. A simple process for this might look as follows:



Reading this, we can quickly see the steps and their sequence. However, this diagram says nothing about how these steps are implemented. Since IBPM is not simply a diagramming tool but is also a powerful execution engine for running instances of such processes, we must also describe how each of the activities are implemented.

IBPM utilizes the concept of a *service* to perform each step. A service provides a reusable implementation of a piece of function. How the server achieves its task is hidden from the caller. Mapping this idea to BPDs, each Activity in the BPD diagram will be mapped to a service and when that step in the process is reached, the associated service will be invoked.

A service is a piece of implemented function of one of the following IBPM types:

- Human Service
- Ajax Service (used only by Human Services)
- Integration Service
- General System Service
- Rules Service

The Service is usually called from the Activity step in a BPD. The identity of the Service to be invoked is defined in the Implementation tab:

Note that the Implementation type is defined as a System Task. From there, the selection of the Service to be called can be selected.

The Task Header area controls what is shown in the Process Portal for this step and what data is maintained after the activity completes.

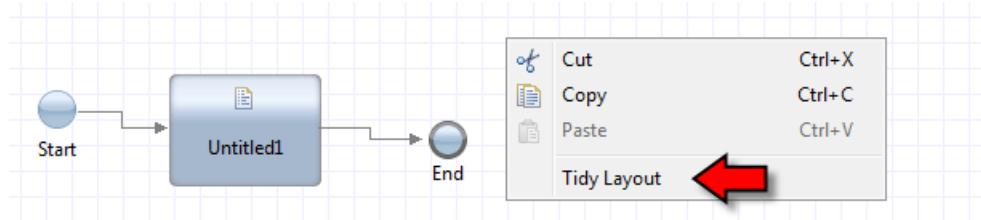
The Subject entry contains a textual description of this step in the process. This is shown in the Process Portal. It is used to provide the end user an indication or summary of what is expected. The Subject line can contain JavaScript and hence access the variables in the process to supply more details of what is needed.

For example, if a step in the process has the following Subject line:

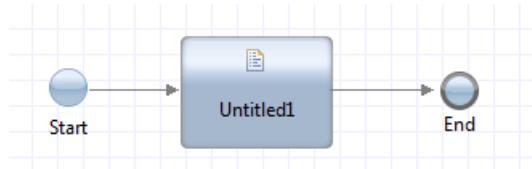
When it is shown in the Process Portal, it could look as follows:

If the Clean Slate check box is selected then at the conclusion of the activity, state data about the activity is discarded.

Each of the service types has a diagram in which the artifacts which implement that service can be placed and visually wired together. The service editors have a feature within them to "tidy" their appearance by lining up activities to smooth out the sequence connection lines. This option can be found when right-clicking on the diagram canvas with nothing else selected. From the resulting context menu, the Tidy Layout option may be selected.



After having selected that entry, the diagram will be reworked to tidy it. An example after a tidy looks as follows:



Caching the results of a service

When a service is executed, it is passed in a set of parameters and returns a set of results. If the service is executed over and over again with the same input parameters, it is likely that the result will be the same even though the work of executing the logic of the service will still be performed. As an optimization and as a way to improve performance, we have the ability to flag that a service is eligible to cache its results. This means that for a given service with a given set of inputs, the service will be executed but the results stored. If a subsequent request for the same service with the same set of inputs should arrive, the previously calculated results will be returned without having to execute the logic of the service itself.

This feature can be turned on or off on a per service basis. In the Overview tab of the service definition, we find an entry called "Service Result Cache":

Common	
Name:	Task Test
Modified:	kolban (Aug 11, 2013 10:06:12 PM)
System ID:	guid:1c834f97110f1739-7342c283:1406813d2a5:-7ff6
Click Edit to add or edit text.	

Service Result Cache			
Enable caching of service results: <input checked="" type="checkbox"/>			
Keep the cached results for the following duration:			
0	Days	12	Hours
0	Minutes	0	Seconds

By default, the caching of the service is disabled. If we enable it by clicking on the check box, the results of that service will be cached. We also have the ability to specify how long cached results should be maintained before we consider them "stale". We can supply an interval value after which a previously returned entry will be considered out of date and the real service will again be executed.

The implementation of the caching mechanism in IBM BPM has a default cache size of 4096 entries. This can be changed through the <service-result-cache-size> property in the configuration files.

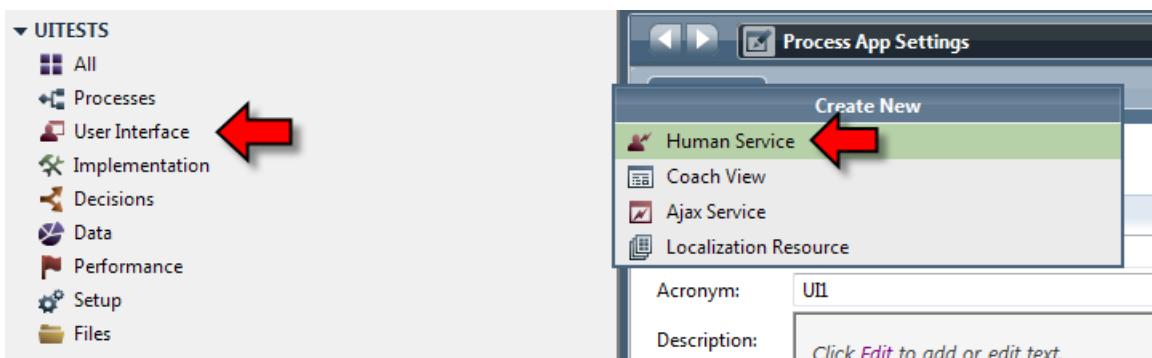
Notes:

1. It is not yet known if 4096 is the size of all service cache entries or a single service cache size
2. It is not yet known what happens when we try and cache more than 4096 entries. Will the latest entry simply not be cached or will another cache entry be purged?

Human Service

The Human Service is arguably the most important service type in IBPM. A Human Service is a component that, when executed, creates a task for a human being to work upon. The BPD process is suspended until the conclusion of that task. The Human Service type provides the inclusion of Coach components which are the descriptions of screen data to be presented to users and data to be retrieved from users.

A new Human Service is created from the User Interface category.



It is extremely common to associate the Human Service definition with a User Task on a BPD.

See also:

- User Task

Exposing the Human Service for starting

In the Overview panel of the Human Service definition, the Exposing section allows us to select how the service should be exposed for starting in a manner other than being explicitly called by a BPD.



The choices available to us are:

- Not Exposed
- With this option, the Human Service is not exposed any further. It can only be invoked by a direct Human Service call from a BPD activity. This is the default setting.
- Administration Service

This option defines the Human Service as appearing within the Process Admin Portal as an administrative activity.

The screenshot shows the IBM Process Admin Console interface. At the top, there's a navigation bar with links for Server Admin, Process Inspector, Installed Apps, Preferences, and Log out. Below the navigation bar is a sidebar containing links like IBM BPM Admin, User Management, Monitoring, Event Manager, Admin Tools, My Password Management, and My Process App Name. A red arrow points to the 'My Process App Name' link. Underneath this sidebar is a main content area titled 'Step 1' with a 'Section Title' input field and an 'OK' button. At the bottom left of the main content area, another red arrow points to the 'My Human Service' link.

See also

- Customizing the Process Admin Console.
- Startable Service

This option allows a new instance of the Human Service to be started from within the Process Admin Portal. It appears as a new entry in the New menu.

The screenshot shows the IBM Process Portal interface. At the top, it says 'IBM | Process Portal'. On the left, there are two sections: 'My Tasks' (Inbox, History, Help Requests, Alerts) and 'My Scoreboards' (My Performance, My Team Performance, Process Performance, SLA Overview, Ad Hoc Reports). The main area is titled 'Inbox' and contains a toolbar with icons for New, Open, Close, etc. Below the toolbar is a list of items: Ad Hoc Test 1, BondProcess, HSx1, HT1, and My Human Service. A red arrow points to the 'My Human Service' item. To the right of this list is a table showing process instances:

Instance Name	Process Definition
Ad Hoc Test 1:658	Ad Hoc Test 1
BPDx:766	BPDx
BPDx:768	BPDx
BPD P1:803	BPD P1

At the bottom of the table, it says '4 results (0.01 seconds)'.

See also: Error: Reference source not found

- Dashboard

This option exposes the Human Service as an entry in the Dashboards menu found on the IBM Process Portal:

The screenshot shows the IBM Work interface. At the top, there's a dark header with the 'IBM' logo, a 'WORK' button, and a 'DASHBOARDS' button with a dropdown arrow. Below the header is a grey navigation bar with 'My Tasks' and 'Saved Searches'. The main content area is titled 'My Tasks' and shows a section for 'Overdue (3)'. A red arrow points to the 'DASHBOARDS' button in the top right corner.

When the menu is pulled-down, the name of the exposed Human Service can be found here:

The screenshot shows a dropdown menu from the 'DASHBOARDS' button. The menu items are: 'My Performance', 'My Team Performance', 'Process Performance', 'My SLA Overview', 'PortalTest1' (which is highlighted with a red arrow), and 'Ad Hoc Reports'.

For Human Services selected to be exposed through the Dashboard, we have an addition option which allows us to export a JSR-286 compliant portlet.

The screenshot shows the 'Exposing' configuration dialog. It has sections for 'Exposed to:' (with 'All Users' selected) and 'Exposed As:' (with 'Dashboard' selected). Below 'Exposed As:' is a link 'Create a Portlet from the Dashboard' (which is highlighted with a red arrow).

See also:

- Exposing a Human Service as a portlet
- URL

With this option, the Human Service is exposed as a directly invokable URL. If the URL is entered into a browser, the user can go directly to the Human Service.

The format of the URL is:

```
http://<hostname>:<port>/teamworks/executeServiceByName?
processApp=<processAppAcronym>&serviceName=<serviceName> [&tw.local.variableName=
value ...]
```

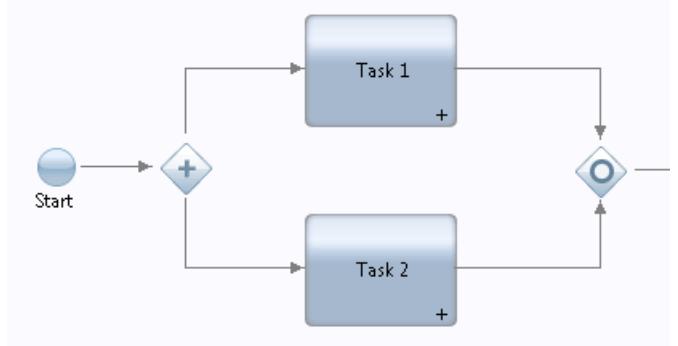
If the user has no login session established with that browser then a prompt for a login will occur. If the Human Service is exposed as an Administration Service, a Startable Service or a Project Page then it is **also** exposed through the URL in addition to the other selected mechanism.

For each of the exposing options, a Participant Group must be selected to determine if the requesting user is authorized to access the service. This Participant group is set in the Exposed to: area.

The screenshot shows the 'Exposing' configuration dialog again. In the 'Exposed to:' section, '<none>' is selected. In the 'Exposed As:' section, 'Not Exposed' is selected (which is highlighted with a red arrow).

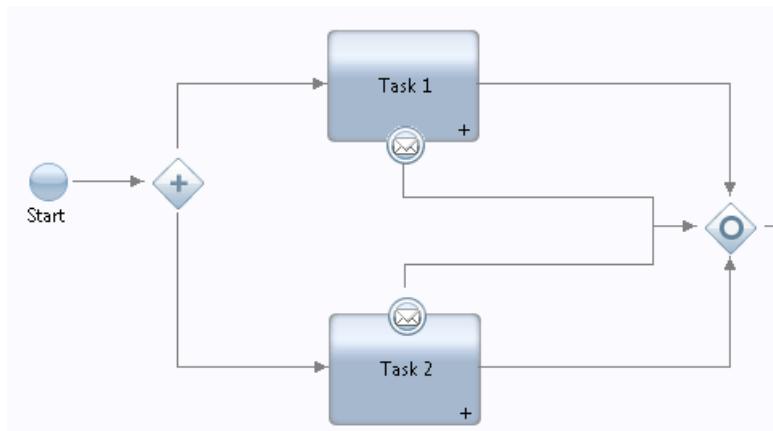
Cancelling outstanding Human Tasks

Consider the notion that we have multiple Human Tasks being executed in parallel as illustrated in the following process diagram. These tasks may consist of approvals where in order to make progress, both tasks have to result in an approved outcome.



If a person reviewing the task work decides to reject the request then we don't want the other person to waste time reviewing. Even if they approve their part, the process as a whole says that the work will be rejected. What we want to achieve is a pattern where we can cancel a task before it is completed if it is no longer warranted that the task be performed.

The way this can be achieved is to add Message Intermediate Events to the tasks that are eligible for being canceled. Here is a new diagram that illustrates this:



The definition of the Message Intermediate Events looks as follows:

Properties Validation Errors Where Used

Step

Simulation

Implementation

Pre & Post

Attached Event Details

Close Attached Activity:

Repeatable:

Message Trigger

Attached UCA: [Cancel UCA](#) .

Condition:

1

Consume Message:

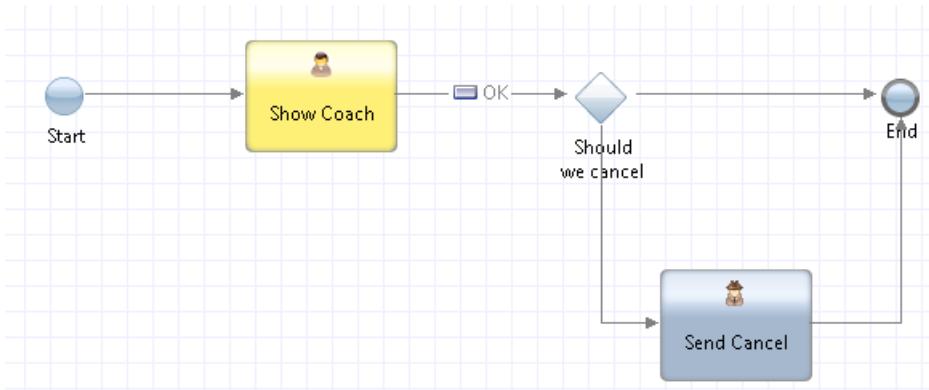
Durable Subscription:

UCA Output Correlation

[ProcessID \(String\)](#)

Note the properties carefully as they are important. The first is that the attached activity (which is the Human Service) is flagged to be closed when the event arrives. This is what ends the associated task. Notice also that Consume Message is unchecked. We want the same event to be propagated to all the tasks that may be waiting for a cancel. Finally notice that we are using the current process instance identifier as the correlation value. We don't want to cancel tasks that belong to other processes.

The Human Service that is shown is defined as follows:



In this example, the user interface shows a check box that allows the user to define if a cancel is needed:

Lombardi Coach

Section Title
Cancel: <input type="checkbox"/>
<input type="button" value="OK"/> <input type="button" value="Cancel"/>

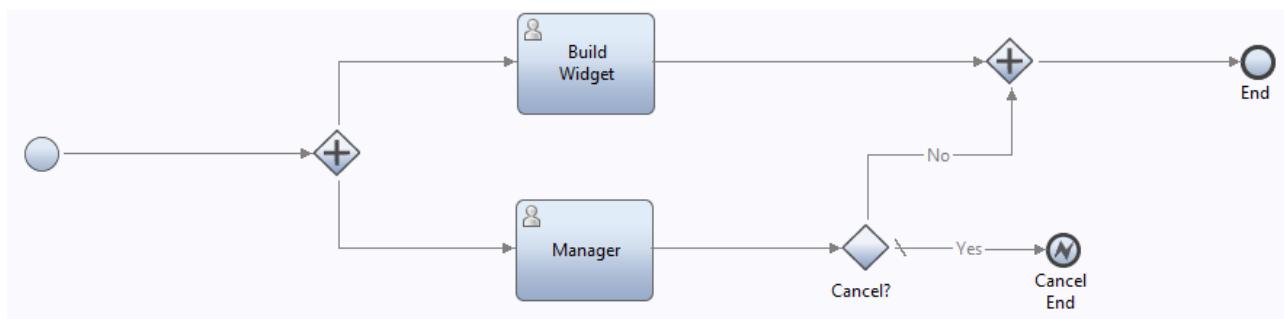
If cancel is checked, then the UCA is triggered which terminates the other task.

Informing a user that their task has been canceled

Imagine a scenario where a user can be given a task and they can look at that task and then signal it completed later. This is not uncommon. For example, imagine a task arrives that says "Go build a green widget". This might take a few days. Having looked at the task, I now go away and start building the widget. What then if the customer cancels the order. Presumably I don't want to continue building the widget so now I have to be notified that I should be interrupted.

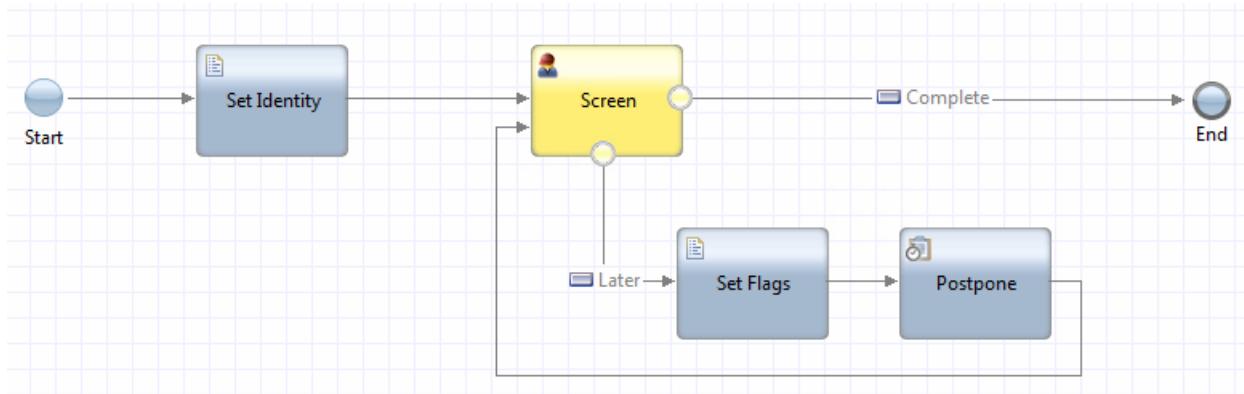
When a task is assigned to a user, either the user has looked at the task or they haven't. In our story, we want to send an email to a user if they have looked at the task and it is then canceled. If they haven't yet looked at it, there is nothing for them to stop doing when it is canceled and hence no email needs to be sent.

Let us start by looking at a high level story:

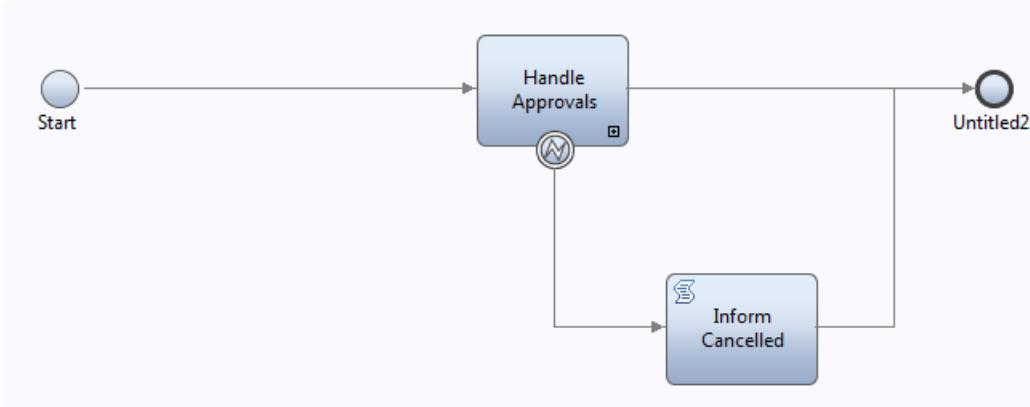


In this story, we can Build a Widget and also the Manager can cancel the request. If the manager selects to Cancel, the exception end event is reached and the sub-process is terminated. That handles the termination, but what about informing the user? The logic of Build Widget handles some of this for us.

When a user claims a task, a step in the solution called "Set Identity" remembers the identity of the user who worked on the task. When the user reads the task, we assume that they hit a button called "Complete Later". This is our indication that they have seen the task. We then set a flag that they have viewed the task.



This however seems to leverage data that appears to be private to the service. Where is the identity saved? Where are the flags saved? The answer to this puzzle is the use of the IBM BPM Shared Business Object concept. Using this technology we can pass in a reference to the shared business object into the service. This business object can then be examined in a calling process.

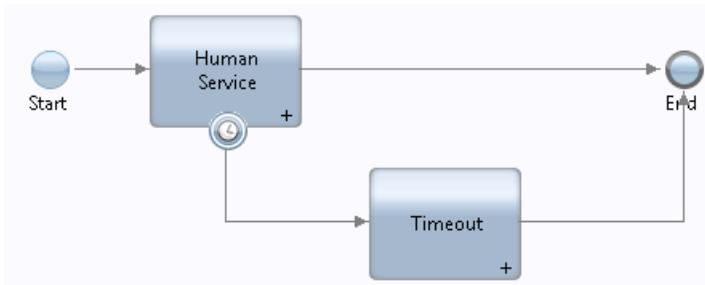


For example, if we catch an exception, we can use what was previously set in the global shared object as indicators of "where we have been" and "what we should do".

Timing out a Human Service

When a process reaches a step which is a Human Service, we are saying that we want a staff member to perform some work. But what if that staff member takes too long or is otherwise away? The process will simply pause or hang at this point. Instead of hanging, we may wish the process to time-out the Human Service and perform an alternative task such as dispatch the work to someone's manager or an alternative person.

We can achieve this through the use of a Timer Event attached to the activity of the Human Service.



See Timer Events for a discussion on the mechanics of Timer Event additions.

If a Human Task has been claimed by a user but has not yet been run, if the activity is timed-out and the user then tried to run it, the following will be shown:



If a Human Task been claimed and run but not completed but a Timer Event has fired and closed the attached activity, the user will **not** be notified that their input was discarded.

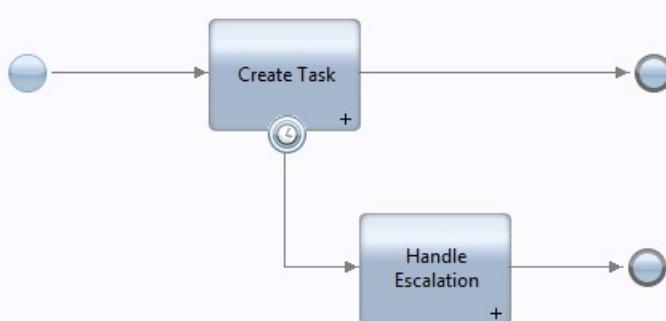
Escalating a Human Task

When a task is submitted to a person for completion, we have the opportunity for that task not to be completed within a prescribed amount of time. Work submitted for human processing can be "lost in the shuffle". People fall sick, take vacation, get distracted, get preempted by other work and a myriad of other possible reasons why work is not completed on time. The ramifications of not completing work vary. In some cases, it is unimportant, in others it can mean a poor perceived quality of service and in the worst case can result in loss of business or penalties for non-

compliance with some set contract or expectation. To address this potentiality, we now turn our attention to the concept of escalation. Escalation is the detection that a task has not reached a processing state we expected by a given time and then taking some remedial action. Ideally, we want to notice that the work state is not where we want it to be before it is too late and we can still do something about it to complete the work before it becomes a problem.

Once we detect that work has not reached the state we expect it to be, we need to perform some action to move the work onwards. From these notions, we find that escalation has two distinct ideas. The first is the detection that an escalation needs to happen because we have not reached the work state we want to reach. The second is we want to perform some escalation action to make further progress. We can separate out these two concepts.

The detection of *when* a task needs escalated can be achieved by attaching a timer event to an activity. The following BPD fragment illustrates a timer event attached to an activity that will cause a handler for the escalation to be executed.



The mechanical details of using a timer event can be found here: Timer Events and won't be repeated. For this discussion, we will simply assume that a timer will fire when a task is *late* or about to be late and the original task will **not** be stopped.

This pretty much takes care of detecting that a task needs to be escalated, so what action should we perform once escalation is required? There are many choices here. Some of the more common patterns include:

- Notifying the owner of the task that it is late
- Notifying the team leader/supervisor/manager of the owning department that it is late
- Automatically re-assigning the task to another individual or team
- Changing the priority of the task

Any of these could utilize the creation of new tasks, sending emails or invoking arbitrary services which could in turn use other technologies such as text messaging, web service calls, paging etc.

For each of these choices, we now need a mechanism that will enable to determine *which* task needs attention. When a task is created, it is created by an Activity in the BPD. There is thus an association between the Task that is created and the Activity which caused it to be created.

In the environment of a BPD there is a system variable called

```
tw.system.currentProcess.tasks
```

This is a list of TWTask objects. Each entry in the list corresponds to a task currently associated with the BPD. By walking this list and looking at the processActivityName attribute of the task, we can look for tasks that were started by the Activity we are looking for. An example code fragment that achieves this is:

```
var task = null;
```

```

for (var i=0; i<tw.system.currentProcessInstance.tasks.length; i++)
{
    if (tw.system.currentProcessInstance.tasks[i].processActivityName ==
activityName)
    {
        task = tw.system.currentProcessInstance.tasks[i];
        break;
    }
}

```

An alternative is to add a post-assignment to the Timer Event to obtain the `tw.system.step.task.id` information. This will be the id of `TWTTask` associated with the activity that was timed out. We can't save the actual task itself, this produces a not-serialized exception if we try to assign to "ANY". However, once we know the task's id, we can use the `tw.system.findTaskByID()` method to retrieve the `TWTTask` by its id.

We can change the priority of the task by setting the `task.priority` field which is writable. Allowable values include:

- `TWTTask.Priorities.Lowest`
- `TWTTask.Priorities.Low`
- `TWTTask.Priorities.Normal`
- `TWTTask.Priorities.High`
- `TWTTask.Priorities.Highest`

If a task needs to be escalated, we may need to know to whom the escalation should be sent. From the Task we can get the `TWUser` (or `TWRole`) from the `assignedTo` property.

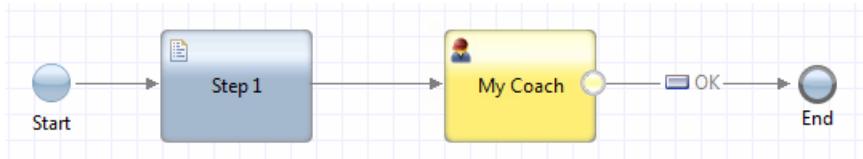
Notifying a user that a task is ready for work

When a Human Task is created by a process, it is ready for someone to work upon it. Typically, the user will be informed that the work is available to them as a result of them looking in their in-box in the Process Portal or through REST API calls. There is also the ability to have IBPM automatically send an email to the user when a task is created. This is set in the Process Portal preferences section.

Within a current task, the current Task ID can be retrieved from the variable `tw.system.task_id`. This can be used by a service or code in the Human Service to send an email to the target user to tell them the task upon which they should work.

Semantics of closing a browser window

Consider the following Human Service:



Imagine that it is started as a task by a process. When a user claims the task, the step called "Step 1" is executed and then the Coach described by "My Coach" is shown to the end user. What happens if the user simply closes the browser? What becomes of the task?

By experimentation, it appears that the state of the service/task is maintained at the point where the

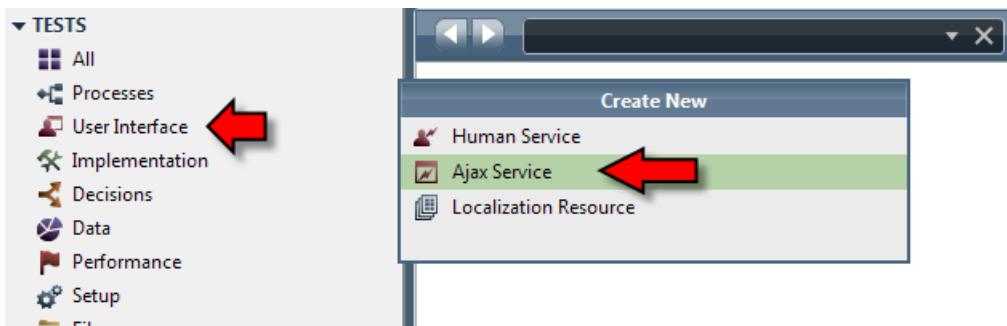
Coach was reached. The task remains in existence in a claimed state. If the task is re-executed, the Coach is again displayed **without** re-running "Step 1". This implies that the state of the service is hardened and restarted when a browser window showing a Coach is closed and then re-opened.

Ajax Service

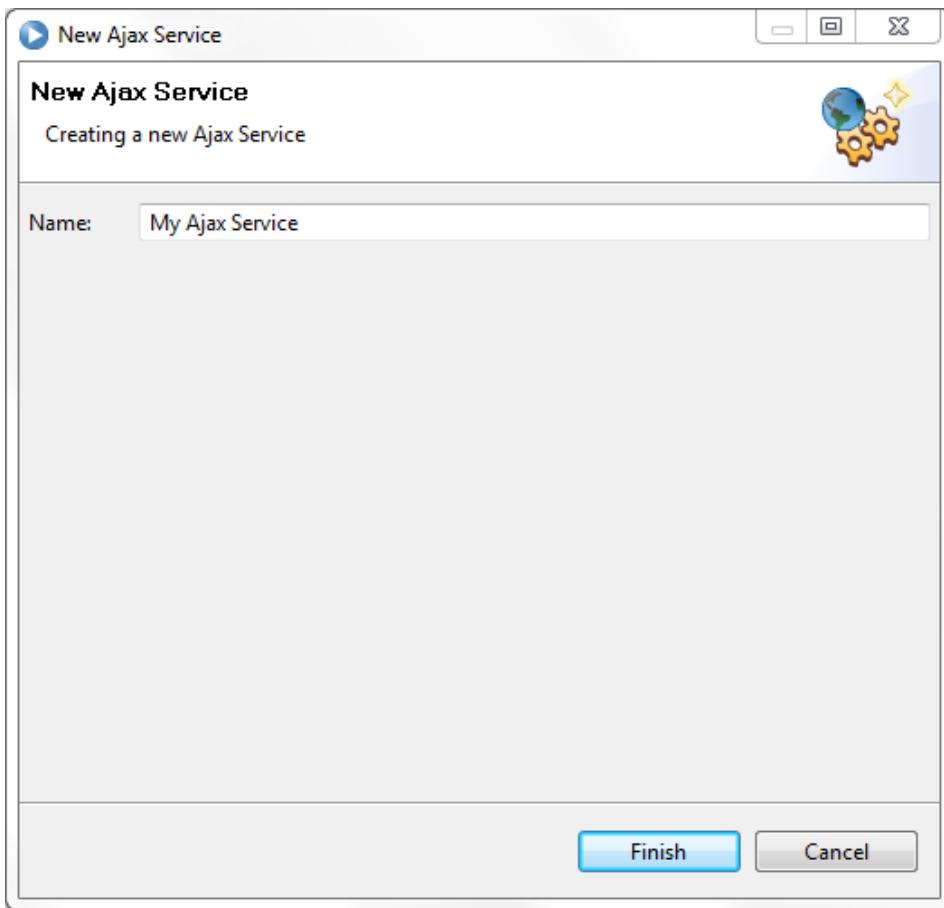
Ajax (which is an acronym for Asynchronous JavaScript and XML) is a description for the practice of constructing web pages that have dynamic content with call-outs to external services to obtain additional data. This technique started when the web browsers provided a function called XMLHttpRequest which allowed the browser to make an HTTP request to obtain data while the page was already loaded but without the page as a whole having to be refreshed.

An Ajax service is used in conjunction with Human Services and coaches. It provides the ability to source data dynamically for the coach by running a IBPM service on the server with the data returned from the server used within the coach.

An Ajax service is defined from the User Interface category within the Library.



When created, it is given a name.



At this point, we must not lose sight of how this service is going to be called. It will be called when a component on the form is selected and changed. When the change occurs, the Ajax service will be called and data will be returned. The returned data will be used to populate the content of a **different** component. If we think in terms of inputs and outputs to this service, we find that it will have a single input which is the value of the component that triggered the invocation of this service. The new value of that component is passed in as an input variable to the Ajax service. The implementation in IBPM **requires** that the name of the input variable be fixed and be called `inputVar`. You **must** define an input variable of type String with this name.

The screenshot shows a tree view of variables:

- Variables
 - Local
 - Input
 - inputVar (String)
 - Output
 - outputVar (NameValuePair) (List)
 - name (String)
 - value (String)
 - listSelected (NameValuePair)
 - name (String)
 - value (String)
 - Private
 - Exposed Process Variables
 - Localization Resources

The data that is returned is dependent on what kind of control the response data from the Ajax

service is associated with. For here, let us assume it is a Single Selection list. A Single Selection list is defined by a list of `NameValuePair` data structures. The `NameValuePair` is a IBPM predefined data structure that contains two fields. One called `name` and the other called `value`. When a list of these structures is associated with a Single Selection list control, each element in the list data contributes one row to the visual list control. The `name` field is used as the visual in the row.

Let us examine this by example. Imagine our business is split into two regions. The East and the West. When we wish to ship product, we ship from the East region or the West region. In each region, we have the choice of possible shipping warehouses from which to ship. If we want to present a Coach to select a warehouse, we might want to be smart and allow the user to first select a region and once the region is selected, then select a warehouse. The allowable warehouse values should be displayed based on the region selected. This is illustrated in the following diagrams.

Lombardi Coach

Shipping Location

Region: East

Ship from:

- Boston
- Boston
- New York
- Miami

Lombardi Coach

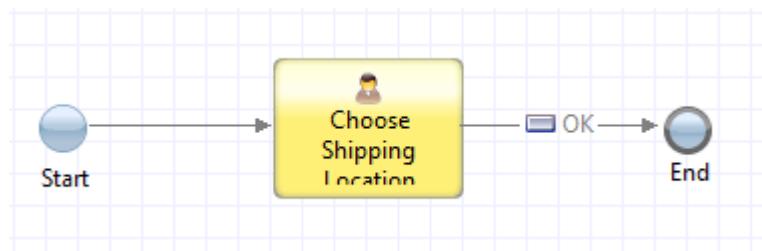
Shipping Location

Region: West

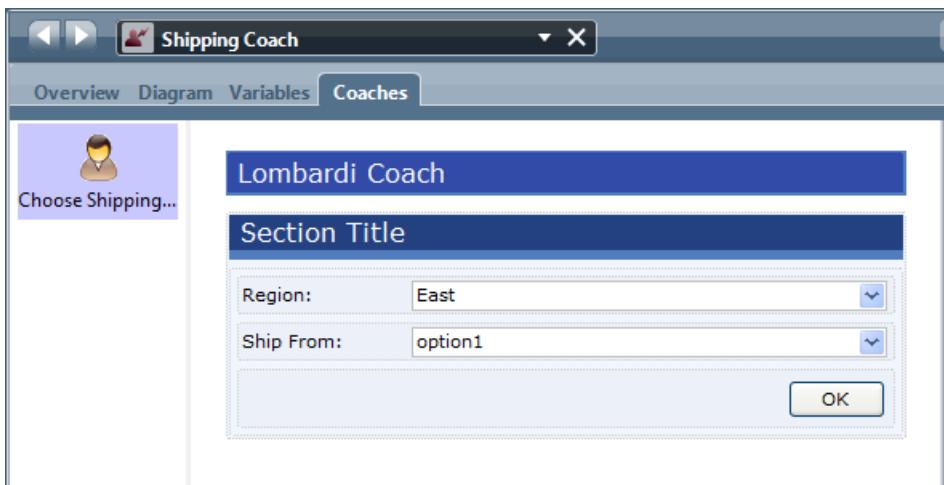
Ship from:

- Seattle
- Seattle
- San Francisco
- Phoenix

To achieve this, we might build a Coach that looks as follows:



With the body of the Coach looking like:



<Needs updated for IBPM>

The Shipping Location section contains two Single Select controls. The first which has the label "Region:" contains static choices:

Value	Display Text
East	East
West	West

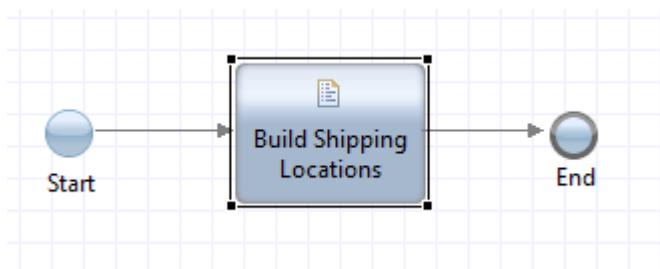
The second control is the more interesting one. Looking at its definition we see that it invokes an Ajax defined service.

Value	Display Text
-------	--------------

Notice three things:

1. It has the radio button selected for an Ajax service
2. It names the **other** control as the trigger that will invoke the service
3. It names an Ajax service that will be called to return the data that will be used to populate **this** control.

Now let us look at the definition of the Ajax service. As a high level diagram, it contains only one Server Script activity:



The variables defined in the service look as follows:

The screenshot shows the 'Build Shipping Locations' service configuration. The 'Variables' tab is selected. The 'Variables' tree on the left lists:

- Local
 - inputVar (String)
- Output
 - result (NameValuePair) (List)
- Private

To the right of the tree are several buttons:
Add Private
Add Input
Add Output
Link Epv
Link Localization
Remove
Move Up
Move Down

Note that there is an input variable that **must** be called `inputVar`. The output variable has no restriction on its name but `result` is a suggested choice. The data type of the output variable is a List of `NameValuePair` data types.

Finally, there is the JavaScript logic contained in the Server Script:

```
tw.local.result = new tw.object.listOf.NameValuePair();
if (tw.local.inputVar == "East")
{
    tw.local.result[0] = new tw.object.NameValuePair();
    tw.local.result[0].name = "Boston";
    tw.local.result[0].value = "Boston";

    tw.local.result[1] = new tw.object.NameValuePair();
    tw.local.result[1].name = "New York";
    tw.local.result[1].value = "New York";

    tw.local.result[2] = new tw.object.NameValuePair();
    tw.local.result[2].name = "Miami";
    tw.local.result[2].value = "Miami";
}
```

```

if (tw.local.inputVar == "West")
{
    tw.local.result[0] = new tw.object.NameValuePair();
    tw.local.result[0].name = "Seattle";
    tw.local.result[0].value = "Seattle";

    tw.local.result[1] = new tw.object.NameValuePair();
    tw.local.result[1].name = "San Francisco";
    tw.local.result[1].value = "San Francisco";

    tw.local.result[2] = new tw.object.NameValuePair();
    tw.local.result[2].name = "Phoenix";
    tw.local.result[2].value = "Phoenix";
}

```

This script won't be explained in detail other than to say that depending on the value of the input data which is the value of the region selection control, one of two different lists of data are built and returned.

Not all components support Ajax service binding, the ones that do are:

- Text Input – The Ajax service should return an array of strings
- Text Output – The Ajax service should return an array of strings
- Combo Box (Single Select – Drop Down List) – The Ajax service should return an array of NameValuePairs
- Table – The Ajax service should return ...?

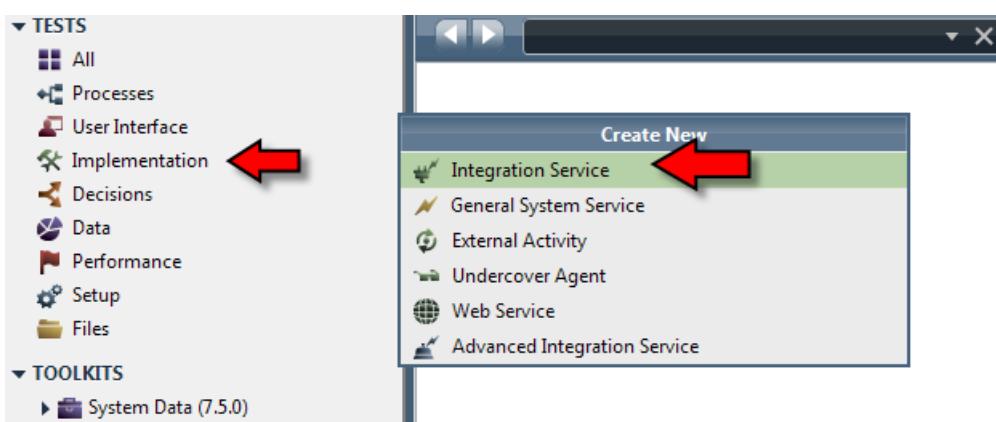
In practice, the data that is returned from the Ajax service call may be dynamically sourced such as through a DB lookup.

See also:

- Error: Reference source not found
- Video - [IBM BPM 7.5 - Calling an Ajax service from a Coach](#) - 2011-06-27

Integration Service

The integration service allows a process to invoke an external entity such as a Web Service or a piece of Java Code. It is the integration service which provides the one of the components for interacting with logic outside of the BPMN environment including true SOA external service.

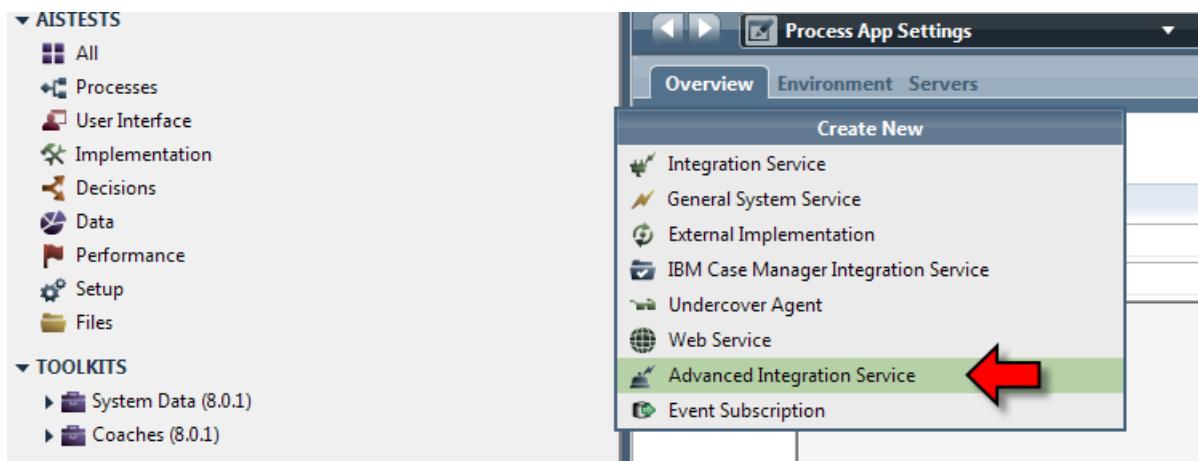


See also:

- Web Service Integration

Advanced Integration Service

The Advanced Integration Service gives us the ability to define a service that is implemented by an IBM BPM Advanced SCA Module.

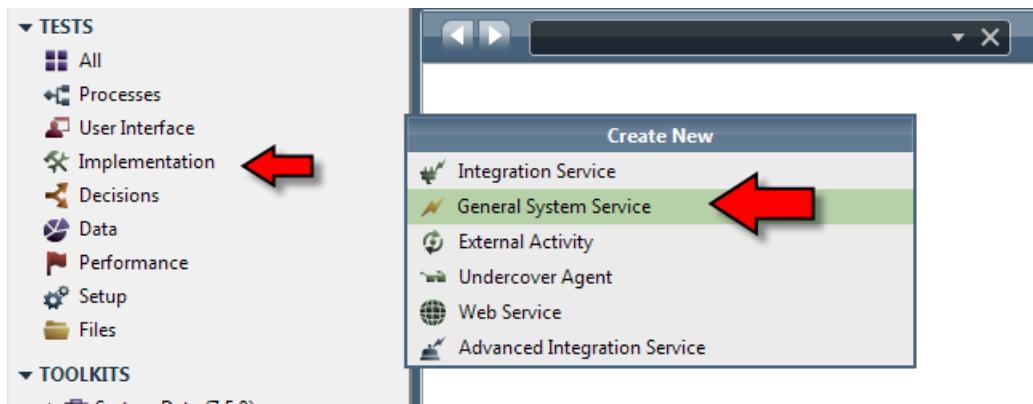


See also:

- Inter-operating between a BPMN process and an SCA module
- DeveloperWorks - [Linking business processes and enterprise services together using IBM Business Process Manager Advanced](#) - 2012-09-26

General System Service

A general system service is used when the BPD needs to work with the data in the process but does not need to invoke either a Web Service or Java Code. The General System Service is the simplest of the service implementation environments.



Decision Service

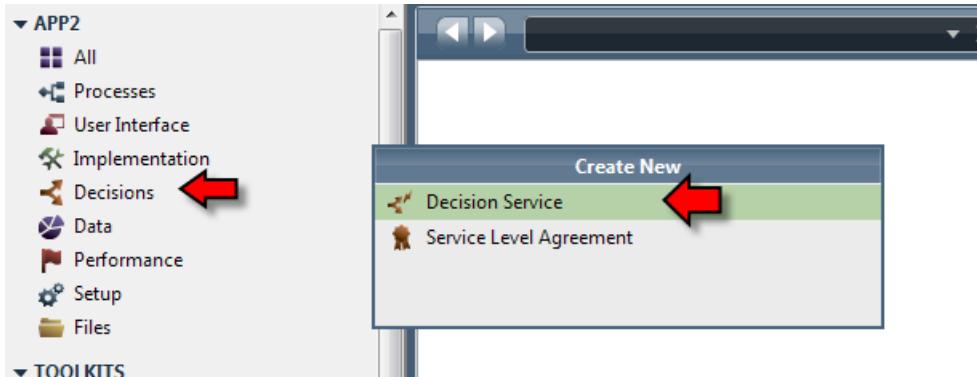
During the execution of a process, we may wish to have *rules* executed. Commonly, rules are decision points in a process where the values of variables in the process are used to determine what to do next. An illustrative example would be to determine the discount to apply for volume orders.

We may have the following notion:

Order Value	Discount
\$0-\$999	0.00%
\$1000-\$1999	5.00%
>\$2000	10.00%

Looking at this table we see that the discount amount is a function of the order value and hence there is a decision to be made on the discount based on the value of the order. This is the type of function that is ideally suited to Decision Services.

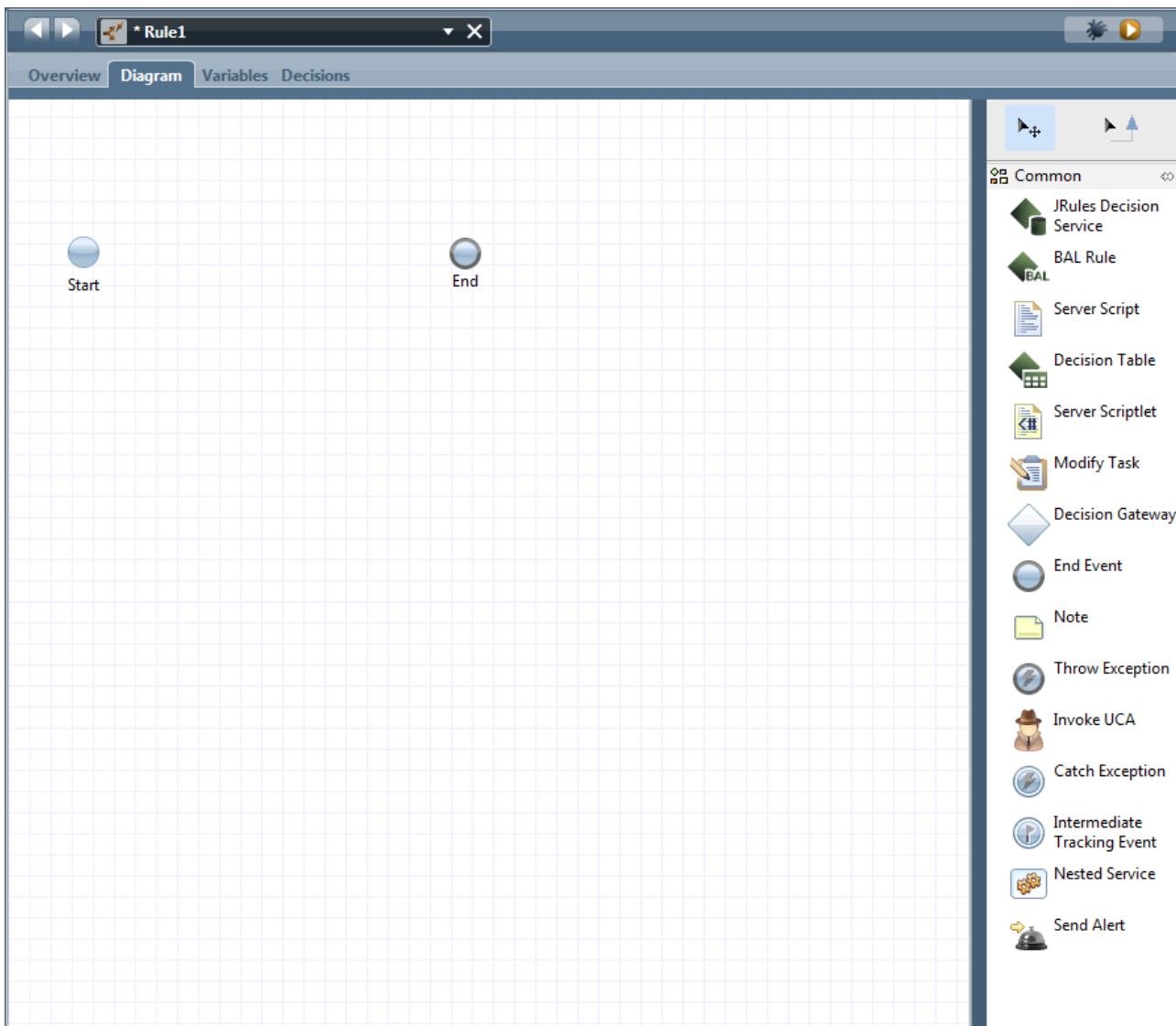
A Decision Service is added to the Library by selecting decisions and adding a Decision Service.



When selected, a new dialog appears into which the decision service name can be entered:



Upon completion of the decision service creation, a panel is shown allowing us to specify the details of the decision:

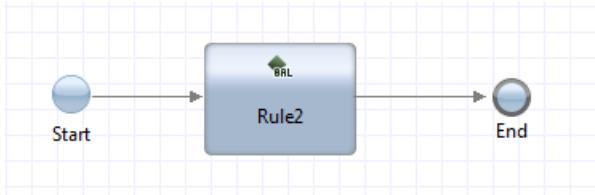


Similar to other service descriptions, we now have the ability to create a service implementation using the building blocks of service creation. For a decision service, we have three items that are specific to this type of service. These are:

- JRules Decision Service – This decision implementation leverages an external instance of the IBM JRules product which is a full function BRMS.
- BAL Rule – This decision implementation type uses the Business Action Language (BAL) as found in the JRules product. No other product is required to use this capability.
- Decision Table – This decision implementation type is the same as found in previous releases of the Lombardi product.

Business Action Language (BAL) Rule

The Business Action language (BAL) Rule mechanism allows us to define a series of if/then/else conditions based on the current values of the variables in the process. When a BAL rule is added to the service canvas it looks as follows:



Switching to the Decisions tab allows us the ability to enter the details of the rule using BAL.



The structure of a business rule is generically:

- definitions (optional)
- if
- then
- else (optional)

In the definitions section we can define local variables/values that are used in the remainder of the rules. For example:

```
definitions
    set low_discount to 5;
```

The full language of BAL is described as part of the JRules product and can be found in the InfoCenter article called [Business Action Language \(BAL\)](#).

Decision Table

IBPM has a concept it calls Decision Tables. To understand what this is about, consider the following concept in a programming language:

```
if (expression A)
{
    some code;
} else if (expression B)
{
    some code;
} else
{
    some code;
}
```

What this describes is a set of expressions and associated code. When the first expression reached evaluates to true, the associated code is executed. In summary, that is what TeamWorks rules describe. The rules allow us to build expressions based on the values of a set of variables that are in scope. A table is visually constructed. The columns in the table represent the variables and the

rows represent possible values for those variables. The rows are then evaluated at runtime from top to bottom and if the overall expression evaluates to true, the associated JavaScript code is executed. Commonly that JavaScript code will assign a value to another variable.

When a Decision Table is added to the canvas, it looks as follows.

The diagram illustrates a 'Rule' node on a canvas, connected by arrows to other nodes. Below the canvas is a screenshot of a software interface titled 'Decisions'. On the left, a sidebar shows a 'Rule' icon. The main area contains a 'Condition (IF)' table and an 'Action (THEN)' section.

Condition (IF) Table:

	orderValue	orderValue	Action Requirement
1	-	< 1000	Discount is 0%
2	≥ 1000	< 2000	Discount is 5%
3	> 2000	-	Discount is 10%
4			

Action (THEN) Section:

Requirement: Discount is 5%

```
1tw.local.discount = 5;
```

Action: `if`

Pattern matching

A special pattern of "-" (just the dash character) can be used to match any pattern. It is a wild card.

Strings

"string value"	Matches the exact string value
{"string value 1", "string value 2"}	Matches either of the string values
!{"string value 1", "string value 2"}	Matches anything except the supplied string values

Numerics

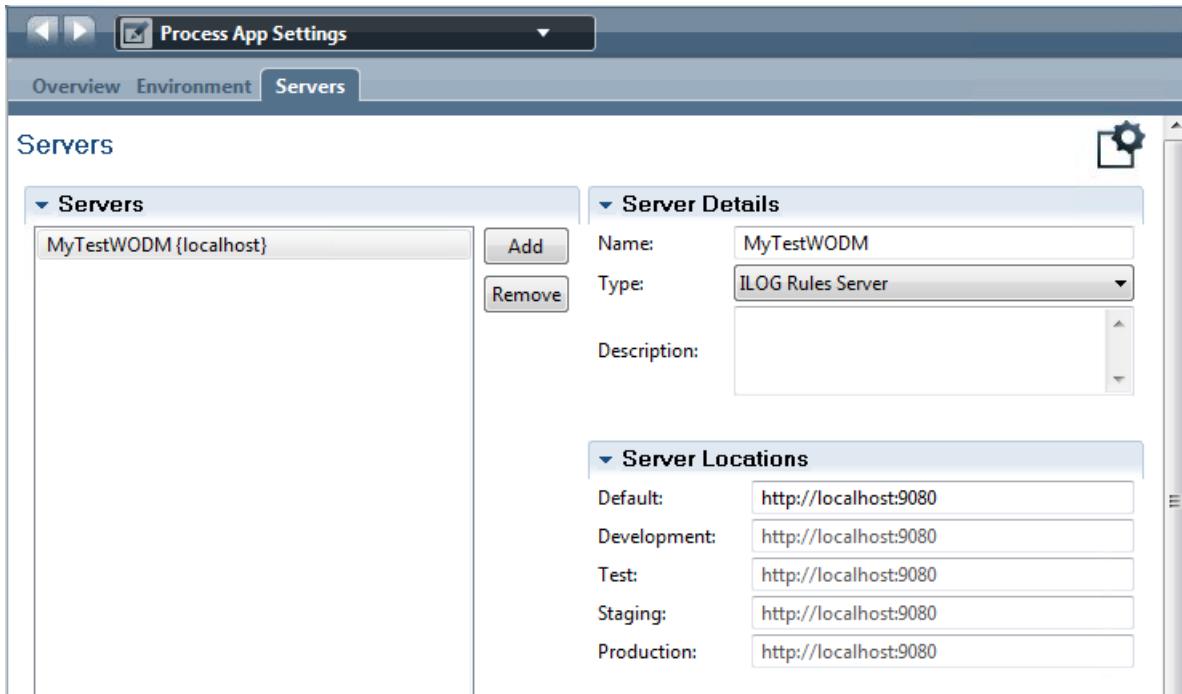
number	Matches exactly the number.
numberLow..numberHigh	Matches anything between numberLow and numberHigh (inclusive)
> number	Matches anything greater than number
< number	Matches anything less than the number
≥ number	Matches anything greater or equal to the number
≤ number	Matches anything less than or equal to the number
{number1, number2}	Matches either number1 or number2
!{number1, number2}	Matches anything other than number1 or number2

Booleans

TRUE	Matches the boolean true
FALSE	Matches the boolean false

Integrating with WODM

In addition to the previous options, we can also integrate with an instance of WODM. The first task is to define the location of the Rule Execution Server to the Process App. In the Process App Settings we can define a new server of type "ILOG Rules Server" and define the hostname and port on which it can be found:



By dragging a component called "JRules Decision Service" into the canvas area we can connect to WODM.

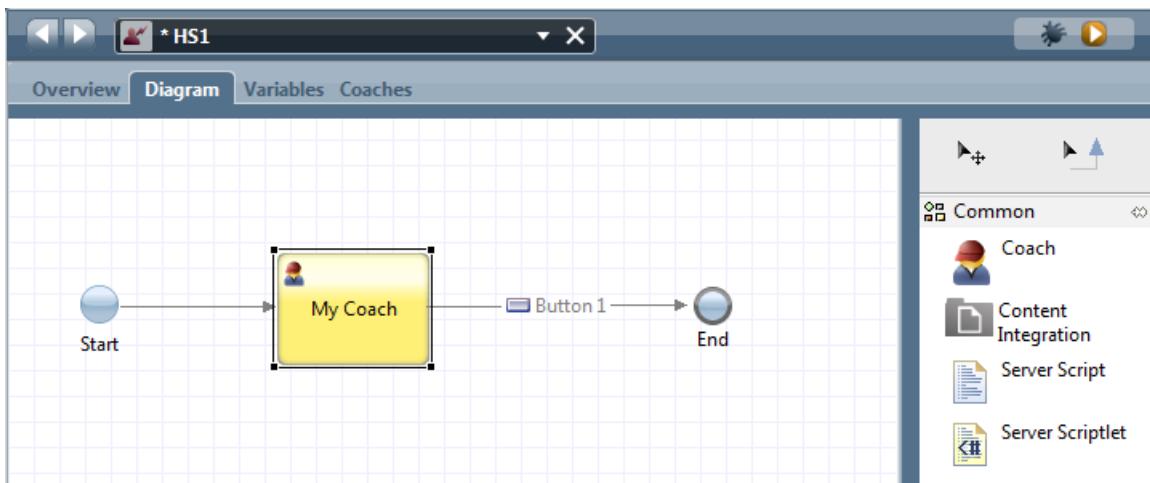
See also:

- [WebSphere Operational Decision Management - WODM](#)

Service Components

Definitions of services exist as part of a Process Application. Each definition for a service has a name and associated "coding". The service is "associated" with a BPD activity in a BPD process diagram. The service can either be created (from new) from the BPD activity or an existing service can be "linked" to that activity. The definition of the service is an "implementation" part of the solution.

Similar to a BPD, a Service has its own visual editor and palette:



The palette is composed of the following parts some of which are only applicable to certain types of service:

		Integration Service	Human Service	Rule Service	Other Service Types
	Coach		•		
	Heritage Coach		•		
	Server Script	•	•	•	•
	Server Scriptlet	•	•	•	•
	Modify Task		•		
	Postpone Task		•		
	Content Integration	•	•		
	Decision Gateway	•	•	•	•
	End Event	•	•	•	•
	Note	•	•	•	•
	Error End Event	•	•	•	•
	Invoke UCA	•	•	•	•
	Error Intermediate Event	•	•	•	•
	Intermediate Tacking Event	•	•	•	•
	Stay On Page Event		•		
	Nested Service	•	•	•	•
	Send Alert	•	•	•	•
	Web Service Integration	•			

	Java Integration	•			
	JRules Decision Service			•	
	BAL Rule			•	
	Decision Table			•	

After dragging and dropping the elements onto the diagram, the elements can be visually wired together with Sequence Lines in the same fashion that BPD elements are also wired together with Sequence Lines. The remainder of this section will walk us through the different types of service components available to be included in the implementation of a service.

See also:

- Sequence Lines

Coaches

When added to a Human Service diagram, the Coach looks as follows:



From the palette, the icon is:



The idea of a Coach is that it describes a screen layout that is to be shown to a user. The screen will contain output data that is populated and shown to the user as well as input data that can be entered by the user and returned to the service. Coaches are such an important area of IBPM that they have their own section in the book found here:

- User Interfaces and User Interaction

Server Scripts

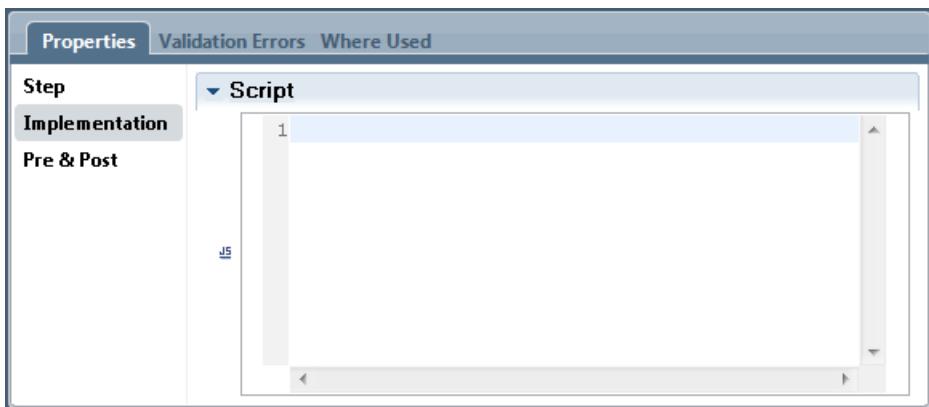
When added to a Service diagram, the Server Script step looks as follows:



On the palette, the icon representing this step looks like:



The purpose of this type of step is to allow the developer to enter a JavaScript fragment of code to be executed at runtime within the Process Server. The properties section of the Step provides a code entry area where the script can be coded:

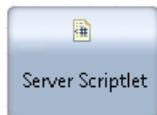


Full entry assist is available to help in building the script.

Server Scriptlet

The purpose of this element is to assign text to a variable but provides an easy entry text input area into which the text can be entered.

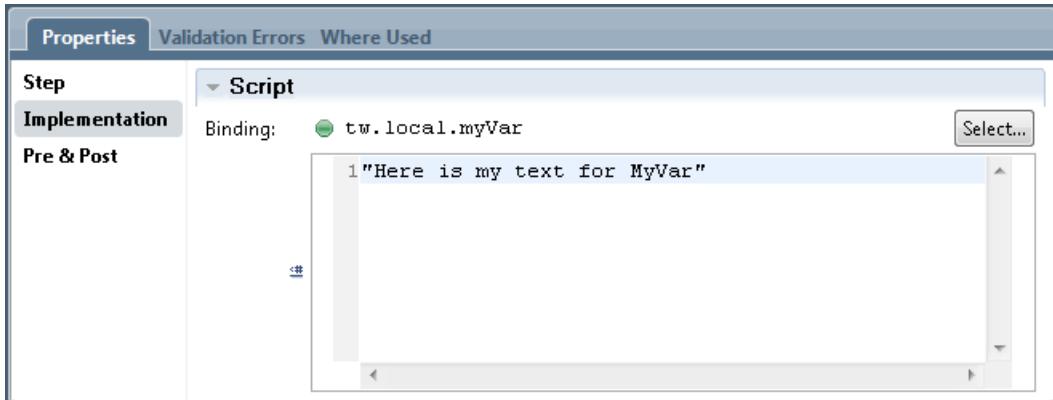
When added to the canvas, the Server Scriptlet step looks as follows:



From the palette of steps, it looks like:



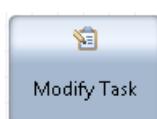
The implementation tab of this Step looks as follows:



Be VERY careful here ... adding quotes in the text adds quotes to the content. I have been bitten by this a number of times.

Modify Task

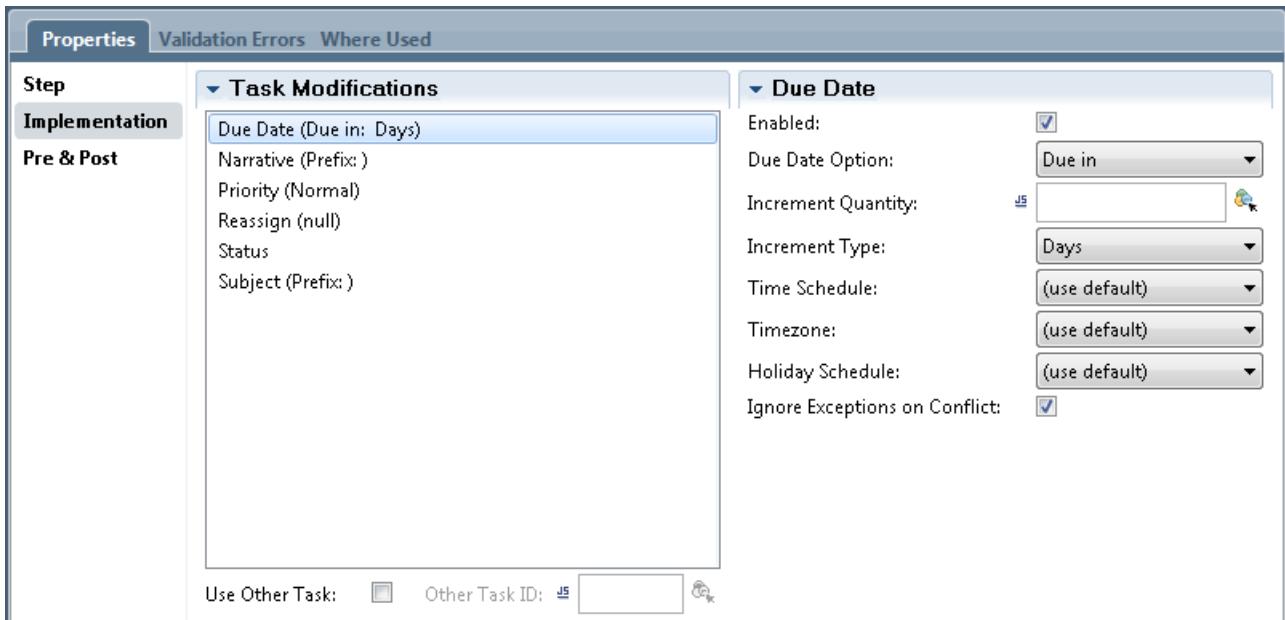
The Modify Task component allows us to dynamically alter a number of the attributes of the current task within a Human Service. When added to the canvas, the Modify Task element looks as follows:



From the palette, it shows as:

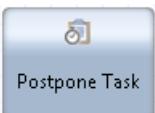


This element can change the characteristics of a Task under logic control.

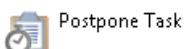


Postpone Task

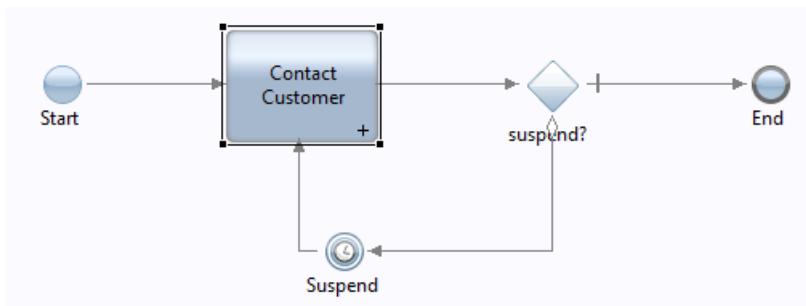
The postpone element allows a task to suspend itself for later processing. What is being postponed here is the completion of the task. The state of the service is saved for later retrieval. When inserted into a Service Editor diagram, the element looks as follows:



From the palette area, it looks like:



When the Postpone is reached, the task is returned back to the in-box where it can be claimed again. If we want to hide the task for a period of time, we have to perform some additional work. One solution is to check at the conclusion of the Coach whether or not we suspended it and, if we did, pause the re-creation of the task for a period of time.

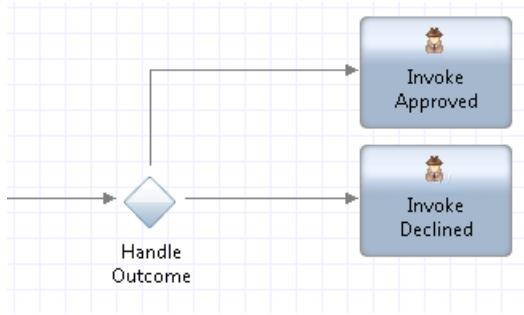


See also:

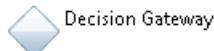
- Vimeo: [Suspending a task for a period of time](#)

Decision Gateway

The decision gateway is used to route the execution path of the service based on the evaluation of an expression. The following shows a Decision Gateway in the diagram.



From the palette, this component looks like:



The implementation settings for the component has the following:



There will be a line for each possible path out from the component. One of the lines will be the default path that will be followed if none of the other expressions evaluate to true.

The labels for each line are of the form:

link name (Step Name)

It is recommended to provide a label for each of the outgoing paths from the Decision Gateway otherwise they will show up as Untitled.

End Event

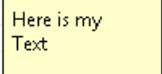
The end event is used to mark the end of the service. On the diagram it looks as follows:



It has the same icon in the palette.

Note

A note can be added into the diagram that provides an aid to a subsequent reader when trying to understand the solution. Text is added in a yellow box that can be positioned in the diagram. An example of how this looks is shown following:



From the palette, the note has the following icon:



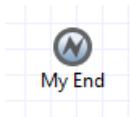
The properties of the note contains a text input area into which the text for the note can be entered:



Having a note in the diagram does not effect the operation of the solution its purpose is solely for readability.

Error End Event

When added to the canvas, the Error End Event looks as follows:



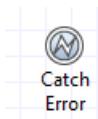
The purpose of this activity is to explicitly throw an error. This would be used when, during the processing of a service, it was detected that something problematic has occurred. For example, a request to purchase a product and the expiration of a credit card has passed or the product is not in stock.

Included in the Properties of this type of element is an Implementation section:



Error Intermediate Event

When added to the canvas, the Error Intermediate Event looks as follows:



Included in the Properties of this type of element is an Implementation section:



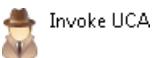
Invoke UCA

Within a Service, we may wish to invoke a UCA. Invoking a UCA can cause the creation of a new instance of a process. Remember that UCAs are *usually* associated with an external service or timer causing them to be invoked. Giving us the ability to initiate a UCA explicitly adds a lot more flexibility to solutions we may build. A primitive is supplied that allows us to explicitly invoke a named UCA.

When added to the canvas, the UCA caller primitive looks as follows:



It is added by dragging a palette entry that looks like:



Once added, its implementation properties look as follows:



The UCA caller primitive's implementation names the UCA that is to be called. The Data Mapping section allows us to map the variables in scope in the process to the parameters expected by the UCA.

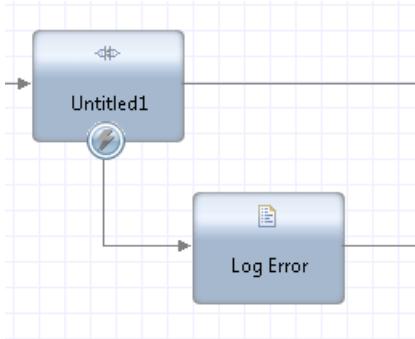
See also:

- Undercover Agents (UCAs)

Catch Exception

If an activity in a service can fail then it is wise to try and handle that failure by taking some remedial action even if it is only alerting an administrator. A Catch Exception handler can be added to activities that can generate failures. To add the handler, drag it from the palette and drop it onto the activity.

Here is an example with a Catch Exception added:



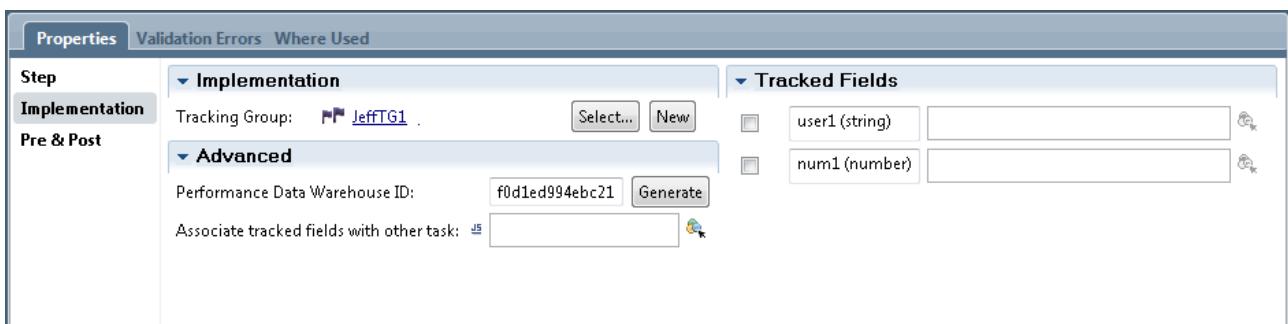
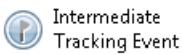
In the palette area, the Catch Exception looks like:



The nature of the exception can be found in the XMLElement data contained in the variable `tw.system.error`.

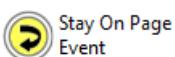
Intermediate Tracking Event

A service can include an Intermediate Tracking Event step. When reached, this will externalize a row into the associated Tracking Group view in the Performance Data Warehouse database.



Stay On Page Event

From the palette, this entry looks as follows



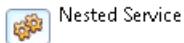
Nested Service

The Nested Service step allows us to invoke another service from within the current service. By using this feature we can modularize our services into reusable pieces. We can build an aggregate service from a variety of other services.

When added to the canvas, the Nested Service step looks as follows:



From the palette of available steps, it looks like:

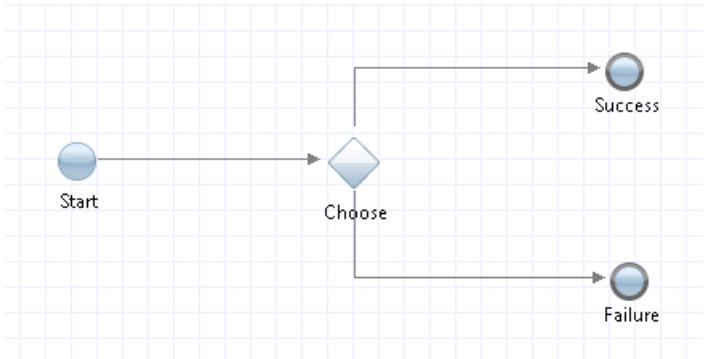


The implementation details of the Nested Service look like:

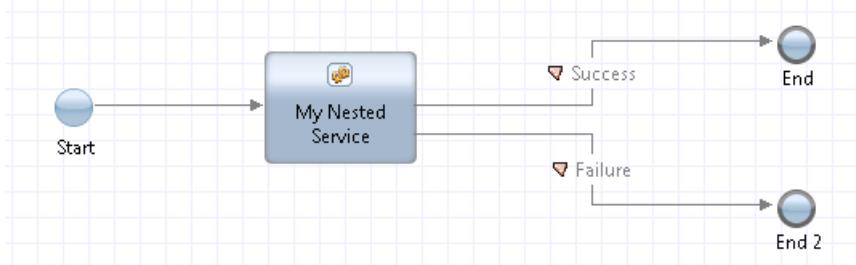


A screenshot of a software interface showing the properties of a step. The tab 'Implementation' is selected. Under 'Attached Service', it shows 'Attached Nested Service: Send Alert'. There are 'Select...' and 'New' buttons.

If the service being called has multiple end event elements, then each of these can contribute their own outgoing link. In the following, we have a simple service with two possible endings:



If we invoke this service in another service through a Nested Service call, we can then have links associated with each of the possible endings (in this case, an ending called Success and an ending called Failure).



Notice that the links are labeled with the names of the end terminals in the called service.

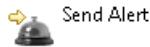
The service to be called for a nested service can be explicitly selected with the "Select" button or, alternatively, a service dragged from the library of services and dropped on the service diagram will automatically add a bound Nested Service node.

Send Alert

When added to the canvas, the Send Alert step looks as follows:

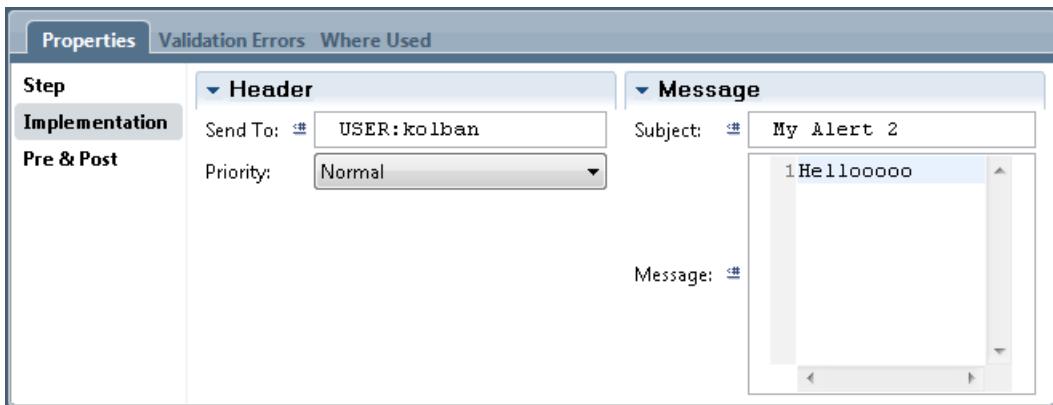


From the palette of available steps, it looks like:



When called, this activity sends an Alert message to the named user or group. This message can be found in the Process Portal.

The implementation section of the step allows us to name the recipient, subject, message and priority.



The Send To field names the recipient which can be either of:

USER: <user name>

ROLE: <group name>

The Alert shows up as an entry in the Process Portal Collaboration section as an entry with a "service bell" beside it.

Collaboration				
Type	Date ▲	From	To	Comment
bell	Oct 17, 2011 4:44:52 PM	Administrator	Administrator	Hello
document	Oct 17, 2011 4:47:46 PM	Administrator	Everyone	eeee

See also:

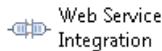
- Error: Reference source not found

Web Service Integration

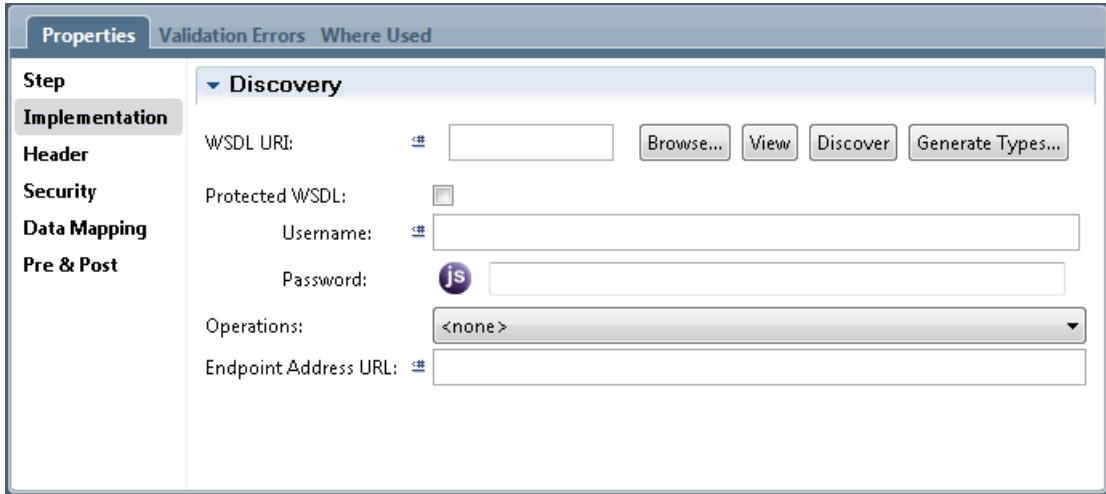
A Web Service integration component allows the service to invoke an external Web Service provider. When added to the canvas, a Web Service integration looks as follows:



It is added by dragging a palette entry that looks like:

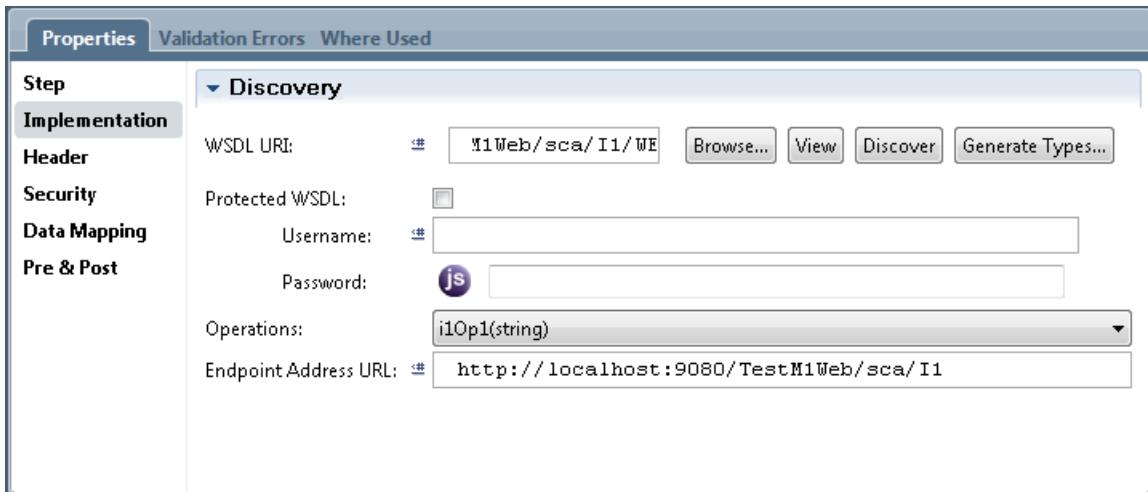


The implementation property page for this service looks like:

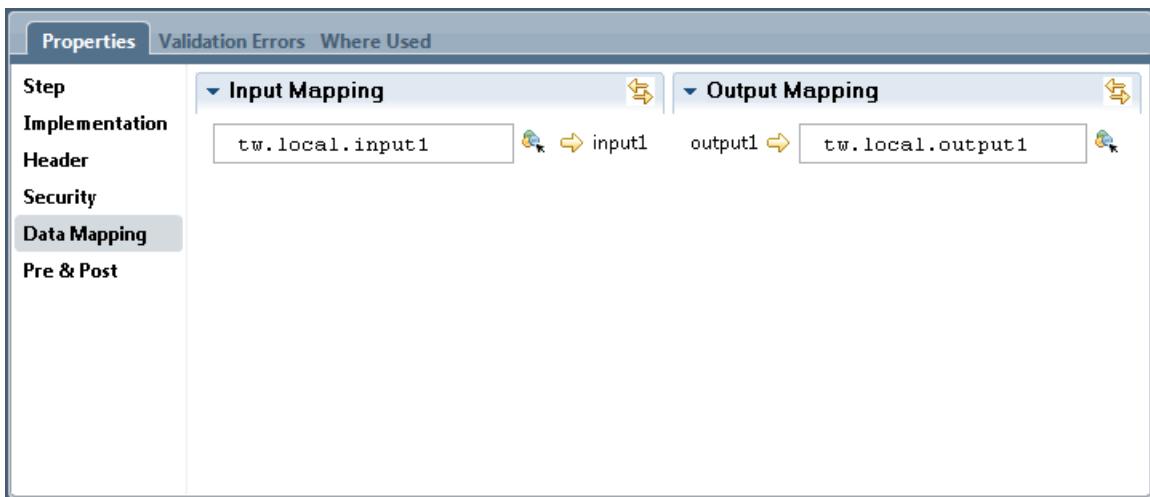


In the WSDL URI : text field, the URL for a WSDL file should be entered. Clicking the Discover button will then cause IBPM PD to retrieve the WSDL pointed to by this URL and parse its content. Once the content has been parsed, IBPM PD will know the potential operations that can be invoked against that service. The Operations pull-down can then be used to select the desired operation.

Here is an example of having entered a WSDL URI, Discovered its operations and having selected an operation:



When a Web Service is called, it commonly expects input data to be provided. On return from the Web Service call, response data may be available. Data in IBPM is held in variables that are local to the IBPM Service and hence we have to map from IBPM Service variables to the input of the Web Service and map from returned Web Service data back to IBPM Service variables. This can be achieved on the Data Mapping tab of the panel.



See also:

- Outbound Web Services

Java Integration

Integration with Java is discussed in a separate section.

See:

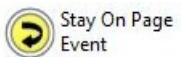
- Java Integration

Stay on Page Event

When added to a Human Service diagram, the node looks as follows:



From the palette, the corresponding icon is:



This activity can only be added to Human Services. Its purpose is that, when reached, the last shown Coach is returned to. Think of it like an implicit wire back to the last Coach shown.

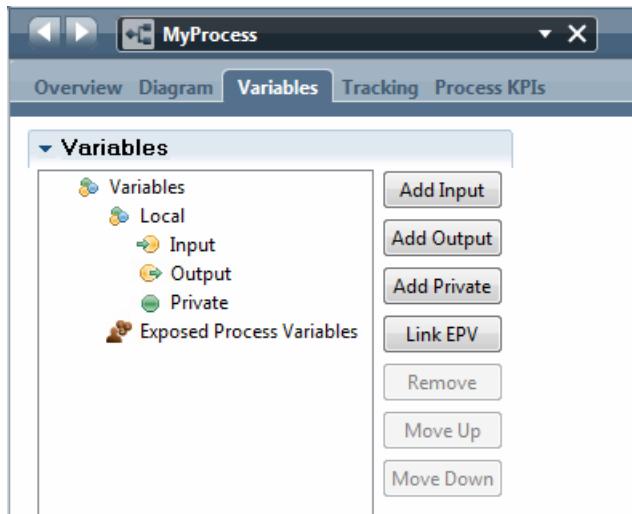
Variables – Process and Service

As an instance of a process executes, that process is going to have data maintained as part of its state during its complete life. The data may be input into the process when it starts or may be calculated or entered as the process progresses through its life.

Variables are used to hold the state data associated with an instance of a process. Variables are "named" entities which have associated data types. Variables occur in two distinct categories of places in IBPM. The first is the set of variables which can be defined in a BPD. These variables have a scope of the entire instance of the process. Variables are also constrained by the process in which they live. Variables in one process instance can't be accessed in another process (without an explicit passing of their values). A change to a variable in one activity will be seen in subsequent activities.

The other place variables can be defined is within a Service implementation. These variables exist for the life of the Service and are no longer available after the service completes.

Variables can be defined with a qualifier of *input*, *output* or *private*. Input variables have their values passed into a process or service when that process or service is initiated. Output variables have their values returned from a process or service when it completes and private variables only have scope within a process or service and are deleted when they end.



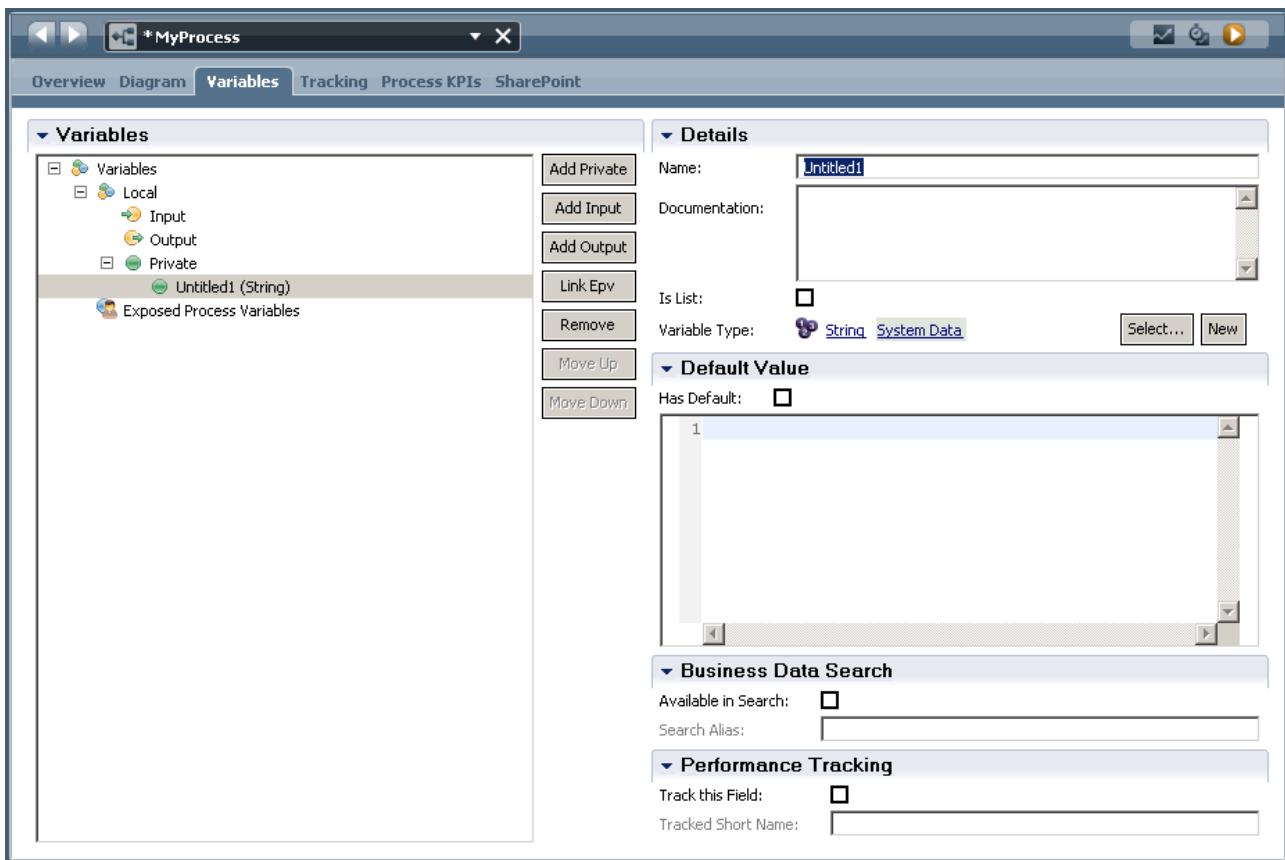
When a variable is created, it can be addressed from within JavaScript with the prefix:

```
tw.local.<variable name>
```

(Note: This prefix `tw.local` is used so frequently that the JavaScript editor has a special shortcut for that called "`twl`". If you enter `twl` and then CTRL+Space for content assist, the `twl` entered characters are changed to `tw.local`.)

When a variable is defined, it is given a name and other attributes including:

- Data type – The type of data represented by the variable
- Default value – An optional default value
- Is List – A flag indicating that the variable represents a list/array of instances of the data type
- Documentation – Human readable documentation on the purpose of the variable



The data types initially supplied available include:

- ANY
- Boolean
- Date
- Decimal
- Integer
- Map
- NameValuePair
- String

These data types come from the System Data toolkit which is always present within the environment.

When a new variable is defined, it can be declared as having a default value. What this means is that an instance will be created and will already have a value without having to explicitly perform a subsequent assignment to it.

System of Record data vs Business Intelligence data

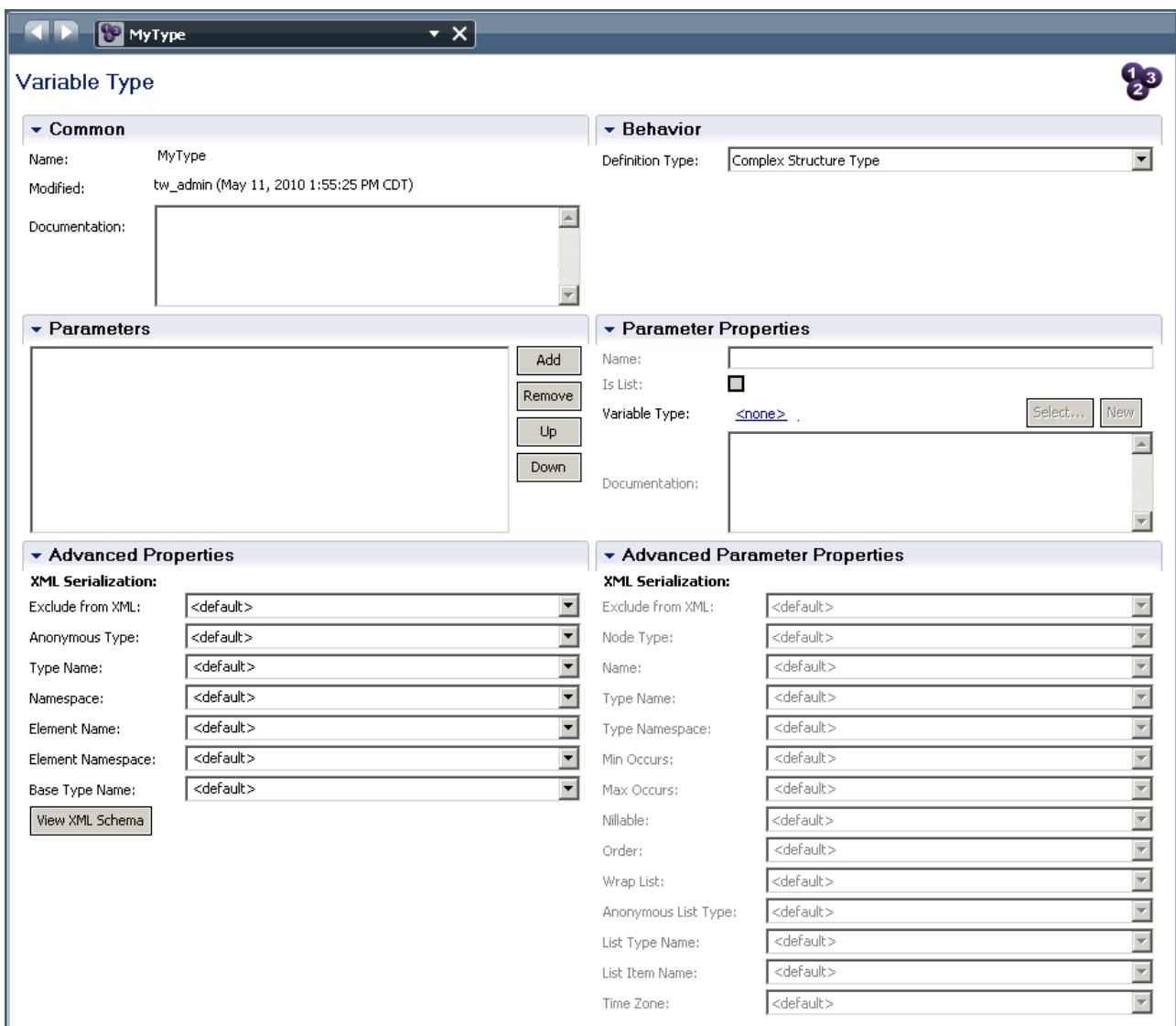
Let us take a few minutes to think about the data used and produced by a process instance. In my opinion, this data can be split into two logically distinct categories. These are "System of Record data" (SOR) and "Business Intelligence data" (BI).

SOR data is information that can not afford to be lost once the process has ended. It is typically used by other applications in the future. As an example, imagine a process which books a seat on an airplane for a future trip I plan to make. It would be very wrong if I turned up at the airport three weeks from now and found that there was no record of my booking. BPM is **not** a repository for SOR data. Instead, BPM could be responsible for causing new SOR data to be created or existing SOR data to be updated. The SOR data is maintained by systems outside BPM such as a database or another application.

BI data is information that could, in principle, be lost without damaging the operation of your business. This data is used to examine how your processes performed in the past. It includes times, paths, people, decisions and data relevant to those decisions. Its primary purpose is to provide the capability to perform business analytics. If it were lost, no customers would be upset, no monies would be lost, no corrective actions need be performed ... the worst that would happen is that you would be unable to ask questions on how your processes performed in the past. IBM BPM can record BI data through the Performance Data Warehouse (PDW) architecture.

Creating new Data structures

The data types supplied with IBPM are important and useful but as we start to model our business processes we will commonly find that we want to create new data structures. New data structures can also be created which can be composed of fields (also called parameters). These new data structures will be used to represent different types of logical data such as "A Customer" or "A Loan".



In the Behavior section of the data type editor, there are two choices:

- Complex Structure Type
- Simple Type

When a complex type variable is created, it must be initialized before it can be used. This can be done through JavaScript with the code:

```
tw.local.myVariable = new tw.object.ComplexType();
```

A similar story is also true for data of type list:

```
tw.local myListOfStrings = new tw.object.listOf.String();
```

Simple Types

The Simple Type allows us to place constraints on certain simple data types.

Simple type – String

Here we can place length restrictions on the string. This includes either a fixed length or a minimum/maximum length.

Simple Type		String Validation	
Type:	String	Length Restriction:	None (Unlimited Length)
Error Message:	<input type="text"/> <input style="float: right;" type="button" value="..."/>		
	<input style="float: left; margin-right: 10px;" type="button" value="Add"/> <input style="float: left; margin-right: 10px;" type="button" value="Delete"/>		

Simple type – Integer

Simple Type	
Type:	<input type="text" value="Integer"/>
Error Message:	<input type="text"/>
Integer Validation	
Minimum:	<input type="text"/> <input checked="" type="checkbox"/> Inclusive <input type="checkbox"/> Exclusive
Maximum:	<input type="text"/> <input checked="" type="checkbox"/> Inclusive <input type="checkbox"/> Exclusive
Precision:	<input type="text"/> <input checked="" type="checkbox"/> Inclusive <input type="checkbox"/> Exclusive
Regular Expression:	<input type="text"/>

Simple type – Decimal

Simple Type	
Type:	<input type="text" value="Decimal"/> <input type="button" value="▼"/>
Error Message:	<input type="text"/>
Decimal Validation	
Minimum:	<input type="checkbox"/> <input type="text"/> <input checked="" type="radio"/> Inclusive <input type="radio"/> Exclusive
Maximum:	<input type="checkbox"/> <input type="text"/> <input checked="" type="radio"/> Inclusive <input type="radio"/> Exclusive
Precision:	<input checked="" type="checkbox"/> <input type="text"/>
Scale:	<input checked="" type="checkbox"/> <input type="text"/>
Regular Expression:	<input type="checkbox"/> <input type="text"/>

Simple type – Date

Simple Type	Date Validation
Type: <input type="text" value="Date"/>	Format: <input type="text" value="d-M-yyyy"/>
Error Message:	<input type="text"/>

Simple type – Time

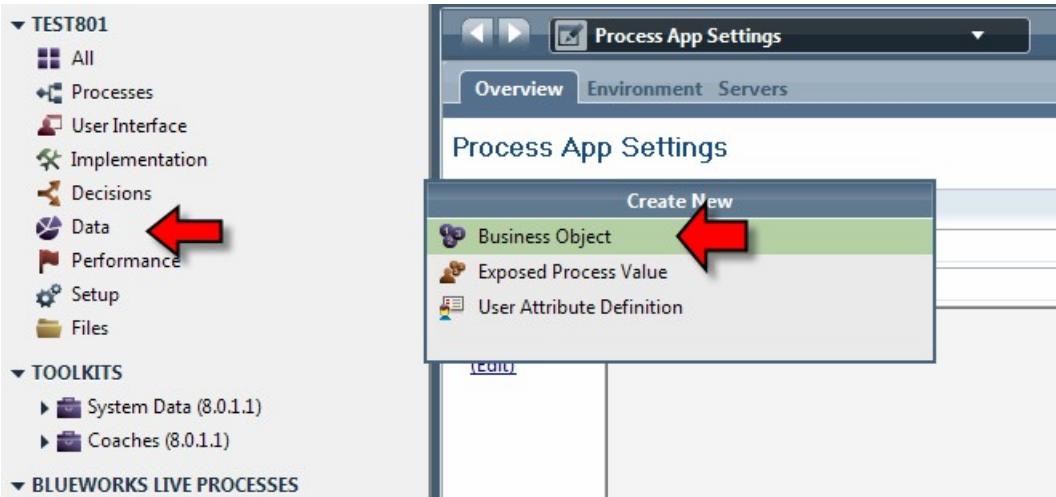
Simple Type	Time Validation
Type: <input type="text" value="Time"/>	Format: <input type="text" value="d-M-yyyy"/>
Error Message:	<input type="text"/>

Simple type – Selection

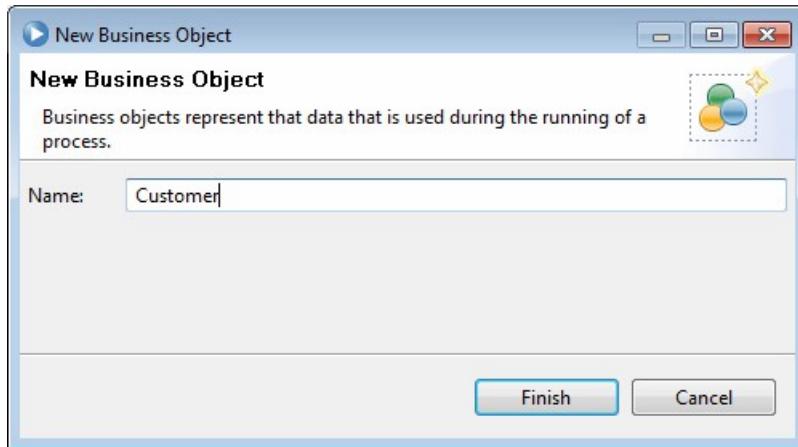
Business Object Types

A Business Object is commonly used to describe a series of attributes that apply to a concept being worked upon in a process. Consider the concept of a "Customer". A Customer has many attributes including *name*, *address*, *payment info*, *purchase history* and perhaps much more. We can define the data type known as Customer by creating a new data type within IBPM PD.

A new data type can be created from the Data entry in the list and selecting Business Object.



Once selected, a dialog window appears asking us to name the new data type we are creating. In this example, we have called our new data type "Customer":



Once the data type has been created, the fields of the new structure may be defined. Interestingly IBPM calls what I call *field* "Parameters". Some products call such data entries "fields", some call them "attributes", others call them "properties" ... so why should we not allow IBPM to call them "Parameters".

An arbitrary number of parameters can be added to the structure by pressing the "Add" button. When a parameter is added, its name and data type can be defined. The data type of a parameter can itself be a complex type allowing for a tree like structure to be created. The parameter can also be flagged as a "List" allowing the one parameter to hold zero or more values.

Common

Name: Customer
System ID: guid:56d35d2f221c8d0e:77332dd6:137a41ab4bf
Modified: admin (Jun 1, 2012 11:04:45 AM)

Documentation: Click [Edit](#) to add or edit text.
[\(Edit\)](#)

Behavior

Definition Type: Complex Structure Type
Shared Object:

Parameters

- name (String)
- address (String)
- phoneNumber (String)

Add Remove Up Down

Parameter Properties

Name:
Is List:
Variable Type: <none> [Select...](#) [New ...](#)
Documentation: [\(View More\)](#)

When a variable type is created, that type is available to all artifacts within the defined Process Application. The type can alternatively be defined within a Toolkit and that toolkit re-used across multiple Process Applications. Creating a Toolkit of data types used across your multiple projects is a good idea.

List Variables

A variable can be defined as a list of some data type. Take care to realize that an IBM BPM List is **not** the same as a JavaScript array. The IBM BPM List data type is mapped to a much richer/different set of semantics. The IBM BPM List data type has the notion of a set of selected items from within the list. This means that given a List, we can also ask that list "Which (if any) items in that list has the property called 'selected'?". This becomes very useful when considering UI based functions.

The List variable has the following operations and properties defined upon it.

listLength - int
listSelectedIndex - Integer
listAllSelectedIndices - *Integer
listSelected - Object
listAllSelected - *Object
f0 toString() - String
f0 toXMLString() - String
f0 toXML() - XMLElement
f0 describe() - XMLElement
f0 removeIndex(Integer listIndex) - void
f0 insertIntoList(Integer position, Object object) - void
f0 listAddSelected(Integer index) - void
f0 listRemoveSelected(Integer index) - void
f0 listClearAllSelected() - void
f0 listIsSelected(Integer index) - boolean
f0 listToNativeArray() - Array
f0 getTypeName() - String
f0 metadata(String key) - Object
f0 isDirty() - boolean

A list variable tracks which values are "selected". An algorithm to remove selected items, the following snippet may be used:

```
for (var i = 0; i < tw.local myList.listLength; i++) {
    if (tw.local myList.listIsSelected(i)) {
        tw.local myList.removeIndex(i);
        i--;
    }
}
```

To add an item to the list, the "insertIntoList ()" method can be used. It is not a "replace" item function but instead will move other entries up as needed.

Q: If I remove an element from the list using removeIndex, is the selected set "adjusted"?

Setting defaults on variables

When a variable is defined in a service or BPD as either input or private, a default value can be provided. The default provides an initial value for private variables and a default value for input variables that are not supplied. The default value is set in the variables tab if the "Has Default" check box is selected. If selected, the text area provides a place in which the default values can be placed. The appearance of the entry of the default values changes depending on whether or not the PD is in advanced mode. If it is in advanced mode, JavaScript can be entered as long as it results in a value for the variable.

Making variables search-able

Process Portal can be used to find instances of processes based upon the values contained in variables within the process. Because IBPM has to do more work to index the values of the variables that are to be looked up, this feature has to be enabled on a variable by variable basis. In the declaration of variables, there is an option to enable searching for Business Data.

The screenshot shows the 'Variables' configuration screen in IBPM. The top navigation bar includes tabs for Overview, Diagram, Variables, Tracking, Process KPIs, and SharePoint. The 'Variables' tab is selected.

Variables Panel:

- Variables category
 - Local
 - Input
 - Output
 - Private
 - loan (Loan)
 - name (String)
 - amount (Decimal)
 - ssn (String)
 - owner (String)
 - Exposed Process Variables

Details Tab:

- Name: owner
- Documentation: (empty text area)
- Add Private, Add Input, Add Output, Link Epv, Remove, Move Up, Move Down buttons
- Is List: (checkbox)
- Variable Type: String, System Data (selected), Select..., New
- Default Value:** Has Default: (checkbox) containing value 1
- Business Data Search:**
 - Available in Search: (checkbox) checked
 - Search Alias: owner
- Performance Tracking:**
 - Track this Field: (checkbox)
 - Tracked Short Name: (empty text area)

Defining and using shared objects

From IBPM v8 onwards, a capability called *Shared Objects* was added. Shared Objects have the ability to mark a Business Object definition as being a Shared Object. See the following image:

The screenshot shows the 'Business Object' configuration interface. In the 'Behavior' section, there is a 'Definition Type' dropdown set to 'Complex Structure Type' and a checked checkbox labeled 'Shared Object'. A red arrow points to this 'Shared Object' checkbox. The 'Parameters' section lists variables 'a (String)', 'b (String)', and 'c (String)'. The 'Parameter Properties' section shows details for variable 'a', including its name, type (String), and documentation. At the bottom, there are 'Advanced Properties' and 'Advanced Parameter Properties' tabs.

Note that Business Objects flagged as shared have a "green" icon as opposed to the "purple" icon of non-shared business objects:

	Non-Shared Business Object
	Shared Business Object

Once flagged, variables created of this BO type are now passed by reference as opposed to passed by value around the environment. What this means is that multiple processes or parallel steps within processes can "see" and "update" the variable simultaneously. If we wish to have multiple processes work with the shared object simultaneously, then we need to pass a "key" that can be used to access the variable in a different process.

This is starting to get subtle so let us try and break this down. Consider the previous screen shot which shows the creation of a new Business Object type called "MySharedBO". Now consider an instance of a process which declares a variable of type "MySharedBO".

We should be able to easily imagine an instance of the MySharedBO Business Object being "held" somewhere for future reference. Imagine that we wished to have a second process leverage the same shared object. The second process can declare a variable of type MySharedBO but ... to get a reference to the object, will have to execute:

```
tw.local.myShared2 = new tw.object.MySharedBO(keyValue);
```

but wait ... where did the "keyValue" parameter come from?

On the original reference to the Business Object, we can access its "key" value with:

```
var keyValue = tw.local.myShared1.metadata("key");
```

The 'metadata ("key")' method returns a string representation of the key to that shared object. Note that "key" in this call is actually an explicit keyword and not a variable name.

When a change is made to the data contained within the Shared Object that data is written to a database behind the scenes. Only when the data is written, will the data be able to be seen by a partner. The data is written when a service that modifies it completes or when an explicit call to the `save()` method (found on the business object) is called.

The data associated with the Shared Object is associated with a process instance that created it. When the process instance data is cleared, so is the Shared Object.

One important area that requires further consideration is **when** a process sees the changes that another process may have made to the data. The answer is **not** quite what one may trivially expect. Imagine two process instances, P1 and P2. P1 creates an instance of a Shared Object and sets its values. P2 then also creates an instance referencing the key from P1. Now P2 changes a value in the data and executes a `save()`. What does P1 now see?

The choices would appear to be:

1. The original value of the field as originally set by P1
2. The new value of the field as set by P2

The answer is "it depends".

This is the concept of the "latest version of the data in the Shared Object". This is what will be used when either a new instance of the Shared Object with the given key is created or an existing instance is re-loaded with an explicit `load()` call. So if P1 executes a `load()`, it will always get the latest data.

A reload of the data within P1 happens automatically when a step in a process is transitioned or

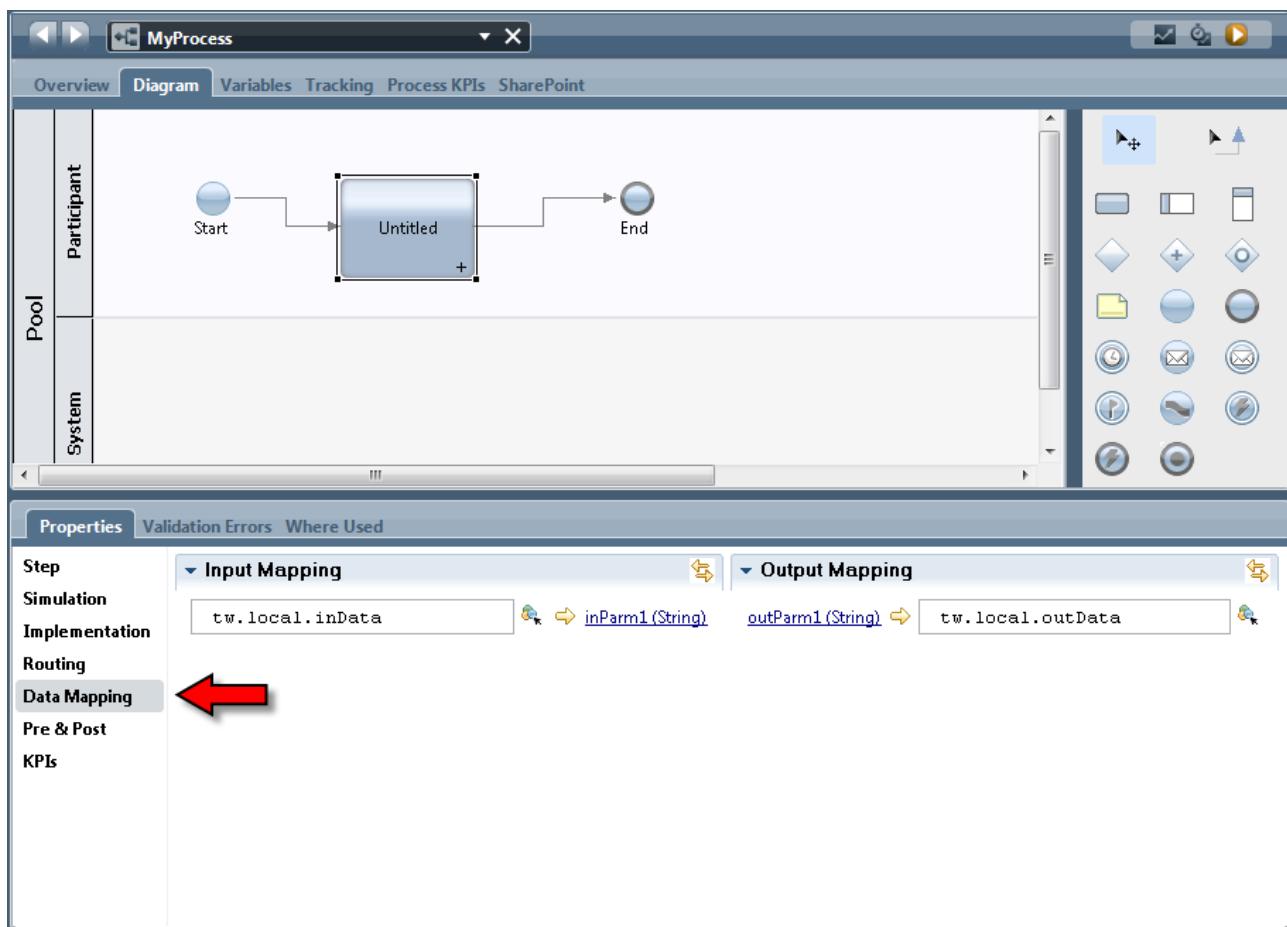
when a task is woken up. However, within the context of a single straight through service, the values of the variable will not change unless it is explicitly re-loaded. The transition from step to step in a process is an implicit re-load as is the awakening of the task.

BPD Variables and Service Variables - Mapping

A Business Process Definition has variables associated with it. Some variables might be input to the process, some may be output from the completed process and some may be local variables only visible to the process.

In addition to the a BPD having variables, so too can a IBPM Service. Just like a BPD, a IBPM Service has input, output and local variables. These are defined in the definition of the IBPM Service.

A IBPM Service is usually invoked from within the context of a BPD through an Activity node. When the node is defined, it is associated with a IBPM Service definition. If we think about this for a moment, we see that the IBPM Service has some input variables that it is expecting to contain data and will return some output variables that make data available to the process as a whole. These variables in the IBM service need to be "mapped" to variables that are in scope within the BPD that is calling the service.



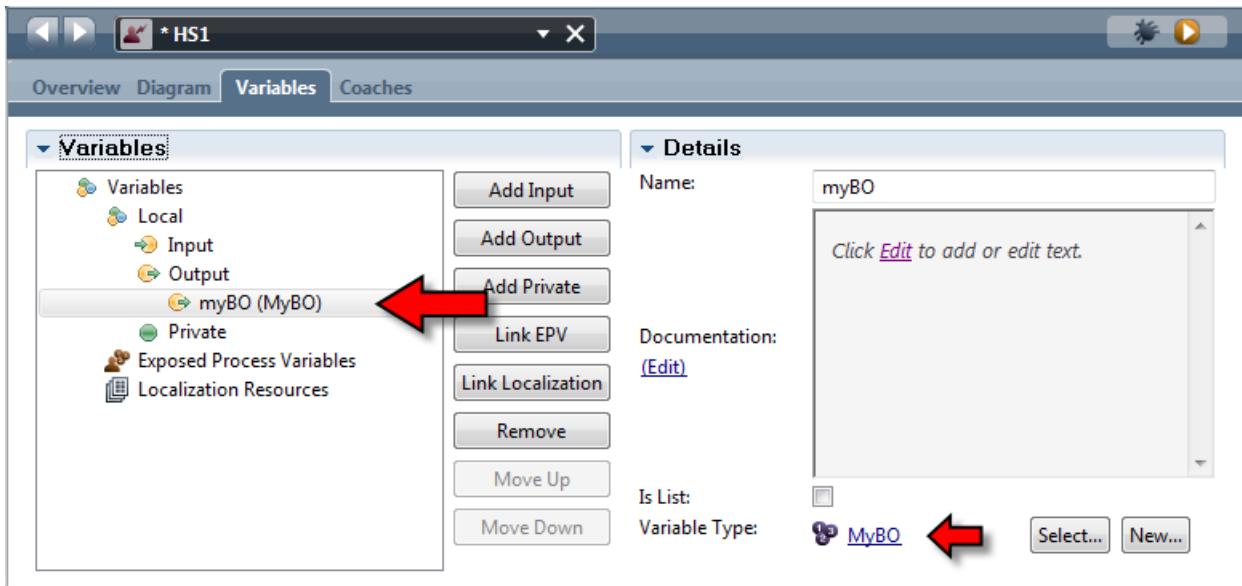
When an Activity is selected in the BPD editor, the properties section shows a Data Mapping tab. Selecting this shows the expected input variables to the associated IBPM Service as well as the generated output variables. These can then be mapped to variables within the current BPD. The icon beside the "boxes" (🔍)provides for a smart search of the defined variables so we do not need to remember the JavaScript variable names.

If a variable is defined as a complex type, it will need to be initialized before values can be assigned to it. Its initial value is "null" which has no fields. A good way to achieve initialization is to check the has default check box in the data type definition.

Variable identity and UUID

Imagine you create a new Business Object type called "MyBO". What is the identity of that Business Object type? As you might imagine, there is more to this simple question than meets the eye. The normal way of thinking is that if I create a data type called "X" then it has an identity of "X". Unfortunately, IBM BPM doesn't think that way. Internally within the heart of the product, all artifacts (including all data types) are given a generated UUID value. It is this opaque UUID that the product uses to track references to data types.

To illustrate, consider the following Human Service definition which has an output parameter called "myBO" of data type "MyBO":



The Human Service definition (called HS1) can be read as:

"This service returns an output called 'myBO' of type 'MyBO'".

However, this is only an illusion. The way the product really thinks of this is:

"This service returns an output called 'myBO' of type
'guid:7b9691c16dd21a42:1e6925ca:142dd70d25a:-7ffa'"

Hmm. That is quite a difference. If we switch on advanced mode in PD and then look at the definition of the business object, we can see that UUID there:

Normally, I wouldn't try and uncover internals knowledge such as this as it is usually irrelevant to the model and used of the BPM product itself however, there are some ugly ramifications associated with this implementation story.

As a thought process, consider the following:

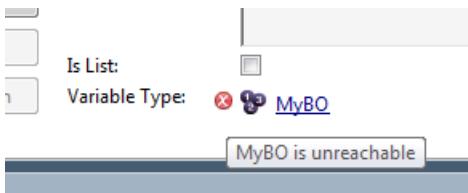
1. Create a new Business Object type called MyBO
2. Create a Human Service that returns a variable of type MyBO
3. Delete the Business Object type called MyBO
4. Create a new Business Object type called MyBO identical in structure to the first

Are we now consistent? It would seem so, we have a business object called MyBO which contains all the fields we want. Unfortunately, we are broken.

In step 2, when we create a Human Service that returned a variable of type MyBO, the Human Service actually defined that it returned a data type of a UUID. That UUID was associated with the data type created in step 1. When we performed step 4, a brand new UUID data type was created that is not the same as that of step 1.

We end up with the Human Service error showing:

If we hovered over the error we would see:



But yet if we examined the list of data types, we would see that MyBO is a defined data type.

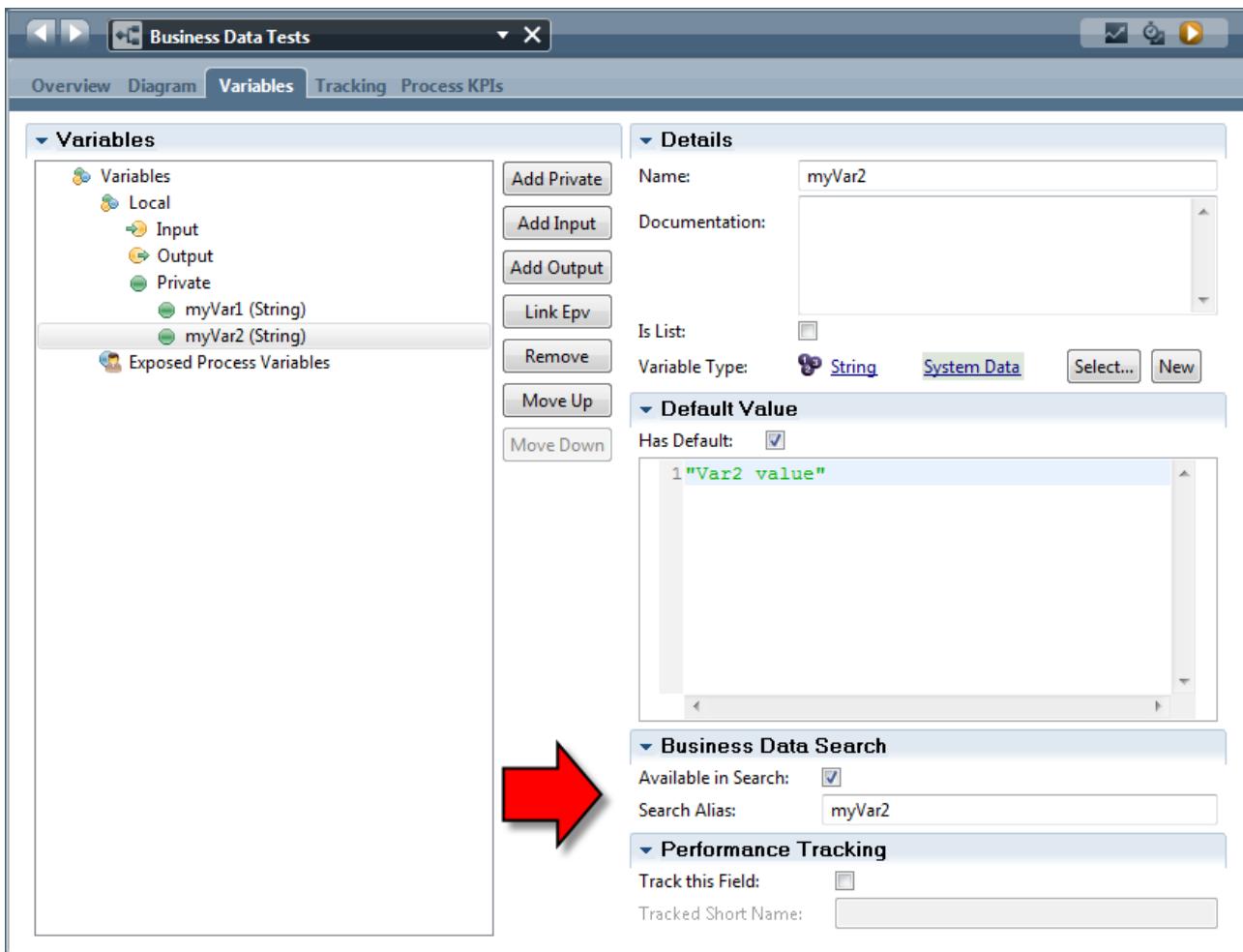
The fix would be to remap the variable type to the new MyBO definition. To most folks (including myself) this is an odd situation to be in.

The moral of the story is to be very careful with the concept of deleting and recreating data types. Deleting something and then recreating it is **not** the same as modifying it. This story also comes into play when you think that you can move data types into toolkits.

Exposing Business Data for Searches

Imagine that we have hundreds (perhaps thousands) of process instances running handling orders for customers. If a customer calls in and asks "What is the status of my order #1234?" how can we find the process associated with this?

When process variables are defined in IBPM, we have an opportunity to flag them as being available in Business Data Searches. This means that we can execute searches using the exposed names and values of variables in a process. For example, if a process has a variable called "orderNumber" and we exposed that for searching, we can now execute a search for process instances where "orderNumber = 1234".



Accessing variables from JavaScript

It is common to want to read or write the values of variables from within a JavaScript environment. When a variable is defined at the BPD or service level, it is available within that BPD or service within the JavaScript scope of:

```
tw.local.*
```

The IBPM PD commonly provides entry assist to show you the names of the in-scope variables that can be used in a JavaScript fragment. Note that the entry assistance accessed by CTRL+Space in the editor can be used to substitute the characters "twl" for the string "tw.local.".

List variables can be accessed using square bracket notation with an index value starting at 0. When assigning to a list, if an index value is used that is greater than the size of the list, this is acceptable. To make this clear, imagine a list with two existing entries which would be [0] and [1]. If we assign to the list with an index of [3], this would produce an entry for [2] of data type ANY and no value.

Environment Variables

Instead of hard-coding values in a process, we may wish to externalize these values and re-use them across our solution. IBPM PD allows us to define environment variables in either Process Applications or in Toolkits. Once defined in there, they can then be referenced in the solution.

The environment variables section can be accessed from the Process App Settings within the Environment tab:

A screenshot of the Lombardi Process Admin Console. At the top, there's a toolbar with icons for back, forward, and search, followed by a dropdown labeled "Process App Settings". Below the toolbar is a navigation bar with tabs: "Overview" (selected), "Environment" (highlighted with a red arrow), and "Services". The main content area is titled "Environment Variables". It features a table with columns: "Key", "Default", "Development", "Test", "Stage", and "Production". To the right of the table are two buttons: "Add" and "Remove". A gear icon is located in the top right corner of the content area.

Variables defined in the Process Application can be accessed as:

```
tw.env.<Env Variable Name>
```

Variables defined in a Toolkit can be accessed as:

```
tw.env.toolkit.<Toolkit Name>.<Env Variable Name>
```

The values of the environment variables can be changed through the Process Admin Console.

After starting it up, select the Installed Apps button:

A screenshot of the Lombardi Process Admin Console. At the top, there's a header with the text "Lombardi | Process Admin Console" and a "Server Admin" button. To its right is a green button labeled "Installed Apps" with a red arrow pointing to it. Below the header is a toolbar with a "Sort Snapshots By:" dropdown set to "Application Name" and buttons for "All" and "Active". The main content area displays a list of application snapshots. The first item is "Kolban1 (KOLB001) - Snapshot #2 - Testing", which has two sub-items: "Inspector Test - 0 instances" and "My Loan - 0 instances". Below this is another item: "Kolban1 (KOLB001) - sn1".

Next, select the application whose environment variables are to be changed. A list of the environment variables and their current values can then be seen:

The screenshot shows the Lombardi Process Admin Console interface. At the top, there's a navigation bar with 'Lombardi | Process Admin Console' on the left, 'Server Admin' and 'Installed Apps' in the center, and a green button for 'Installed Apps'. Below the navigation bar, the title 'Kolban1 (KOLB001) - Snapshot #2 - Testing' is displayed, followed by tabs for 'Exposing', 'Role Bindings', and 'Environment Vars'. A red arrow points to the 'Environment Vars' tab. In the main content area, there's a search bar with 'abc' and a value '1' in a dropdown menu.

The environment variables default values can be based on the setting of the `environment.type` attribute set in the `install.properties` used when IBPM is installed.

To make changes to environment variables, one must be a member of the administrators group.

Exposed Process Values (EPVs)

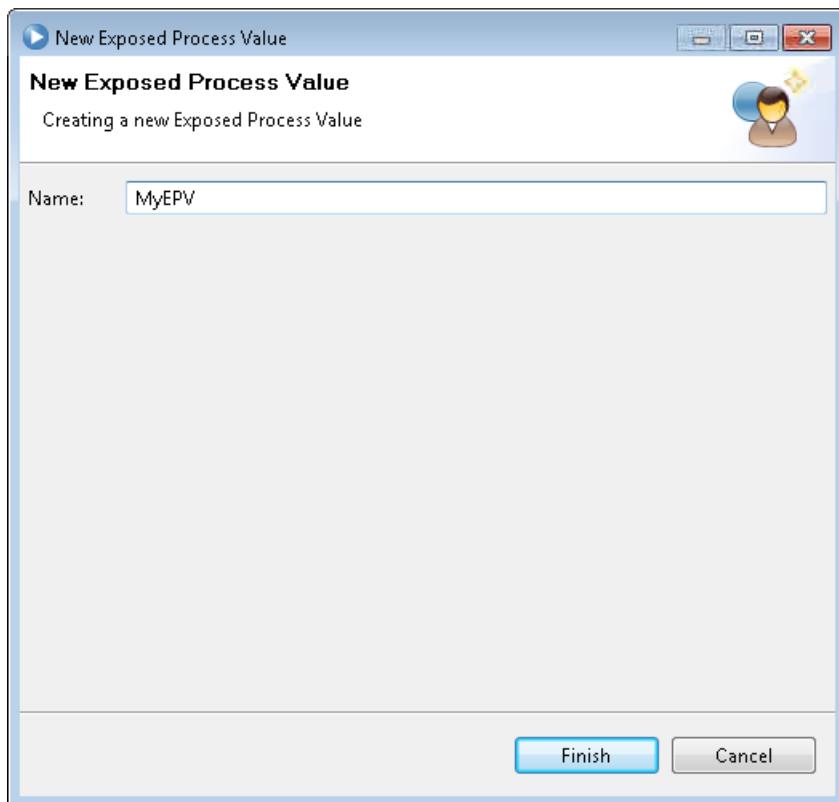
Consider a business process that utilizes some business data value such as the current tax rate or the interest rate on loans. This value is obviously not a constant as it may change over time. What we want is a way to supply such values to processes while at the same time making them easy to modify as needed. The concept of Exposed Processes Values (EPVs) is the IBPM solution.

An EPV is a named container that holds one or more variables where the values of those variables can be defined through the Process Administration console.

An EPV is defined within a Process Application or Toolkit from the Data catalog in the library.

The screenshot shows the 'Process App Settings' dialog. On the left is a sidebar with categories like CEMT-T (All, Processes, User Interface, Implementation, Decisions, Data, Performance, Setup, Files) and TOOLKITS (System Data, Coaches, Coach Bonus). The 'Data' category is selected. The main area has tabs for 'Overview', 'Environment', and 'Servers'. A 'Create New' wizard is open, showing options: 'Business Object' (selected), 'Exposed Process Value' (highlighted with a red arrow), and 'User Attribute Definition'. A note at the bottom says 'Click [Edit](#) to add or edit text.'

When the wizard appears to create a new EPV, a name can be supplied for it.



Once created, the editor provides a rich set of settings for it. At its basic level, we define a set of one or more variables and give each variable a name, data type and default value.

The screenshot shows the 'Exposed Process Value' configuration editor for an item named 'MyEPV'. The interface is divided into several sections:

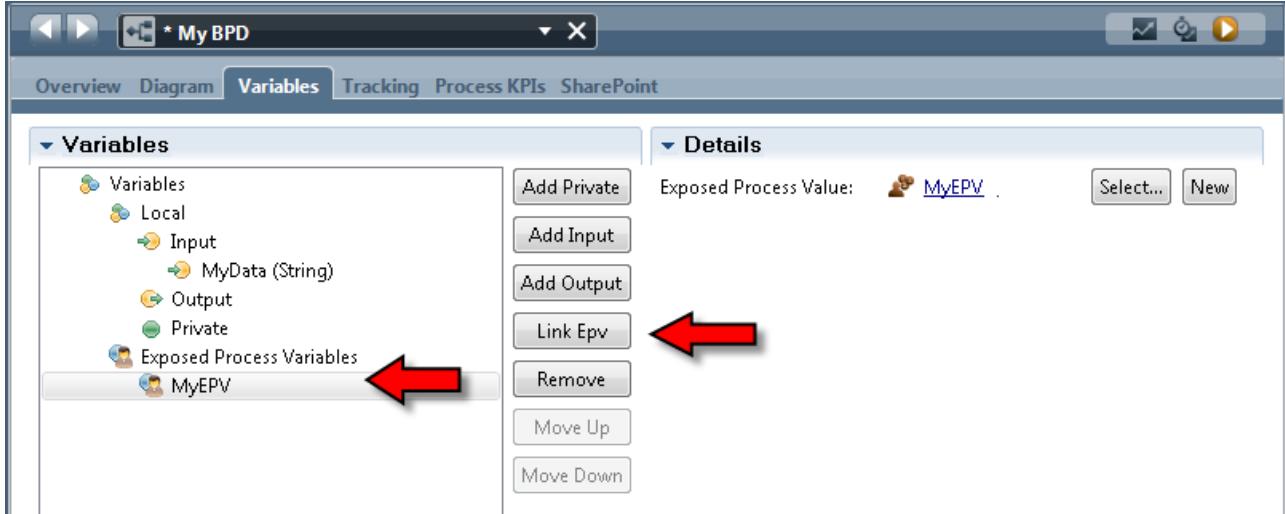
- Common:** Name: MyEPV, Modified: tw_admin (September 17, 2010 11:39:06 AM CDT), Documentation: (empty text area).
- Details:** Feedback E-mail Contact: (empty text area), External Description: (empty text area).
- Exposed Process Value Variables:** Untitled1 (String), Untitled2 (String), Untitled3 (String). Buttons: Add, Remove.
- Variable Details:** Untitled3 (External Name), Untitled3 (Variable Name), Untitled3 (External Description), Untitled3 (Default Value), Untitled3 (In-Progress Tasks Use New Values), Untitled3 (Variable Type: String). Buttons: Select..., New.
- Exposing:** Exposed to: <none>, Buttons: Select..., New, X.

Updates and access to the EPV is performed through the Process Admin Console. Only those users

to whom the EPV is exposed can modify its value. The Exposed to section names a group of users who are allowed to manage the EPV.

The Feedback E-mail contact provides the email address of the "owner" of this EPV. This allows users to contact the owner for questions and requests.

In a BPD or service definition, within the variables section, an EPV can be linked or associated with the BPD or service:



Once the link has been made, the values of the EPV can be used within the solution. The JavaScript expression:

```
tw.epv.[EPV Name].[EPV Variable Name]
```

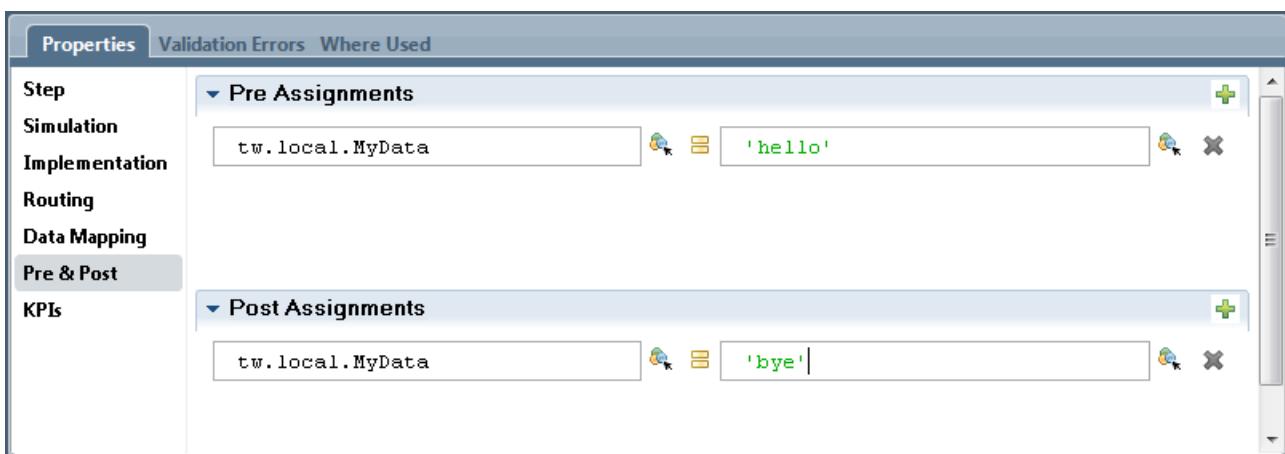
can be used to refer to those values.

See also:

- Admin Tools > Manage EPVs

Pre and Post Assignments

Associated with an activity element is the concept of pre and post assignments. These sections of the activity definition allow for updates of variables prior to executing the core of the activity and also to allow for updates immediately following the core of the activity.



The number of pre and post assignments (if any) on an individual activity is configurable.

There have been a number of discussions on the value and appropriateness of pre and post

assignments. The questions arise when thinking about readability and maintainability of a solution that uses this technique. The markers added to an activity to indicate that it has pre or post assignments are small and easily missed. There is a school of thought which says that this capability should not be used often and instead, explicit steps be added to the process diagram to achieve the same effect.

Variables and XML

Each complex variable has an XML representation. To examine this, let us initially look at a complex data type called `Type1` that looks as follows:

The screenshot shows a software interface for managing variable types. At the top, there's a toolbar with icons for back, forward, and close, followed by the title "Type1". Below the toolbar, the main area is titled "Variable Type".

Common Section:

- Name: Type1
- System ID: guid:c23d30e8591083d4:-2f86c106:12d195953d9:-7fea
- Modified: tw_admin (December 25, 2010 6:12:10 PM CST)
- Documentation: (empty text area)

Behavior Section:

- Definition Type: Complex Structure Type

Parameters Section:

- a (String)
- b (Boolean)
- c (Integer)

Parameter Properties Section (for parameter 'a'):

- Name: a
- Is List: (checkbox)
- Variable Type: String
- Documentation: (empty text area)

At the bottom, there are tabs for "Advanced Properties" and "Advanced Parameter Properties".

This structure contains three fields called "a", "b" and "c". If we create an instance of this variable and execute `tw.system.serializer.toXml` method, we get the following:

```
<variable type="Type1">
  <a type="String"><![CDATA[value for a]]></a>
```

```

<b type="Boolean"><![CDATA[false]]></b>
<c type="Integer"><![CDATA[123]]></c>
</variable>

```

Breaking this apart, we see an element called `<variable>` as the root with `type` attribute that names the data type of the variable as a whole. The nested elements are named after the fields within the structure again using the `type` attribute to define the data type of the fields.

Given an XML representation of the data, we can convert back to a variable instance with the method called:

```
tw.system.serializer.fromXml()
```

This takes either a `String` or `XMLElement` object as a parameter and returns a new `Object` instance.

For variables defined as lists, the structure is a little different.

```

<variable type="Item[]>
  <item type="Item">
    <name type="String"><![CDATA[Item1]]></name>
    <quantity type="Integer"><![CDATA[10]]></quantity>
  </item>
  <item type="Item">
    <name type="String"><![CDATA[Item2]]></name>
    <quantity type="Integer"><![CDATA[20]]></quantity>
  </item>
</variable>

```

In the Data Type definition, there are properties for the XML serialization:

Advanced Properties		Advanced Parameter Properties	
XML Serialization:	Exclude from XML: <default>	XML Serialization:	Exclude from XML: <default>
Anonymous Type:	<default>	Node Type:	<default>
Type Name:	<default>	Name:	<default>
Namespace:	<default>	Type Namespace:	<default>
Element Name:	<default>	Min Occurs:	<default>
Element Namespace:	<default>	Max Occurs:	<default>
Base Type Name:	<default>	Nillable:	<default>
View XML Schema		Order:	<default>
		Wrap List:	<default>
		Anonymous List Type:	<default>
		List Type Name:	<default>
		List Item Name:	<default>
		Time Zone:	<default>

Notice the `View XML Schema` button. When clicked, an XML Schema representing this data type is shown. Notice also that the name space for this schema looks *odd*. It is created from the hostname and port number of the machine running the authoring environment. This can cause problems as this schema value will change depending on where the schema is displayed. If your IBPM solution is going to utilize XSDs it is strongly recommended to explicitly name a Namespace value. This can be any constant of the form:

```
http://<value here>
```

Using Variables

Since variables in IBPM are available within JavaScript then any JavaScript functions that execute on such variables can be used.

If a variable is used to represent a business object, remember that the variable is a reference to that

data and **not** its value. For example, if we have code that looks as follows:

```
var x = new tw.object.MyDataType();
var y = x;
```

Then the variable `y` is actually a reference to the same object as the variable called `x` and importantly not a copy. Changing a field in `x` will cause the exact same field in `y` to be also changed. If the desire is to explicitly make a copy of a variable, consider using the IBPM provided XML serializer and de-serializer. For example:

```
var x = new tw.object.MyDataType();
var temp = tw.system.serializer.toXml(x);
var y = tw.system.fromXml(temp);
```

On a broader scale, there is a question on what happens when a business object is passed from one entity to another (for example, a process to a sub-process)? In computer science there is the concept of pass-by-value vs pass-by-reference. With pass-by-value, a copy of the variable is passed and hence changes to that copy do not affect the value of the original variable. With pass-by-reference, a reference (pointer) to the original variable is passed. This means a change to the content of one variable immediately affects the value of the other variable. In IBPM both pass-by-value and pass-by-reference are used in different circumstances.

Scenario	Passing mechanism
Process invokes a sub-process	Pass-by-reference
Process invokes a service	Pass-by-value
Service invokes a nested service	Pass-by-reference

When assigning a value to a Business Object, one can use the JavaScript style of definition. For example:

```
tw.local.myData = {
  a: "My A Value",
  b: 123,
  c: 3.141,
  d: {
    x: "X Value",
    y: "Y Value"
  }
};
```

would work for an appropriate definition of the data type of `myData`. The use of this technique can dramatically simplify data assignment.

Determining the type of a variable

On occasion, you may find that you have a local JavaScript variable and not know what type it is. You can dynamically determine its type. See the following example:

```
var myDate = new tw.object.Date();
log.info("The data type of the object is: " + Object.prototype.toString.call(myDate).slice(8, -1));
```

The case of the mysteriously changing variables

Consider the following story:

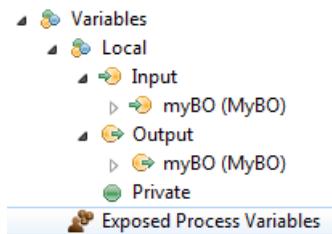
First, imagine a data type called "MyBO" that has three fields:

- a (String)
- b (String)
- c (String)

Now imagine a process called "LinkedProcess1" which looks as follows:



The parameters to this process are:



What this process does is take an instance of "MyBO" as input, set the value of property "a" to say that it is "Linked Process 1" that set it and finally create a task that shows the content of the MyBO data type.

If we ran it by itself, an example of what the task would show is:



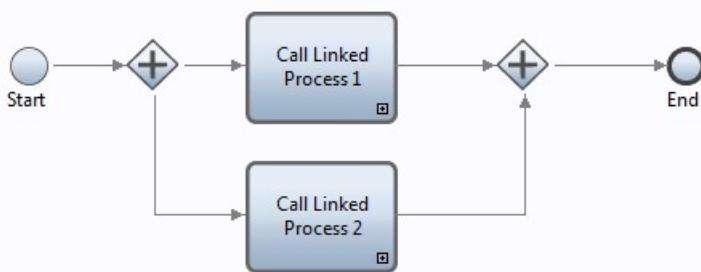
Now imagine a second process called "LinkedProcess2". It looks like:



It is identical to "LinkedProcess1" with the exception that it sets the "a" property to say that we are in "Linked Process 2".

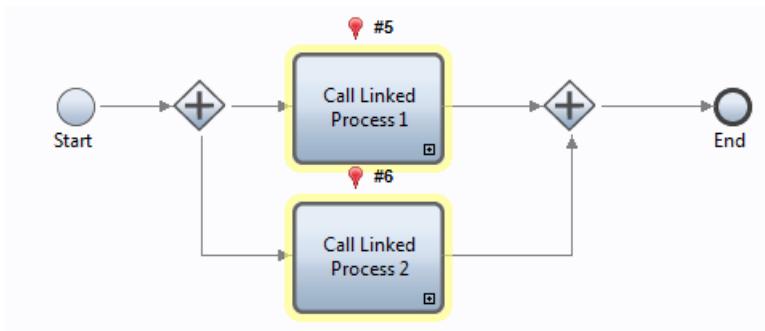
Now ... here comes the next part ...

Imagine one last process called "MyProcess" which looks like:



This process has a single "MyBO" variable which is passed as input to two parallel linked processes. These are mapped to "LinkProcess1" and "LinkProcess2".

When I run this process, as expected, I see it block waiting on two tasks:



And I see two tasks available to be worked upon:

Status	Owner	Subject	Priority	Due Date	Task Id
Received	(ROLE) All Users	"Linked Process 2 - Show MyBO"	Normal	Dec 19, 2013 8:...	250
Received	(ROLE) All Users	"Linked Process 1 - Show MyBO"	Normal	Dec 19, 2013 8:....	251

Now I open the task for "Linked Process 1" ... and here is what I see:

A	Linked Process 1
B	B1
C	C1
OK	

So far, all as expected.

Now I open up the task for "Linked Process 2" ... and here is what I see:

A	Linked Process 1
B	B1
C	C1
OK	

Do you see it? :-)

The "a" field has the **wrong** value!!! It should say:

A	Linked Process 2
B	B1
C	C1
OK	

So what happened?

The answer can be found in the IBM BPM documentation found here:

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r5m0/topic/com.ibm.wbpm.bpc.doc/modeling/topic/_declaring_variables_A.html

Here is what caught my eye:

[IBM Business Process Manager, V8.5, All platforms](#) > [Programming IBM Business Process Manager](#) > [Developing using the JavaScript API](#) > [Declaring and passing variables](#)

How variables are passed in Process Designer

Using variables, business data in Process Designer is passed between processes and linked processes, between processes and services, and between services and services.

Variables capture business data. If the business data is a simple type (for example, a String) then the variable contains a value of the business data. If the business data is a complex type then the variable is a reference to an object containing multiple values.

Variables can be passed by reference or by value, as described in the following table.

Table 1. How variables are passed

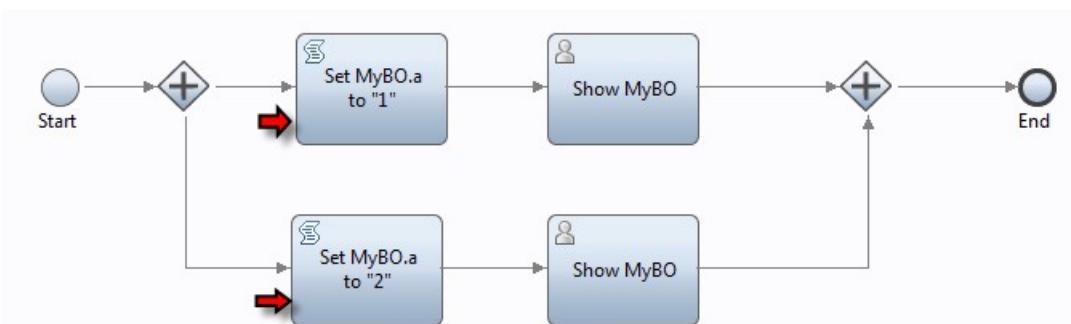
From	To	Type	Pass by
Business process definition (BPD) activity	Service	Simple	Value
BPD activity	Service	Complex	Value (the BPD and Service have separate copies of the business object)
BPD activity	Linked BPD	Simple	Value
BPM activity	Linked BPD	Complex	Reference to the same business object
Service	Nested service	Simple	Value
Service	Nested service	Complex	Reference to the same business object

Variables passed by reference

Most data interactions in a process are passed by reference (from BPD to BPD or from service to service). Therefore, most of the time the same object

When we call from one process to another as a link, the variable is passed by reference and not by value (a copy).

What this means ... is that our BPM process called "MyProcess" (see above) ... is logically the same as:



And if we look closely at this diagram, we see that the two marked activities BOTH set the same business object instance field to two different values ... in parallel and since one of them will run before the other in reality, the last one executed will set the single value ... and it is that single value that is shown in the Coaches.

With this understanding in mind, we can now see how to avoid the problem ... have two business objects ... one for "Linked Process 1" and one for "Linked Process 2" ... and don't share the same business object between both.

User Interfaces and User Interaction

It is common to want to present information to users and have those users return responses. Such human interactions were called Human Tasks in the old WPS product. In IBPM, they are called Human Services. Within a BPD we can define a User Activity and define a Human Service as the implementation of that User Activity. Contained within a Human Service are one or more "Coach" units. Each Coach corresponds to a single screen the user sees.

A Coach describes the visual nature of what a user sees and associates that with data contained within the process. The Coach is created in the Process Designer tool within the Human Services editor. Coaches may only be included in Human Service definitions. They may not be defined in other service types.

With the arrival of the IBM BPM v8.0 release, the implementation and design of Coach technology radically changed. To provide upgrade protection for users who had been using releases prior to v8, the **old** Coach technology remains in the product. This older implementation is termed "Heritage Coaches". For new users, the Coaches that should be used are the current "default" Coaches which are sometimes referred to as "Next Generation Coaches". All references to Coaches in this document will refer to the default Coaches found in v8 unless explicitly stated to be otherwise.

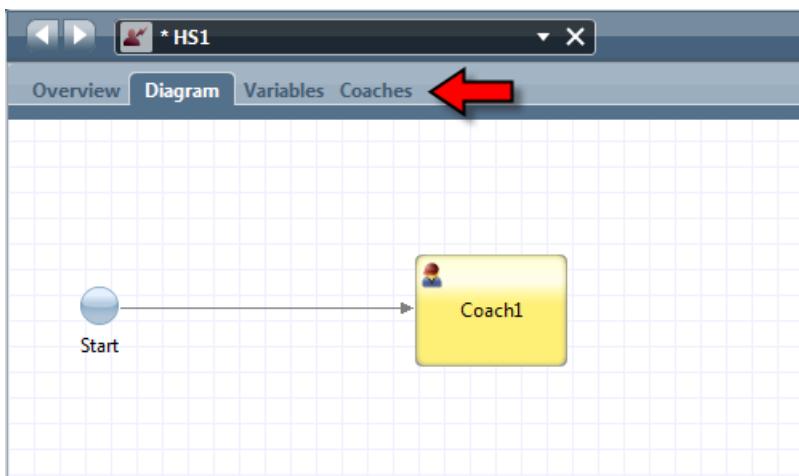
In the service diagram editor, a coach is represented by the following building block.



This can be created from the palette by dragging and dropping the following icon:

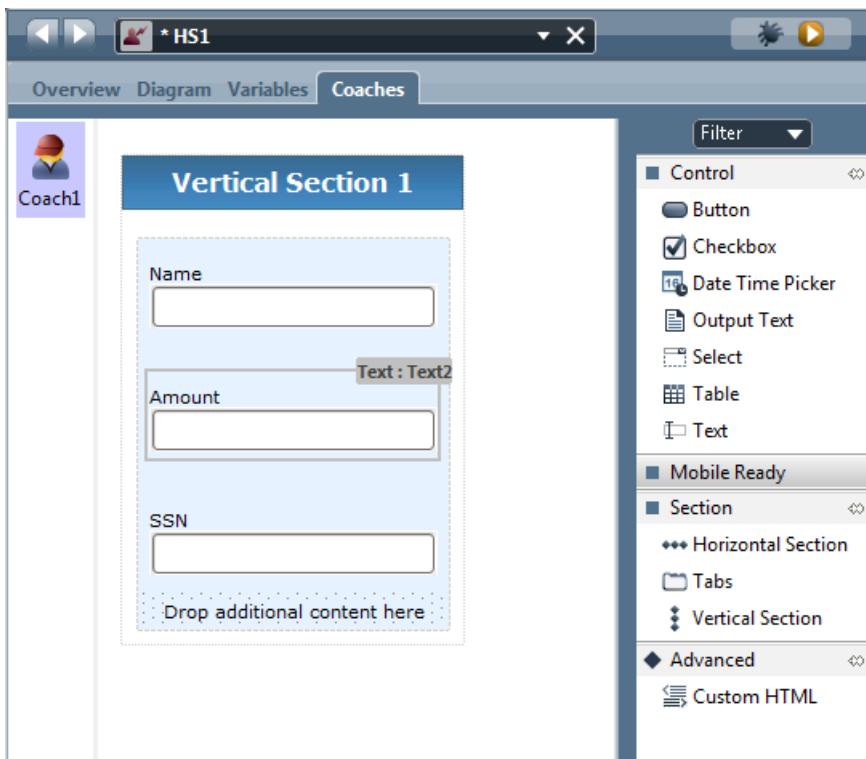


The Human Service editor has a tab that lists all the Coaches in that human service definition.



When a coach component is opened, a canvas area into which the visual aspects of the screen can be drawn. This editor is called the Coach Designer.

To the right of the canvas is a palette of selectable building blocks from which a page can be created.



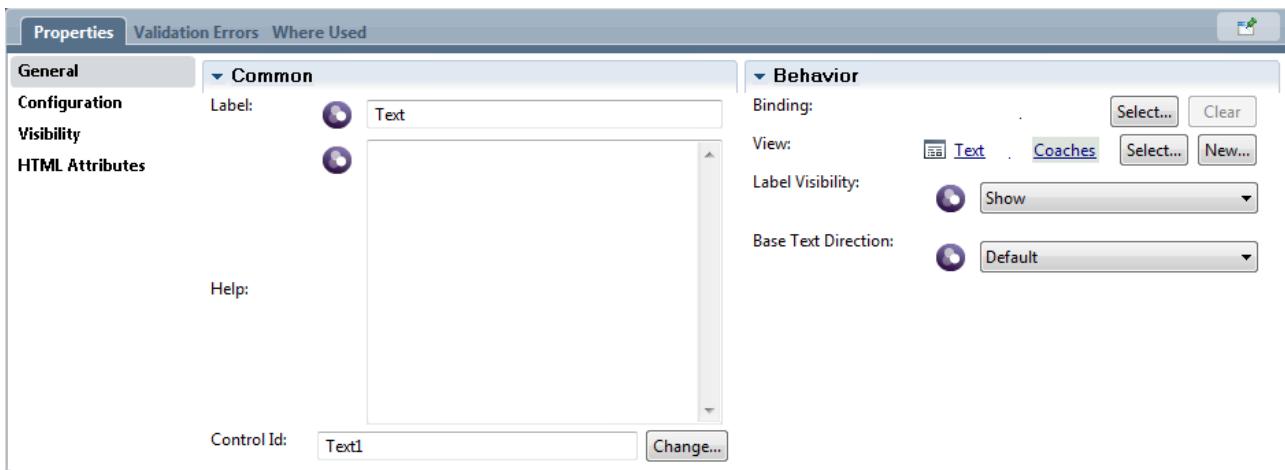
To the left of the canvas area is the list of all coaches in this particular Human Service definition. The coach which is to be edited can be selected and the canvas editor changes its focus to that coach instance.

Coach Views

Contained within a page are visual building blocks that, when aggregated together, form the content of the page. IBM BPM calls each of these building blocks a "Coach View". Other UI technology products (such as Dojo) would consider these to be called "widgets" or "components" so if you are more familiar with those names, feel free to think of a Coach View as being a Widget or Component.

General Coach View settings

When a Coach View is dropped onto the Canvas, the properties area contains a variety of settings:



In the General area we have quite a few items of interest to us. First there is the `Label`. This is a string value that can be used as a label on many of the Coach Views.

The `Binding` property allows us to select a variable instance who's content will be used to populate the Coach View and/or will be populated by the Coach View.

Next the `View` entry names the type of Coach View that will be displayed on the Coach.

The `Control Id` names the unique identity of the Coach View within the Coach. Quite why we didn't call it the "Coach View Id" is unclear and feels like a hangover from the previous release.

The `Label Visibility` entry determines if the label associated with the Coach View will be shown when rendered. For some Coach Views such as the section views, we simply want them to be containers with no headings associated with them.

The `Base Text Direction` is used to define text direction layout for national language support.

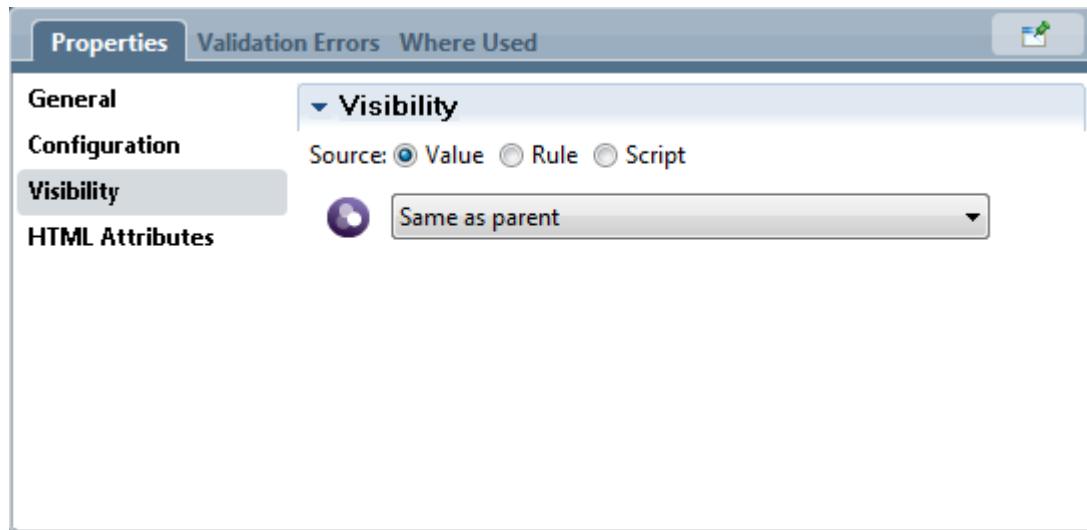
Visibility Coach View Settings

The Visibility settings controls some additional aspects of the Coach View. The phrase "Visibility" is an odd one as to my mind it should control whether the Coach View can be seen by the user (visible) or not shown on the Coach (hidden). However, the values of this property allow for much more. The property value can take one of a defined set of possibilities. These are:

Value	Meaning	String value
Default (Same as parent)	Take the value from the parent Coach View	DEFAULT
Required	The Coach View is visible, editable and must have a value supplied for it.	REQUIRED
Editable	The Coach View is visible and editable	EDITABLE
Read Only	The Coach View is visible but not editable	READONLY
None	The Coach View is not visible and does not reserve any space for it	NONE
Hidden	The Coach View is not visible but does reserve space for	HIDDEN

| it |

The value for the property can be supplied in a number of different ways. The first is by direct selection:

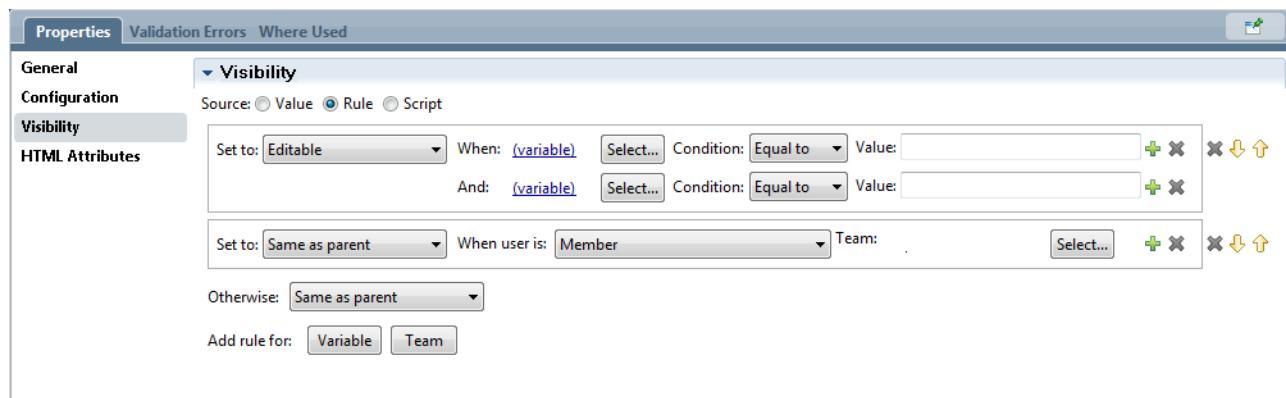


The pull-down options are:

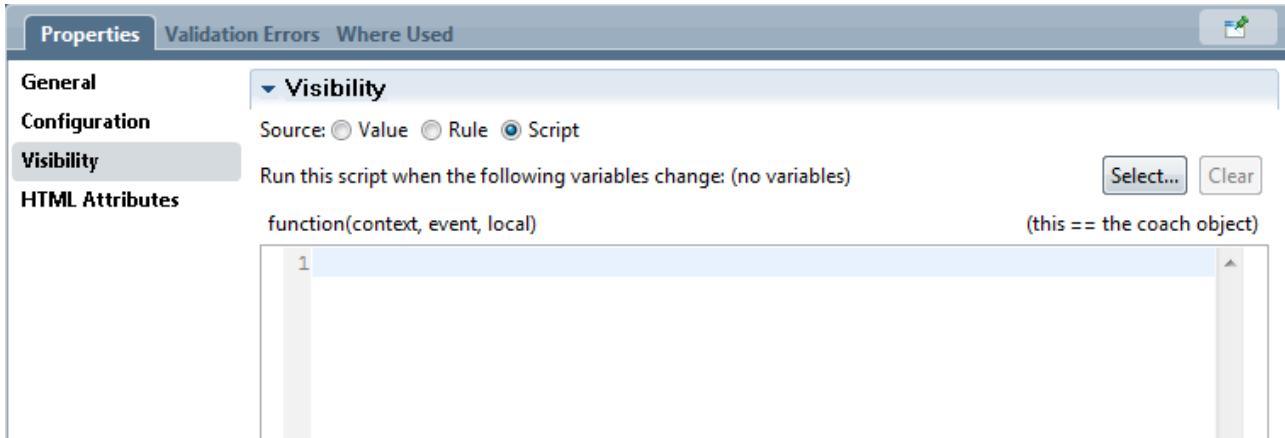
- Same as parent
- Required
- Editable
- Read-only
- None
- Hidden

In addition, the property value can be supplied by a named variable. The value contained within the variable will then be used to control visibility.

The second style for setting the property is Rule. In this mode we can define a set of expressions that are evaluated. The first expression that evaluates to true is the one that sets the visibility property value. The rules can use the values of variables or team membership (or lack of membership) to determine the values.



The final style is called Script.



IBM supplied stock Coach Views Controls

When IBM BPM is installed, IBM supplies a starter set of Coach Views that are basic controls and are available out of the box to be used within Coaches. These stock controls are:

Button

This control shows a button on the Coach web page. When clicked, it can broadcast a boundary event that will inform the Human Service that the coach has completed.

From the palette it looks as follows.



When dropped on the canvas it looks like:



Example visuals on a web page:



The binding for this control is to a Boolean valued variable which records whether or not the button was clicked.

Configuration:

Name	Default	Description
Allow multiple clicks	FALSE	Determines whether or not the button is allowed to be clicked repeatedly. If the button is not allowed to repeat then after its first click it will be disabled.

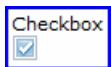
If the button's visibility is set to "READONLY" then it is disabled and will not respond to button presses but still be visible in a disabled state.

See also:

- Boundary Events

Checkbox

Checkbox



This control shows a check-box on the Coach web page. When clicked, it will toggle its value from true to false or false to true.

Example visuals on a web page:

Checkbox 1

Checkbox 1

Checkbox 1

Yes No

Checkbox 1

Yes

The binding for this control is to a Boolean value variable which records whether or not the checkbox was checked.

Configuration:

Name	Default	Description
Show As	Checkbox	Describes how the checkbox will be shown in the coach. Options are: <ul style="list-style-type: none">• Checkbox• Two Radio Buttons• Switch
True Label	Yes	The label used for radio buttons and switch styling for true value
False Label	No	The label used for radio buttons and switch styling for false value

To set the size of the Text control, the following CSS can be used:

```
div[data-viewid="viewId"] {  
    width: 20em;  
}
```

This CSS can be included in an HTML widget ... eg:

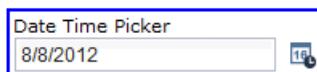
```
<style type="text/css">  
div[data-viewid="myTextId"] {  
    width: 20em;  
}  
</style>
```

Date Time Picker

The Date Time Picker provides a mechanism to show a date (and optional time) to the user and, if desired, allow them to pick a new value. From the palette, this control looks as follows:

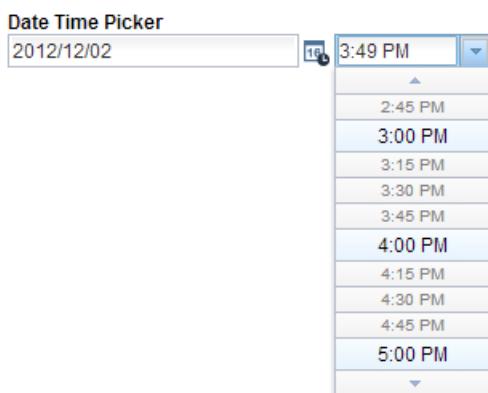
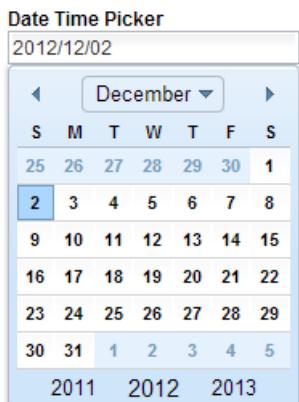
 Date Time Picker

When added to the Coach, it looks like:



This control allows the user to select a date and time.

Example visuals on a web page:



The binding for this control is a variable of type `Date`. This will hold the date value selected.

The configuration options for this view are:

Name	Default	Description
Show Calendar	On Click	A calendar from which a date may be picked can be shown. How and when this is shown is configurable <ul style="list-style-type: none">• On Click – The calendar is shown when the user clicks in the text field for the date• Inline – The text field for the date is never shown instead, the calendar is always shown• Never – The text field is only shown. No calendar is displayed.
Calendar type	Gregorian	<ul style="list-style-type: none">• Gregorian• Hebrew• Islamic
Include Time Picker	FALSE	When set to true, an entry text box and selection for a time value is shown.
Date Format	"MM/dd/yyyy"	A string representation of a date format used to show the date. This models from the Java <code>SimpleDateFormat</code> patterns.
Blackout Dates	None	A list of dates which the user may not select.

The width of the Date Time Picker can be changed using CSS, for example:

```
.Date_Time_Picker .dijitDateTextBox {  
    width: 50px;  
}
```

Decimal

The Decimal Coach View is used to show and enter decimal numbers.

Decimal

- **Currency** – If the data is used as a currency value, setting this to the ISO 4217 coding formats the data as a currency.
- **Other Currency**
- **Currency Symbol** – An override to supply the currency symbol.
- **Hide Thousands Separators** – Set this boolean to be true to not show thousands separator characters.
- **Decimal Places** – Number of places after the decimal point to show. The default is 2.
- **Minimum Value** – The minimum value that the field should contain.
- **Maximum Value** – The maximum value that the field should contain.
- **Step Size** – If set, the control becomes a spin control with up and down markers. The increment or decrement is the value of this property.

Horizontal Section

*** Horizontal Section



This control is a layout control/section. It allows other controls/sections to be added as children to it. Each child will be laid out to the right of the previous child forming a horizontal row of controls.

Example visuals on a web page:

Default



Round corners



Square corners

A screenshot of a Horizontal Section control. The title bar says "Horizontal Section 1". Inside, there are three items: "Button 1" (blue button), "Checkbox 1" (checkbox icon), and "Button 2" (blue button). The items are aligned to the left by default.

Right aligned

A screenshot of the same Horizontal Section control, but with the "Align Right" configuration option set to true. The items are now aligned to the right.

The Horizontal section can be bound to a list and the content of the section will repeat for each entry in the list.

Configuration:

Name	Default	Description
Show Border	FALSE	If the value is true then a border will be shown around the horizontal section
Square Border Corners	FALSE	If set to true and he borders are shown, then the corners of the border will be square instead of round.
Align Right	FALSE	If set to true then the content of the section will be aligned to the right.

The width of a section can be controlled by setting the control's CSS "width" style to a value. The default appears to be "100%". This can be set in the HTML Attributes section of the control using the style attribute with an explicit width value.

A screenshot of the Properties panel. The "HTML Attributes" tab is selected. In the "Override" section, there is a box containing "style=width: 500px;". In the "Attribute Details" section, "Name: style" and "Value: width: 500px;" are displayed.

Children in the Horizontal Section appear to be "top" aligned. This can sometimes cause layout problems where items appear too high. A solution to this is to over-ride the following CSS Style:

```
.Horizontal_Section > div > div > * {
    vertical-align: middle;
}
```

Here is an example of two <div> elements within a Horizontal Section:

A screenshot of a Horizontal Section control titled "HS1". It contains two square div elements: one yellow and one teal. They are positioned at the top of the section, appearing to be vertically aligned at the top edge.

Notice the padding around them. What is we wanted to remove that? Creating a class called noSpaceHS defined as:

```
.noSpaceHS > div > .BPMSectionBody.LastContentBox, .noSpaceHS > div > div {  
    margin: 0px;  
    border-spacing: 0px;  
}
```

and assigning noSpaceHS as a class to the section will result in:



Image

The Image control is used to add an image onto the page. When seen from the palette, it looks as follows:



When added to the page, it looks like:



The binding property of the Image control is a String which we treat as a URL. The image located at the end of that URL is what will be shown in the Coach. Commonly, we may wish to package the image with the Process Application as a managed file. We can select the managed file from the Select button and the Web Files section:



The selected image is shown within the Process Designer canvas.

This control has the following configuration properties:

Alternate Text	
Height	
Width	
Caption	
Caption Vertical Position	Above or Below
Caption Horizontal Position	Left, Right or Center

Integer

This Coach View allows us to enter an Integer value. From the palette, it looks as follows:



and when added to the Coach, it appears as:

The configuration of this view provides the following properties:

- Hide Thousands Separators – If checked, the thousands separator will not be shown. For example, by default, if "1234" is entered, it will be displayed as "1,234". If this option is checked, then only "1234" will be shown.
- Minimum Value – Defines a minimum value for the field. If a value less than this is entered, the field immediately shows the following:

- Maximum Value – Defines a maximum value for the field. If a value more than this is entered, the field immediately shows the following:

- Step Size – If a step size value is supplied, then up/down arrows are added to the side of the control to change the value. If the up arrow is pressed, the value increases while if the down arrow is pressed, the value decreases. If used in conjunction with minimum and maximum values, the step will not be allowed to break those constraints. When shown on the screen, it looks as follows:

If a validation error is encountered, the Coach View shows the following indication:

To set the size of the Integer control, the following CSS can be used:

```
.em20Integer .dijitNumberTextBox {
```

```
    width: 20em;  
}
```

This CSS can be included in an HTML widget ... eg:

```
<style type="text/css">  
.em20Integer .dijitNumberTextBox {  
    width: 20em;  
}  
</style>
```

When added to a table, the CSS appears to be:

```
.em20Integer .dijitTextBox {  
    width: 20em;  
}
```

Output Text

 Output Text

Output Text

This control displays output text directly in the Coach web page.

Example visuals on a web page:

Output Text 1

The binding for this control can be a String variable. The content of the variable will be used as the content in the Coach. If no variable is bound then the label text can be used for the text shown.

To set the size of the Output Text control, the following CSS can be used:

```
div[data-viewid="viewId"] label {  
    display: inline-block;  
    width: 20em;  
}
```

This CSS can be included in an HTML widget ... eg:

```
<style type="text/css">  
div[data-viewid="myTextId"] label {  
    display: inline-block;  
    width: 20em;  
}  
</style>
```

Radio Buttons

Radio buttons present a series of click-able items that are grouped together. Only one of the items may be selected. Clicking a different item will result in the previously selected item being unselected and the new item being selected. For those who do not know why this user interface style is called "radio buttons", I'd suggest you dial a colleague on the telephone and ask them.

From the palette, the radio buttons control looks as follows:

 Radio Buttons

When added to the page, it looks like



The properties of the Radio Buttons control are:

Property	Description
Selection Service	
Selection List	A list of Strings or objects. Each entry in the list will contribute a new radio button entry.
Selection Service Input Text	
Display Name Property	
Disable Sorting	
Layout	Vertical or Horizontal

The result of showing a radio button Coach View can look as follows:

Transfer Direction
 InBound OutBound Internal

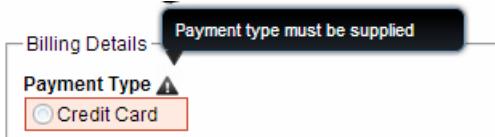
The width of a radio button area (an individual button) can be set to a constant using the following CSS recipe:

- Create a CSS Stylesheet the looks like

```
.radioWidth80 .dojoxMultiSelectItem {
    width: 80px;
}
```

- Add the radioButton80 class to the class list of the radio button Coach View

If a validation error is thrown, it will be shown to the user:



Select

The Select Coach View provides a drop-down widget. This is also known as a Select or Combo Box in HTML/Web programming.

In the palette, the "Select" item looks as follows:

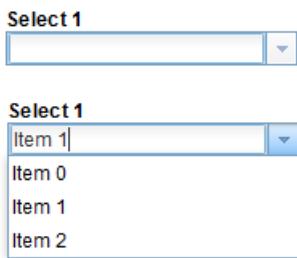


When added to the canvas, the following is shown:



This control allows the user to select a value from a set of possible values.

Example visuals on a web page:

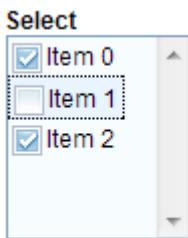


The binding for this control should be a List variable. The content of the list will be used for the content within the combo box. The entries in the list can be simple Strings. Each string will then be a single entry in the combo box. Alternatively, the entries in the list can be business objects. If business objects are used, we need to tell the Coach View which field in the list should be used for display. This can be set with the "Display Name Property". What ever items in the list are selected will also be initially selected in the visuals.

Configuration options:

Name	Default	Description
List Type	Single Selection	<ul style="list-style-type: none">• Single Selection• Multiple Selection
Selected Item		If the list type is Single Selection, then this entry defines which entry was selected.
Selected Items		If the list type is Multiple Selection, then this entry defines which entries are selected.
Selection Service		A service which, when called, will return a list of possible values to be shown as selectable. The signature for this service should be an input of "text" of type String with an output of "results" which should be a List of ANY.
Selection Service Input Text		Input data to be sent to the selection service. This appears as the input parameter called "text". If this property is bound to a variable then when the variable changes its value, the Selection Service is called to generate a new list of values to show.
Display Name Property	Name	The name of a property in the list of Business Objects to be used as the list name.
Value Property	Value	The name of a property in the list of Business Objects to be used as the list value.
Disable Sorting	FALSE	By default, the items in the selection list are displayed sorted alphabetically. This may be different from the order in which the items actually occur in the bound list data type. If we want the items to be shown in the order that they occur in the list, we need to set this property to "true" which disables alphabetic sorting of the items.

If the List Type property is set to be Multiple Selection, the list shows as:



A good data type for binding to this control is the IBM supplied "NameValuePair" where the display property is called "name" and the value property is called "value".

The height of the select area can be controlled with

```
<Root> .dojoCheckedMultiSelectWrapper {
    height: xxx;
}
```

To set the visual width of the Select control, the following CSS can be used:

This CSS can be included in an HTML widget ... eg:

```
<style type="text/css">
.em20Select div.dijitComboBox {
    width: 20em;
}
</style>
```

Select Sample 1 – Selection based on other selection

In this sample, we will discuss how we can achieve the following effect. If the first select chooses the US state of Texas, then the cities will be Texas cities.

State

City

- Fort Worth
- Dallas
- Austin

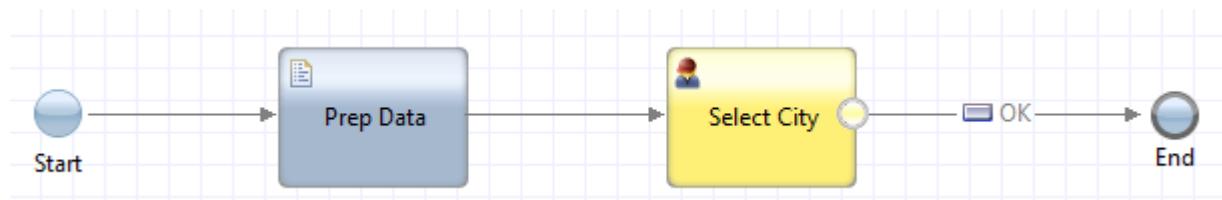
If the user selects California, then the cities will be Californian cities:

State

City

- San Francisco
- Los Angeles

The Human Service that contains the solution looks as follows:



It prepares the static data for states and then shows the Coach.

The Coach looks like:

A screenshot of a user interface titled "Coach". It contains two search input fields: one labeled "State" and another labeled "City", each with a magnifying glass icon. Below these fields is a blue rectangular button labeled "OK".

which is basically two Select controls. One for the State and one for the City.

The State control is bound to a list of Strings which represent the states:

A screenshot of the "Behavior" configuration panel for a select control. It includes fields for "Binding" (set to `listOfStates [] (String)`), "View" (with "Select" and "Coaches" options, and a "Select..." button highlighted with a red arrow), "Label Visibility" (set to "Show"), and "Base Text Direction" (set to "Default").

The State control also has a configuration option as shown:

A screenshot of the "Properties" configuration panel for a select control. The "Configuration" tab is selected. It shows "List Type" set to "Single Selection", "Selected Item" set to `selectedState (String)` (with a "Select..." button highlighted with a red arrow), and other fields for "Selected Items", "Selection Service", and "Selection Service Input Text".

What this means is that when the selection value changes, that value will be stored in the "selectedState" variable.

The second Select, the one called "City" is configured as follows:

Properties Validation Errors Where Used

General

Configuration

Visibility List Type: Single Selection

HTML Attributes

Selected Item: Select... Clear

Selected Items: Select... Clear

Selection Service: Choose Cities

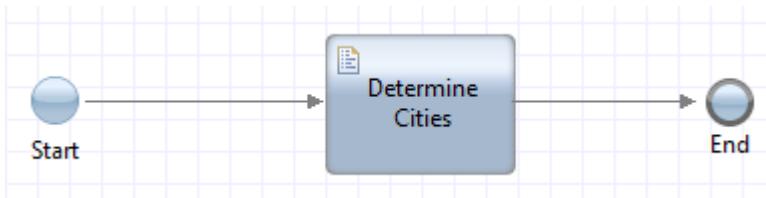
Selection Service Input Text: selectedState

Display Name Property:

Value Property:

Disable Sorting:

It uses a dynamic Selection Service to select the values it will show. The input parameter to the service is the variable "selectedState". What that means is that when "selectedState" changes, the Selection Service will be called to generate new values for the control. Here is the what the "Choose Cities" AJAX service contains:



The variables for this service look as follows:

Variables

- Variables
- Local
- Input
 - text (String)
- Output
 - results (ANY) (List)
- Private
- Exposed Process Variables
- Localization Resources

The input variable called "text" is passed in the "selectedState" value. The "results" variable returns the list of cities. In my sample code, I have hard-coded the following code but it could just as easily have been a database query or some other logic:

```

1 log.info("Choose Cities called with " + tw.local.text);
2 if (tw.local.text == "Texas") {
3     tw.local.results = [ "Fort Worth", "Dallas", "Austin"];
4 } else if (tw.local.text == "California") {
5     tw.local.results = [ "San Francisco", "Los Angeles" ];
6 }

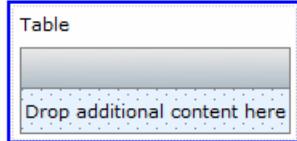
```

Table

From the palette, the table looks as follows:



When dragged and dropped onto the canvas it shows as:



The table control provides a mechanism to show a list of business objects in a row/column format. When the table control is added to the canvas we can then insert further controls as its children. Each of these children will be used to visualize a particular column of data. The control will be repeated in the same column for each row that is displayed. Some common control types that make useful children include:

- Output Text – a cell with a text representation of a value that is not editable
- Text – a cell that displays a text representation of a value which can be edited
- Checkbox – a cell that displays a boolean representation of a value which can be edited

Examples of visuals on a web page:

	Output Text 1	Output Text 2	Output Text 3
<input type="radio"/>	f1-0	f2-0	f3-0
<input type="radio"/>	f1-1	f2-1	f3-1
<input type="radio"/>	f1-2	f2-2	f3-2

	Output Text 1	Output Text 2	Output Text 3
	f1-0	f2-0	f3-0
	f1-1	f2-1	f3-1
	f1-2	f2-2	f3-2

	Output Text 1	Output Text 2	Output Text 3
<input checked="" type="checkbox"/>	f1-0	f2-0	f3-0
<input type="checkbox"/>	f1-1	f2-1	f3-1
<input checked="" type="checkbox"/>	f1-2	f2-2	f3-2

Name	Default	Description
Selection Type	Single Selection	<ul style="list-style-type: none"> • Single Selection • Multiple Selection • No Selection – If selected, no initial column is inserted to allow for selection.
Double Click-to-Edit		
Set Editable Columns		
Show Add Button	no	When selected, a button (+) appears beneath the table which allows us to add a new row into the table.
Show Delete Buttons	no	When selected, a (X) button appears to the right of each row. If clicked, the selected row will be deleted.
Add Button Hover Text	none	Text which is shown in a tooltip when the mouse hovers over the Add Button. Only meaningful if the corresponding "Show Add Button" is also selected.
Delete Button Hover Text	none	Text which is shown in a tooltip when the mouse hovers over the Delete Button. Only meaningful if the corresponding "Show Delete Buttons" is also selected.
Disable Sorting		
Enable Pagination		
Number of Rows Per Page		
Initial Width	700px	Initial width of table
Column Widths	(evenly sized)	Comma separated list of integers
Column Width Unit	%	CSS Units for widths
Hide Columns		

Sizing the table

In the properties of the table there are three attributes that define the size of the table and the columns contained with. The first is "Initial Width". This value defines the size of the table as a whole. Setting this to be 100% causes the table to fill the available space.

The next parameters work together. The first is "Column Width Unit" which is a CSS sizing selector such as "%" or "px". The second is a comma separated list of "Column Widths". There should be a value for each of the columns in the table.

Adding rows

New rows can be added by enabling the "Show Add Button". However, this is only one technique and has some draw backs. Using this mechanism:

- The add button is hard-supplied as a "+" icon
- No initial values can be defined for the new row

By realizing that the table reflects the current list data, any changes made to the list data are immediately reflected back in the table. What this means is that we can provide our own "add row" capability. Imagine adding a button to the page (say beneath the table). The button can be styled in any fashion we choose. When the button is pressed, it can call-back to a server script which can create a new record of data with initial values and add that record into the list of existing records. The server script can then loop back to the original coach. This will have achieved the same function as the default add button but will have provided a wealth of additional options to us.

Disabling selection

It is possible to disable a table from showing anything selected. To do this, we need to copy the Table Coach View and add:

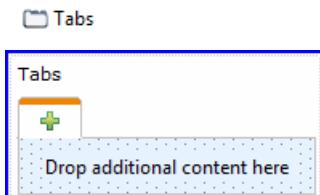
```
selectionMode: 'none'
```

to the construction properties of the DataGrid. In addition, we can switch off the cell highlighting with:

```
.claro .dojoxGridCellFocus {  
    border: 1px solid transparent !important;  
    border-color: transparent #E5DAC8 #E5DAC8 transparent !important;  
}
```

Tabs

The Tabs Coach View is a container for other Coach Views. It allows a page to be displayed where the user can select from a set of Coach Views to be shown at one time. The selection is made via a tab at the top of the control.



The width and height of the tab container is controlled by the style class called "BPMTabControl". Knowing this you can change the size of the container as in:

```
.BPMTabControl {  
    width: 250px;  
    height: 250px;  
}
```

The width can be set to 100% to size the tab container to be the width of the page. As of the time of writing, the height must be sized to be a fixed pixel amount. No auto-sizing capability has yet been determined.

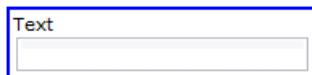
Text

The text field displays a text input box that can be bound to a string. The value of the bound string is shown within the text area. From here, a new value can also be entered.

From the palette, the Text Coach View looks as follows:



When shown on the Canvas, it looks like this:



The configuration properties of the Text Coach View are:

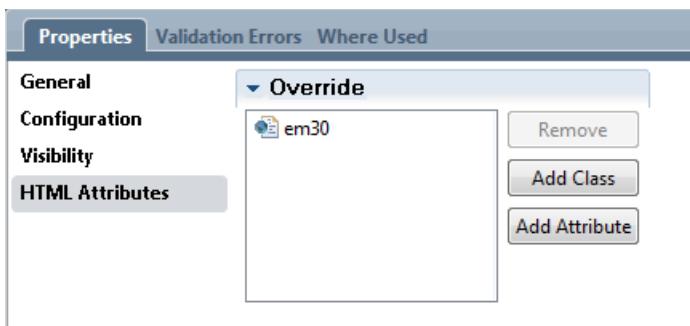
Name	Default	Description
Enable Autocompletion		
Autocompletion Service		

Autocompletion Delay		
Validation	N/A	A JavaScript regular expression that will be applied to the entered data. If the data entered does not match the syntax, a client side error will immediately be shown.

To set the size of the Text control, the following CSS can be used:

```
.em30 div.dijitComboBox {
    width: 30em;
}
```

With the above CSS class defined, we can now add "em30" as a class in the HTML Attributes of the Text Coach View:



This CSS can be included in a custom HTML Coach View ... eg:

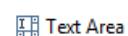
```
<style>
.em30 div.dijitComboBox {
    width: 30em;
}
</style>
```

If a validation error is encountered, the Coach View shows the following indication:

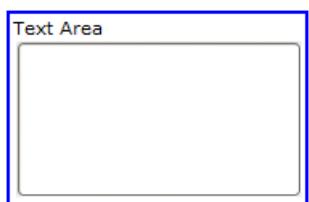


Text Area

The Text Area Coach View shows an area of the screen into which text may be entered. From the palette it looks as follows:



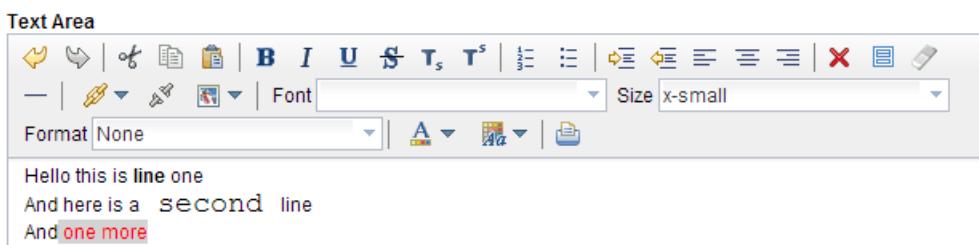
and when added to a Coach it looks like:



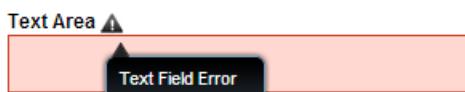
This Coach View has a configuration option called "Text Area Type". This can have one of two possible values either:

- Plain Text – The view is shown as plain text
- Rich Text – The view allows rich text (HTML) to be entered

When in Plain Text mode, the style of the Coach View remains as it is. However if Rich Text mode is enabled, when the Coach View has focus, it changes to show rich text selection capabilities as illustrated in the following image:



If a validation error is encountered, the Coach View shows the following indication in both plain text and rich text modes:



Take care when using this Coach View as it seems to wish to add newline characters at the ends.

To style the body of the plain TextArea, the following CSS can be used:

```
.CV_TextArea .dijitTextArea {
  color:red;
  text-align:center;
  font-weight:bold;
}
```

And add the class CV_TextArea to the Coach View.

Vertical Section

The vertical section control allows us to have multiple coach views placed in a vertical column. On the palette, the control looks as follows:



When dragged and dropped into the layout area, it looks as follow:



The Vertical section can be bound to a list and the content of the section will repeat for each entry in the list.

If the label is "nulled", then the section will show without a visual container.

Name	Default	Description
Show border	FALSE	Adds a border around the body of the container.
Square Border Corners	FALSE	If set to true and the borders are shown, then the corners of the border will be square instead of round.

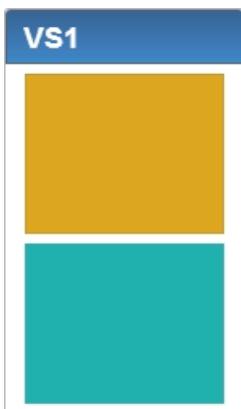
The height of a vertical section can be explicitly set with:

```
.MyClass .BPMBorder {  
    height: 500px;  
}
```

Where "MyClass" is the name of a class added as an HTML attribute to the vertical section.

The width of a vertical section can be set with the width property of the HTML attributes.

Here is a Vertical Section with a couple of colored <div> blocks within it.



See that there is space at border of the contained children. A common request is how to switch off that space. If we were to use a CSS explorer, we would find that a class called "BPMSectionBody" is adding a margin to the right and left. As of 8.5.0.1, this margin is "1em". If we create a class definition that looks like:

```
.noMargin .BPMSectionBody {  
    margin: 0px;  
}
```

and then add the class noMargin to the Vertical section, we would eliminate the left and right space:



In addition, a class called CoachView also adds a top and bottom margin of 0.5em.

```
.noMargin .CoachView {  
    margin: 0px;  
}
```

There is one final style that adds a margin at the bottom of the Vertical Section:

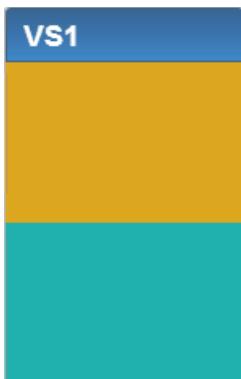
```
.noMargin > div.BPMBorder > .BPMSectionBody.LastContentBox > .CoachView:last-child {  
    margin: 0px;  
}
```

putting it all together we have:

```
.noMargin .BPMSectionBody, .noMargin .CoachView, .noMargin > div.BPMBorder >
```

```
.BPMSectionBody.LastContentBox > .CoachView:last-child {  
    margin: 0px;  
}
```

which will result in:



Content Box

From the palette, the content box looks as follows:



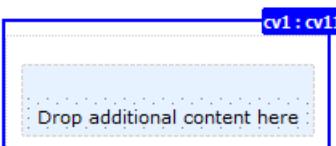
Once added to the canvas, it is shown as:



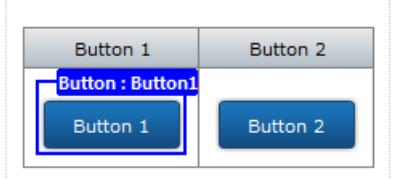
The Content Box is an advanced control that shows up in the Advanced Section of the palette only when building Coach Views. Its purpose is to hold other Coach Views that will be added later by the UI designer.

As an example of using this control, consider a custom horizontal layout style. The Content box would allow us to add arbitrary Coach Views into its space at design time.

Within a Coach where the Coach View that contains the Content Box is added, the Content Box shows up as:



Name	Default	Description
Layout Preview	Vertical	<p>One of:</p> <ul style="list-style-type: none">• Vertical

	<ul style="list-style-type: none"> • Horizontal  <ul style="list-style-type: none"> • Table  <ul style="list-style-type: none"> • One at a time 	
This View will manage its own Content	FALSE	

It is important to understand how a Content Box creates HTML. First, it will insert a `<div>` element with a class of "ContentBox". Its immediate children will be the children added to the Content Box in Process Designer.

For example, imagine we have a Content Box that looks as follows in Process Designer:



As we see, it contains two further Coach Views of type Text.

The inserted HTML looks like:

```
<div class="ContentBox">
  <div class="Text">...</div>
  <div class="Text">...</div>
</div>
```

```
// AMD Mapping:
// dojo/_base/array → array

var cbElement = dojo.query("> *", this.context.element)[0];
var cbContentArray = dojo.query("> *", cbElement);

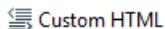
array.forEach(cbContentArray, function(item, I)
{
  // Do something with each item
});
```

Each of the children will have an attribute called "data-viewid" which will correspond to the ControlID or viewID of the Coach View that introduced it.

Custom HTML

Custom HTML allows us to inject raw HTML into the generated Coach shown to the user.

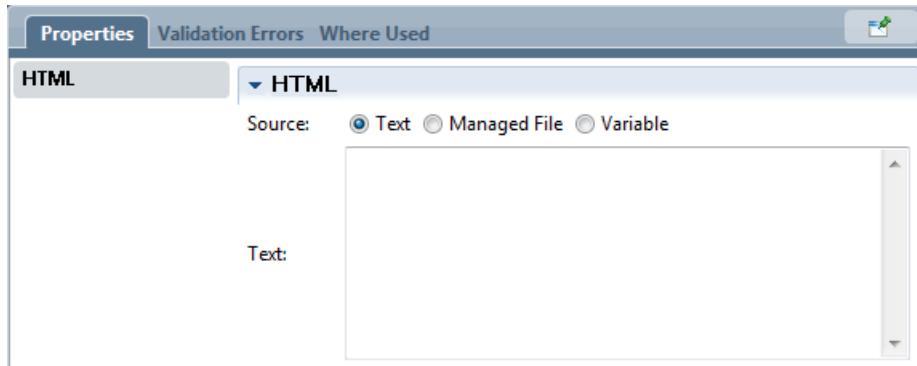
From the palette it looks as follows.



When added into the canvas area, it looks like:



It's properties area contains the following:



The HTML can be entered directly into Process Designer, come from a managed file's content or come from the content of a variable.

For Custom HTML components added to a Coach, the values of variables can be inserted by enclosing them in dual curly braces:

```
"{{tw.local.myvariable}}"
```

If a Coach View contains a Custom HTML component, it can access the `tw.businessData` or `tw.options` scoped variables.

Styling with CSS

The Coach Views supplied with IBM have a certain "look" to them. Since these Coach Views eventually resolve to HTML sent to the browser, we find that the look is defined by standard Cascading Style Sheets (CSS). This also means that the look can be changed by changing the CSS applied.

Looking in the Toolkit that supplies the Coach Views (called Coaches) we find the master style sheet called "`coach_ng_controls.css`". (Note: the "ng" here stood for "Next Generation ... i.e. the post v8 and beyond coaches). It is this style sheet that defines the look. In principle, one could replace or edit this file with new CSS definitions and the result would be a new look. However, I do **not** recommend doing that. This is an IBM supplied file and a patch/new-release could reset the file back to its default and you would have lost your changes. Rather, I suggest overriding this.

Let's look at an example. Here is a basic Coach view that has a Horizontal Section.



We see that it is colored default "blue". What if we want our horizontal sections to have a different look. One of the first things we need to do is determine which styles are affecting the existing look. To do this, I use Google's excellent "Chrome" browser which has in-built development tools. Switching to these, I can select an HTML "part finder" and click somewhere on the page. This takes me to the HTML that renders that part. Once found, I can then see the class names and styles that are being applied to that HTML. Again using Chrome, I can dynamically "tweak" those styles to see what changes I would want to make in a custom style sheet to change the look.

The screenshot shows the Google Chrome Developer Tools interface. At the top, there are tabs for 'Elements', 'Resources', 'Network', 'Sources', 'Timeline', 'Profiles', 'Audits', and 'Console'. The 'Elements' tab is active. In the main pane, the HTML structure is displayed, showing a 'Horizontal Section' header and its content. The CSS panel on the right shows the computed styles for the selected element, specifically focusing on the 'BPMSectionHeader.box.blue' rule. The bottom status bar shows the path: 'html > body > #mainBody > #Horizontal_Section1 > div#Horizontal_Section1_Border > h2.BPMSectionHeader.box.blue'.

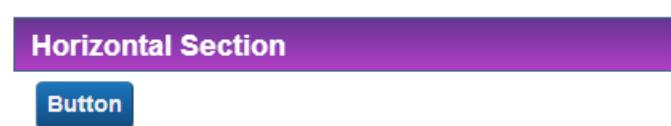
For example, changing:

```
background-image: -webkit-linear-gradient(bottom, #1B76BC 0%, #124B80 100%);
```

to

```
background-image: -webkit-linear-gradient(bottom, #A21BBC 0%, #4B1280 100%);
```

resulted in:



Obviously, this change is not permanent, so now we need to look and see what would be necessary to affect these changes in a broader scope.

Our first pass is to insert a "Custom HTML" control into the page which contains the following:

```
<style>
.BPMSectionHeader.box.blue {
    background-image: -webkit-linear-gradient(bottom, #A21BBC 0%, #4B1280 100%);
}
</style>
```

What this has now done is overridden the original CSS property with the new property. Effectively creating a union of the two CSS styles with the later definitions replacing the same previous definitions.

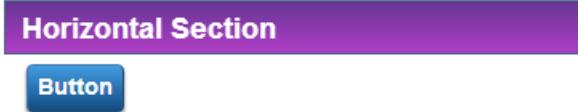
Here is a second example for the button:

```
.BPMButton {
    background-image: -webkit-gradient( linear, left bottom, left top, color-stop(0, #4B1280), color-
stop(1, #A21BBC) )
}
```

which results in:



This looks like we are done ... but not so fast. If we hover over the button we get:



What is happening here is that styles can be qualified. For example, we can have one style for "normal" display and a second style to be shown when the cursor is hovering over the part.

Changing the definition to:

```
.BPMButton, .BPMButton:hover {
    background-image: -webkit-gradient( linear, left bottom, left top, color-stop(0, #4B1280), color-
stop(1, #A21BBC) )
}
```

Changes the style for both normal and hover. Digging further, we find that the button still turns blue when it is held down. Looking at the generated HTML for a button press, we find that the implementation code for the button adds a new class when it is pressed. That class is called "BPMButton-down".

The final styling for our button is:

```
<style>
.BPMSectionHeader.box.blue {
    background-image: -webkit-linear-gradient(bottom, #A21BBC 0%, #4B1280 100%);
}

.BPMButton, .BPMButton:hover, .BPMButton-down {
    background-color: #4B1280;
    background-image: -webkit-gradient( linear, left bottom, left top, color-stop(0, #4B1280), color-
stop(1, #A21BBC) );
}
</style>
```

Great!! ... well ... not so fast. This works great in Chrome ... but will it work in FireFox and Internet Explorer? The answer is not quite. You see, the colors used in IBM BPM are not flat

colors but are instead "gradients" and the CSS for gradients is browser specific. Yuk!! Fortunately, this is such a common problem that there are some elegant gradient generators for us. See for example:

<http://www.colorzilla.com/gradient-editor/>

To see how this works, we use the web page listed above and pick a green gloss:

The screenshot shows the ColorZilla Gradient Editor interface. On the left, under 'Presets', a grid of color swatches has a red arrow pointing to a green one. Below it, a preview bar shows a horizontal green gradient. The 'Name' field is set to 'Green Gloss'. Under 'Stops', there are two entries: 'Opacity' and 'Color', each with a slider and a location percentage input. Under 'Adjustments', there are buttons for 'hue/saturation...' and 'reverse'. On the right, the 'CSS' panel displays a multi-browser compatible CSS background declaration. A large red arrow points from the preview bar towards the CSS code. A tooltip in the CSS panel says 'Current gradient has opacity, switching color format to 'rgba''. At the bottom, there are buttons for 'Comments' and 'IE9 Support (?)', and links for 'import from image' and 'import from css'.

Then we copy the generated CSS into our style. After application (and a few tweaks), we get:



This principle of overriding the default CSS formatting through the use of the Custom HTML Coach View within coaches can then be applied to each of the different controls to achieve a completely different look. So far we have concentrated on backgrounds but every aspect of the appearance is customizable including fonts (type, size, color etc), margins, boundaries, spacings and much more.

Once we have built our styles, we can then save them in a new CSS file and then make that a

managed file. Once it is a managed file, we can then include the same styling in multiple Coaches over and over.

Making Left Labels

When adding text or other fields into a Coach, the label for the field is shown above the input area as shown in the following image:

Without styling

Street	Hello
City	Hello
State	Hello
Zip	Hello

This is fine, but it is a commonly asked request that the labels be shown to the left of the data input as in:

With Styling

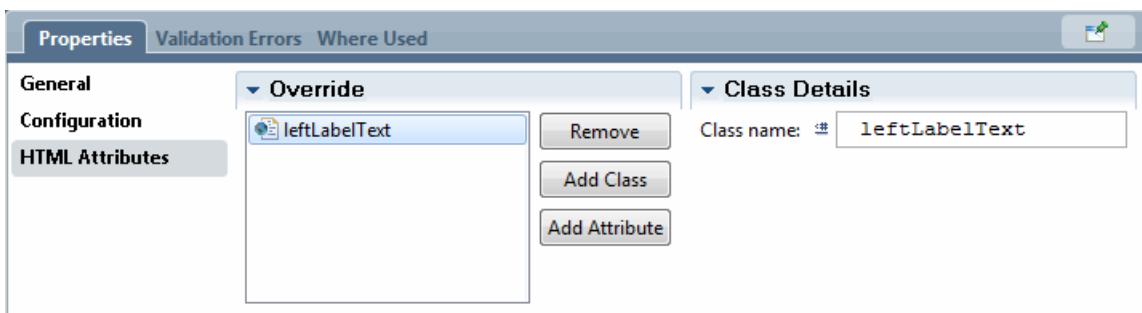
Street:	Hello
City:	Hello
State:	Hello
Zip:	Hello

It was initially thought that this would require modification of the existing Coach Views to achieve but after some study of CSS, it was found that this could be achieved extremely easily. The following style sheet fragment shows how this can be done:

```
<style>
.leftLabelText .textLabel {
    display: inline-block;
    width: 60px;
}

.leftLabelText .content {
    display: inline-block;
}
</style>
```

For the text elements that you wish to have with left labels, add the class called "leftLabelText" to the HTML Attributes:



And that is all there is to it. The styling will be slightly different for the various controls:

Select control
.leftLabelSelect .selectLabel { display: inline-block; width: 60px; } .leftLabelSelect .content { display: inline-block; }
DateTimePicker control
.leftLabelDateTime .textLabel { display: inline-block; width: 60px; } .leftLabelDateTime .BPMDateTimePicker, .leftLabelDateTime .dateTimePickerReadonlyDiv { display: inline-block; }
TextArea control
.leftLabelTextArea .textLabel { display: inline-block; width: 160px; vertical-align: top; } .leftLabelTextArea .dijitTextArea { display: inline-block; width: 30em; }

Adding a TextBox icon

A nice effect is to add an icon into the inside of a text box. For example:



This can be achieved with CSS similar to the following:

```
.dijitInputInner {  
    padding-left: 20px !important;  
    background-image: url('http://img.viralpatel.net/2008/11/magnifying-glass.gif') !important;  
    background-repeat: no-repeat;  
}
```

Styling a Dojo button for BPM

Buttons supplied by IBM Coaches have a particular look and feel, here is the recipe to create such a styling for Dojo buttons:

```
<style>  
.claro .myBPMBUTTON .dijitButton .dijitButtonNode {  
height: 26px;  
color: #f2f2f2;  
background-color: #1b69a9;  
-moz-border-radius: 4px;  
border-radius: 4px;  
text-align: center;  
font-size: 14px;  
font-weight: bold;  
background-repeat: repeat-x;
```

```

padding-bottom: 0px;
border: 1px solid #21486c;
border-top-color: #4178aa;
border-left-color: #4178aa;
border-bottom-color: #21486c;
border-right-color: #21486c;
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#1B76BC',
endColorstr='#124B80');
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr='#1B76BC',
endColorstr='#124B80')";
background-position: 0px 0px;
background-image: linear-gradient(bottom, #124B80 0%, #1B76BC 100%);
background-image: -o-linear-gradient(bottom, #124B80 0%, #1B76BC 100%);
background-image: -moz-linear-gradient(bottom, #124B80 0%, #1B76BC 100%);
background-image: -webkit-linear-gradient(bottom, #124B80 0%, #1B76BC 100%);
background-image: -ms-linear-gradient(bottom, #124B80 0%, #1B76BC 100%);
background-image: -webkit-gradient( linear, left bottom, left top, color-stop(0, #124B80),
color-stop(1, #1B76BC ) );
}

.claro .myBPMBbutton .dijitButton .dijitButtonNode:hover {
    color: #fff;
    border-top-color: #4178aa;
    border-left-color: #4178aa;
    border-bottom-color: #21486c;
    border-right-color: #21486c;
    filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#429CE2',
endColorstr='#124B80');
-ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr='#429CE2',
endColorstr='#124B80')";
background-image: linear-gradient(bottom, #124B80 0%, #429CE2 100%);
background-image: -o-linear-gradient(bottom, #124B80 0%, #429CE2 100%);
background-image: -moz-linear-gradient(bottom, #124B80 0%, #429CE2 100%);
background-image: -webkit-linear-gradient(bottom, #124B80 0%, #429CE2 100%);
background-image: -ms-linear-gradient(bottom, #124B80 0%, #429CE2 100%);
background-image: -webkit-gradient( linear, left bottom, left top, color-stop(0, #124B80),
color-stop(1, #429CE2 ) );
    -moz-box-shadow: 0 2px 3px rgba(0,0,0,.5);
    -webkit-box-shadow: 0 2px 3px rgba(0,0,0,.5);
    box-shadow: 0 2px 3px rgba(0,0,0,.5);
}

.claro .myBPMBbutton .dijitButtonDisabled .dijitButtonNode, .claro .myBPMBbutton
.dijitButtonDisabled .dijitButtonNode:hover {
    color: #a0a0a0;
    background-color: #113f66;
    cursor: not-allowed;
}

.myBPMBbutton .dijitDisabled * {
    cursor: not-allowed;
}
</style>

```

```

.claro .myBPMBbutton .dijitButton .dijitButtonNode[disabled],
.claro .myBPMBbutton .dijitButton .dijitButtonNode[disabled]:hover {
    color: #a0a0a0;
    background-color: #113f66;
    cursor: not-allowed;
}
</style>

```

Data Validation

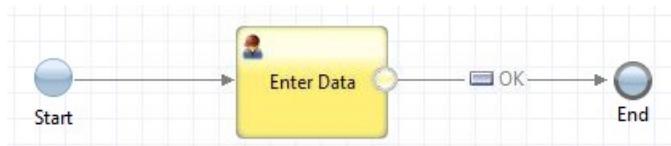
When we present a Coach screen to a user we are expecting that user to enter some new information. Once the user has entered their data it is common for them to click some form of submission button to complete the coach and move on to the next activity. Unfortunately, humans have the ability to enter incorrect information, miss adding necessary information or make other types of mistakes. Ideally we want the coach to examine the data entered and, to the best of its ability, validate that the data entered is logically correct and consistent. Generically, this is known as data validation. Let us consider a trivial Coach that looks as follows:

Name	<input type="text"/>
Amount	<input type="text"/>
<input type="checkbox"/> Approved	
OK	

We can imagine a user working with this screen. Examples of data validation rules for this might be:

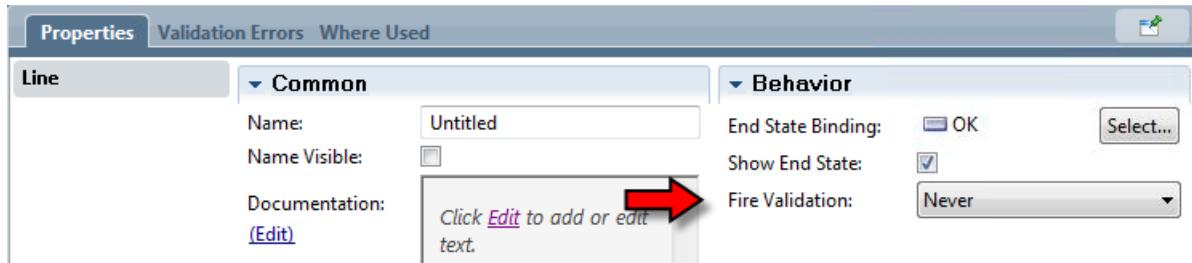
- Name must be entered
- Amount must be entered
- Amount must be positive and greater than zero
- Approved may not be checked if amount is greater than 10000

Looking at our Human Service diagram, the basic flow of a normal Coach mechanism is:



This should be read as the Human Service starts, the Coach is shown and when the OK button is pressed the Human Service completes. At this basic level, there is no validation. Let us now look at how to add our validation functions.

First we define *when* the validation is to be performed. The answer to this (in our story) is when the OK button is pressed. Clicking on the OK labeled link, we see the following properties:



The one of interest to us here is the entry called "Fire Validation". The default value of this is "Never" which means that no validation will occur. Changing this to "Before":

Properties Validation Errors Where Used

Line

Common

Name: Untitled

Name Visible:

Documentation: (Edit)

Click [Edit](#) to add or edit text.

Behavior

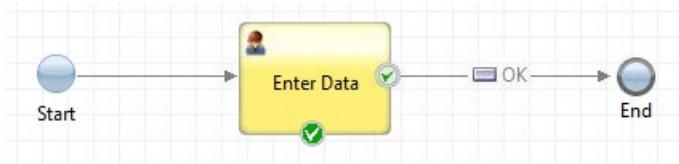
End State Binding: OK Select...

Show End State:

Fire Validation: [Before](#)

informs the solution that when the link is followed, validation is to be performed. Although we have not yet described how to achieve this, at a high level you can now imagine that some how validation will be tested and, if the validation fails, it is as *though* we had not followed the link. Turning that around, the link will only be followed if the validation passes its tests.

After setting the Fire Validation property, the Human Services diagram changes:



Specifically, two new entries can be seen attached to the Coach. The first is that the attachment point on the Coach for the OK link now has a check-mark associated with it. This is a visual indication that this link has validation enabled and associated with it.

The second change is that there is now a new attachment point on the Coach which is shown at the bottom of the Coach. It is a filled green circle with a check mark inside it. This is the core of our story. When the coach recognizes that it needs to perform validation, control is passed through this validation attachment. Attached here will be logic which will perform the data validation rules to ensure that the data entered is correct. When we finally reach the end of job, we will have a Coach which, if rules fail, will visually indicate those errors:

Name

Amount
9,999,999.00

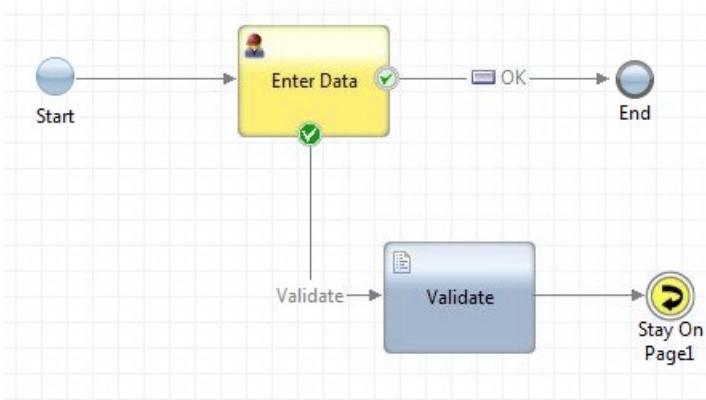
Approved

OK

Now we need to start getting technical and product specific and think about how the mapping between the logic of our rules is made known to the Coach so that it can display correctly.

First we introduce the data structure called "CoachValidation". We should think of this as a container for zero or more validation errors that we detect in our own logic. This data structure is supplied for us by IBM. When we execute our validation logic, we will populate this data structure with any errors we find. If we do not detect any errors, we will not have added any entries to the structure and this will be an indication that the Coach is "good" and that the link that caused the validation may proceed.

Within a Human Service, a singleton variable called "tw.system.coachValidation" is pre-defined. This is an instance of "CoachValidation". It is *this* variable that will be used to communicate between our validation logic and the Coach. With this in mind, we can fill in more of the picture. Our Human Service now looks like:



The way to interpret this is that when the "OK" link is to be traversed, validation is to be performed. Control will now pass through the validation attachment point to a Server Script called "Validate". This will perform the rules necessary for checking our variable values. This Server Script will update the singleton variable called "tw.system.coachValidation" and the return control back to the Coach. The Coach will now check to see if there are any errors indicated and, if not, allow the "OK" link to be followed to completion. If there were indicated errors, the "OK" link will not be followed and the screen will display indications of the errors. This will be the end of the validation until the next time the "OK" link is to be followed at which time the story will be repeated until there are no further validation errors.

Now we can turn our attention to the logic contained within the Server Script:

```

if (tw.local.name == "") {
    tw.system.addCoachValidationError(tw.system.coachValidation, "tw.local.name", "A name must be entered");
}
if (tw.local.amount == undefined || tw.local.amount <= 0) {
    tw.system.addCoachValidationError(tw.system.coachValidation, "tw.local.amount", "Invalid number");
}
if (tw.local.approved == true && tw.local.amount > 10000) {
    tw.system.addCoachValidationError(tw.system.coachValidation, "tw.local.approved", "Too big to approve");
}

```

Within the script, we perform data validation rules. If a rule fails, we then execute the method called:

```
tw.system.addCoachValidationError
```

This method takes three parameters:

Parameter	Description
coachValidation	An instance of a CoachValidation object which acts as a container of discovered errors. This is likely to be the predefined tw.system.coachValidation object.
variable_path	The path to a variable that has failed the validation. It is important to note that the string supplied here is the name of the variable (as known to the Coach) which is in error. We are not validating individual Coach Views/controls in the Coach ... rather we are validating data. What this means is that when we encounter an error, we need to tell the coach which "data" is problematic and the Coach itself then determines which controls need to be updated to show the error.
message	A text message that can be shown to the user to explain the reason for the failure.

Executing this method adds a new validation error entry to the CoachValidation container. Remember that the singleton variable called "tw.system.coachValidation" is the variable

that must eventually contain the errors for the Coach to work with them. It appears that each time the Validation attachment is fired, this variable is initialized to an empty set of errors (which is what we want).

From the previous description, we see that a validation service is associated with a Coach however it is a single validation service. We may want to execute different validations depending upon which button was pressed. One way to achieve this is to bind a Button control to a boolean variable. When the button is pressed, the associated variable is set to true. This can then be used as a distinction expression to determine which validations to perform.

Boundary Events

Within a Human Service a Coach can loosely be thought of as a screen or web page. When the user interacts with the Coach there will come a time when the user would like to end their interaction with the web page and pass control back to the server. This will logically mean the "completion" of that Coach. To flag this, the Coach generates what it calls a "boundary event". This is an odd name but I can't think of anything better. It flags the exit from the current Coach and the entry into the next activity in the Human Service diagram.

By default, Button Coach Views within a Coach will generate a Boundary Event when they are pressed. For each boundary event added to a Coach, the result will be the ability to wire from the Coach to a subsequent activity within the Human Services definitions. Initially, when a new Coach has been created, there will be no Coach Views within it. This means that until Coach Views which generate boundary events are added, the Coach can not be wired as the source of a link in the Human Services editor.

See also:

- context.trigger(callback)

Dynamically controlling visibility based on user

Note: As of v8.5, the visibility settings of Coach Views have been dramatically updated.

A recurring pattern that keeps appearing is the desire to show a Coach where the fields in the coach are editable or not editable based upon the group a user belongs to.

For example, imagine a group called "managersGroup". If the user is a member of this group, he can edit the a field.

Managers Comments

I approve this message

OK

If he is not a member of this group, he will be disallowed and any attempt to edit the field will be prevented.

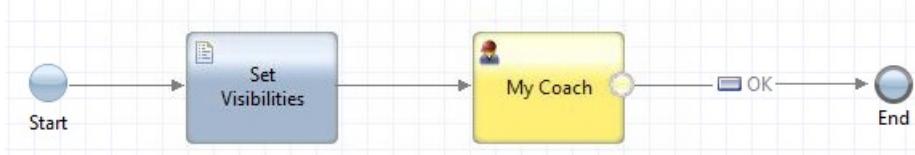
Managers Comments

No comments

OK

If we remember that a field has a visibility attribute and that attribute can be set to "EDITABLE" or "READONLY" then we can build some logic that determines whether or not the current user is a

member of the "managersGroup". If yes, we can set a variable to be "EDITABLE" and if not, set the same variable to be "READONLY". If we then bind the visibility attribute of the control to this variable, at run-time, the control will behave appropriately.



Here is the code for the "Set Visibilities" script:

```

tw.local.commentsVisibility = "READONLY";
var user = tw.system.user;
if (user == null) {
    log.info("User is null!");
} else {
    if (user.roles == null) {
        log.info("No roles!");
    } else {
        for (var i=0; i<user.roles.length; i++) {
            if (user.roles[i].name == "managersGroup") {
                tw.local.commentsVisibility = "EDITABLE";
                break;
            }
        }
    }
}
  
```

The TextArea coach view is then bound to the "commentsVisibility" variable:



and we have achieved the effect we were looking for. This recipe can be repeated for different fields. A field can also be set to "NONE" to hide it from view so we can also hide a control from visibility. Note that this should not be used for security as the data is still present at the Coach but simply not shown.

IBM sample Coach Views

On the IBM BPM Wiki there are a ton of additional Coach Views that are supplied "as-is". What this means is that the Coach Views can be freely downloaded and used in your own BPM solutions but if something goes wrong with them, you are basically on your own. Because of this, the Coach Views are provided with full source and the onus is upon you to read and review that source and treat the Coach View as if you had written it yourself. If you come back to the provider of the Coach View and are looking for support, there is no assurance that you will get any. This section of the book provides some examples of using some of these Coach Views for different purposes.

Data Change Boundary Trigger

The [Data Change Boundary Trigger](#) is a non-visual Coach View. It monitors data variables and when a variable changes, fires a boundary event.

See also:

- Boundary Events

Data Change Boundary Trigger - Updating one list when another changes

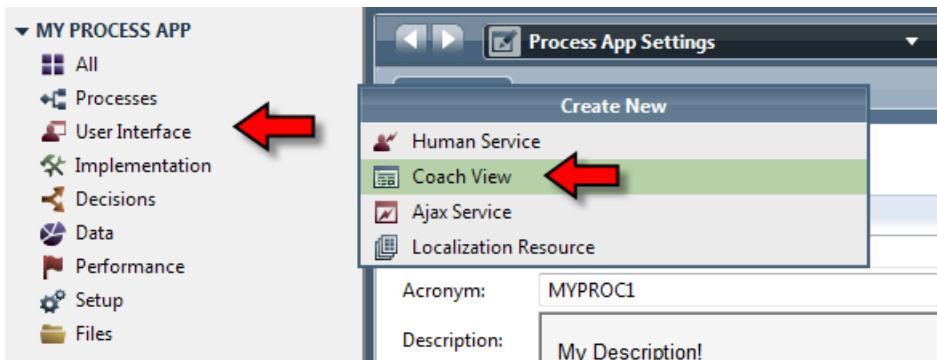
Imagine we have two lists on the screen. When the selected entry in one list changes, we want the content of the second list also to change. The Data Change Boundary Trigger can help to make that happen. What we do is insert a Data Change Boundary Trigger into the coach and bind it to the data for the first list "listSelected" property. When the first list changes its selection, a boundary trigger now fires. In the Human Service logic, we can now determine what the new selection of the list is and change the content of the second list correspondingly. A sample called "KolbanTK – Test – Data Change Boundary Trigger – List Changes" is supplied with the toolkit.

Building new Coach View types

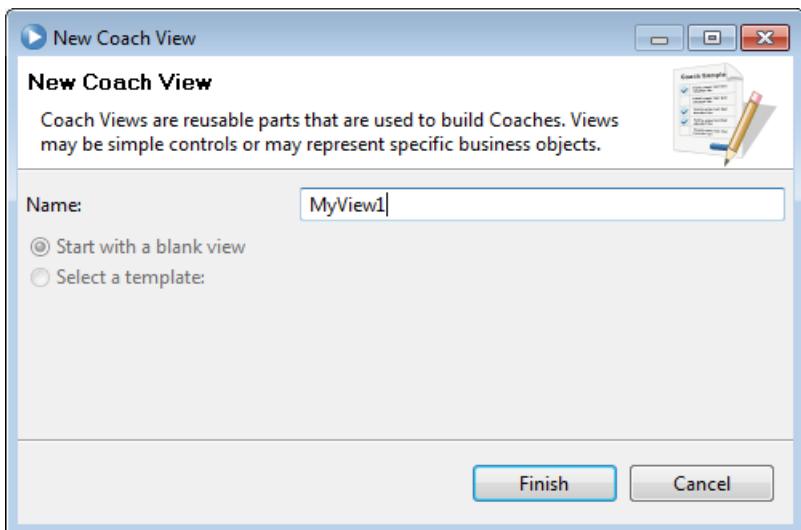
The Coach Views supplied out of the box by the product are certainly not the only Coach Views available. Coach Views can be obtained from IBPM community web sites as-is or you can build your own Coach Views. When building your own, it is important to understand the architecture associated with Coach Views. We will now look at that architecture in more detail.

The high level notion behind a Coach View is that it will have a visual appearance within a Coach in order to present data to the user, solicit data from the user or improve the look and feel of the Coach page with visual decoration or layout. The Coach View achieves all of this by a combination of technologies including HTML and CSS which is interpreted by the browser, JavaScript for logic inclusion and Dojo (or other UI toolkit) for visual representation.

A new Coach View can be created from the User Interface menu entry and selecting Coach View:



A name must be supplied for the new Coach View. The name entered must be a valid JavaScript identifier.



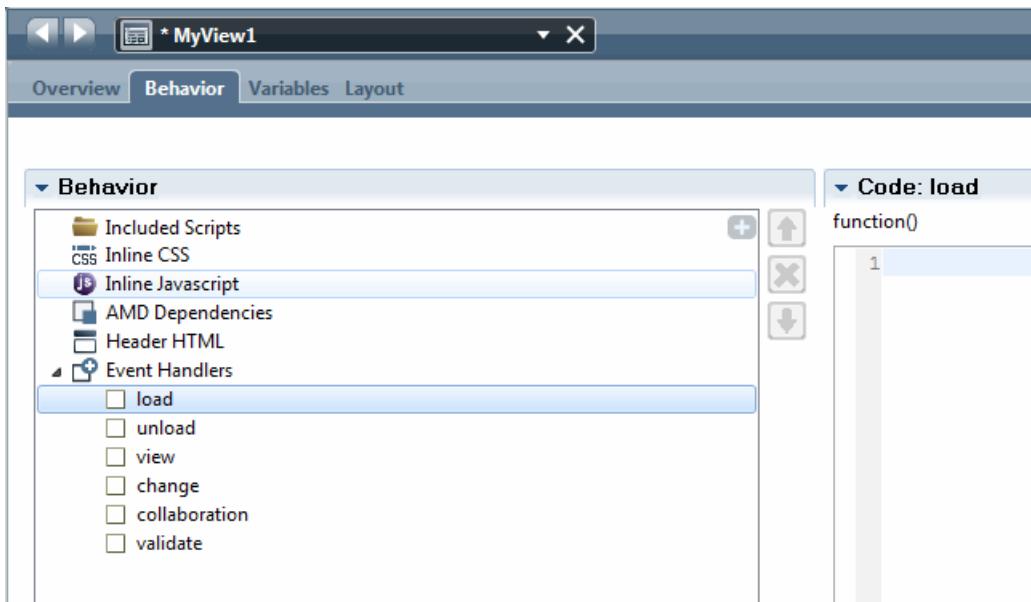
Once created, the Coach View editor page is shown. This consists of a number of separate tab pages. On the Overview page we have some general settings:

The meanings of the attributes available to be changed are shown in the following table:

Tags	The tags associated with the Coach View are extremely important. The tags define which sections within the palette in the Coach Designer editor screen the new Coach View will appear. If no tag is defined, the Coach View will simply not be shown and hence not be usable.
Palette Icon	The graphic image to be used as the icon in the palette area. If no image is supplied a default image that looks like '◆'. IBM has chosen icons 16x16 pixels wide and hence it is suggested that custom Palette icons be the same size. If images other than this size are supplied, they will be re-sized.
Layout Image	The graphic image to be used as the icon in the coach composition area
Use URL binding	If the data binding type for the coach is "URL" then use the data returned by that URL

	as the preview image in the layout. There are very few potential users for this. The only one that seems to make sense is the Image control.
Preview Label Position	<ul style="list-style-type: none"> • Top • Right • Bottom • Left • Center (stretch image to fit) • Header
Can Fire a Boundary Event	Set to true if this control can act as a boundary event source. If checked, then a wire can be drawn from the Coach in the Human Service.
Use as a Template	
Supports a Label	This flag declares whether or not the label should be used within the layout.
Prototype-level event handlers	The documentation on this boolean option is quite vague. It claims that it is related to a performance optimization but no indication on the improvement gained is supplied. It appears that the event handlers for the various Coach View events (load, change etc) are internally defined as either function on the Coach View object or as prototype members on the Coach View object. By default, they are functions. By changing to prototype by checking this box, performance gains are said to appear. The documentation seems to then describe the use of attaching variables to the object instance as opposed to them existing "in context".

The next page of interest to us in the Behavior page.



This has sections for

Included Scripts	These are JavaScript files defined as managed files which can be included in the page.
Inline CSS	This section allows us to define CSS definitions.
Inline JavaScript	This allows us to add a fragment of JavaScript. Since it is not defined when this script will be run, it is best to assume it will be run sometime "before" any of the other callback methods and be run once only. The context of the JavaScript is the view object. One use of this function is to perform any one-time initialization such as defining functions and hanging them off the "this" object.
AMD Dependencies	Names of AMD includable modules and the aliases associated with them. See below
Header HTML	This is HTML that will be placed in the header of the page inside the HTML <head> tags.

Event Handlers

A set of JavaScript callback functions.

AMD Dependencies

Dojo has switched to using the Asynchronous Module Dependencies (AMD) technology to load JavaScript modules into pages. In the AMD Dependencies section, we can name modules and an alias JavaScript variable that will be used to hold a reference to the entry point into that module.

Asynchronous Module Definition (AMD) dependencies

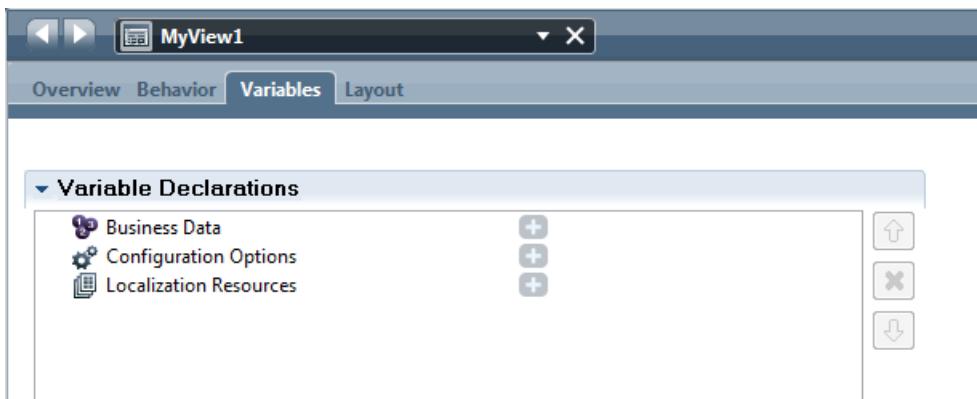
Register the ID and alias for each module that you want to load dynamically:

Module ID	Alias	
dojox/grid/DataGrid	dataGrid	Add Remove Up Down

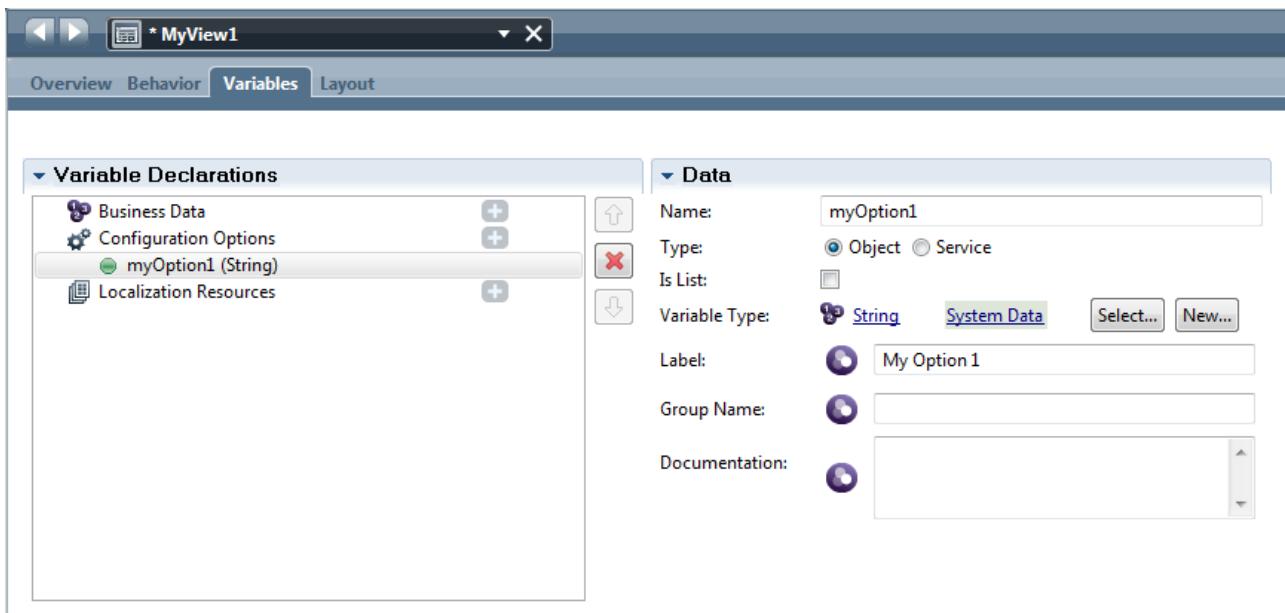
See also:

- [Asynchronous Modules Come to Dojo 1.6](#)
- [The Dojo Loader](#)
- [Dojo FAQ: What is the map config option? - Sitepen - 2013-07-03](#)
- [Dojo FAQ: What is the difference between a package, a path and an alias? - Sitepen - 2013-06-20](#)

The Variables tab allows us to define data associated with this Coach View.



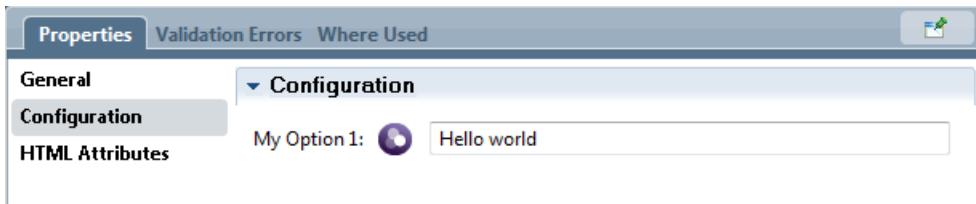
Amongst these options we will find an entry for defining Configuration Options:



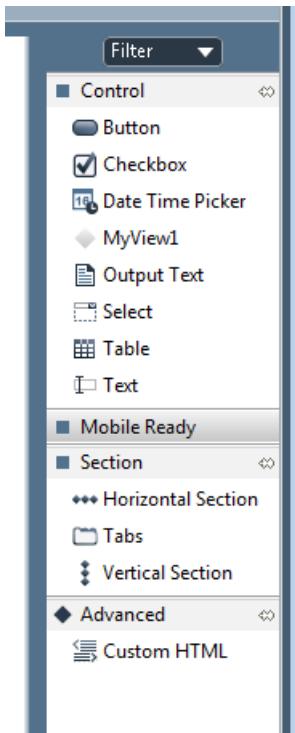
Each configuration option has a variety of properties:

Name	Description
Name	The name of the configuration option as referenced in the code by <code>this.context.options.<name></code>
Type	<ul style="list-style-type: none"> Object Service
Is List	A boolean to flag this configuration option as being a list/array of values or a singleton.
Variable Type	The data type of this variable. The variable type has a direct bearing on how the configuration property is shown. For example, a Simple Type → Selection will result in a pull-down with the options defined in the selection being the only options.
Label	The label to be shown in Process Designer in the Coach designer beside this configuration option
Group Name	Here we can supply the name for a "group". Grouped entries will show up in the Coach editor under the same heading.
Documentation	Documentation for readability

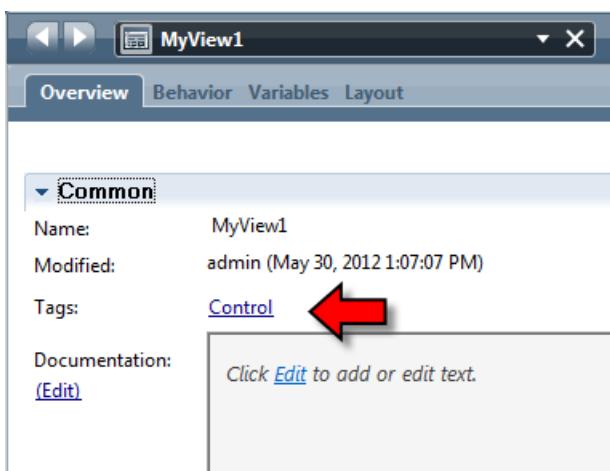
Here is an example of what a configuration option entry would look like in Process Designer:



Once a Coach View has been created, we can see it in the palette in the Coach Designer.



The palette is split into groups. When a new custom Coach View is created, it must be tagged as belonging to one or more of these groups.



If it is not tagged then the Coach View will show up in a section called "No Tags".

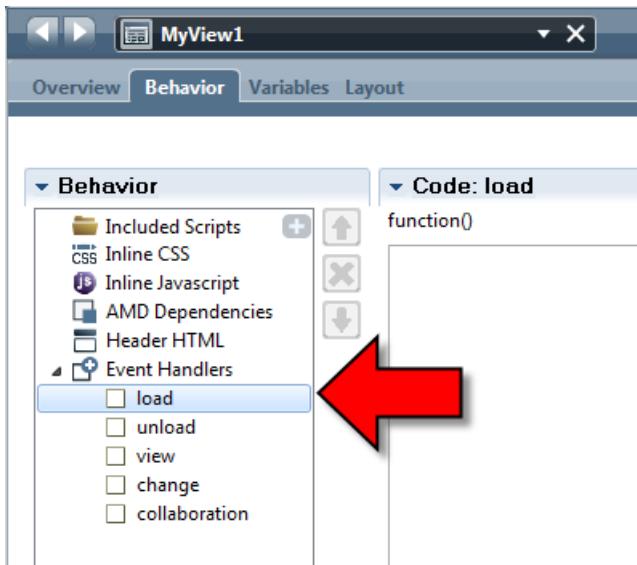
Coach View Event Handlers

During the life of a Coach, each Coach View will be sent certain events over time. When these events arrive, a Coach View can respond to these events in different ways by invoking custom code. When an event happens, a callback function provided by the Coach View can be executed. Six event types are defined by the Coach View architecture. These are:

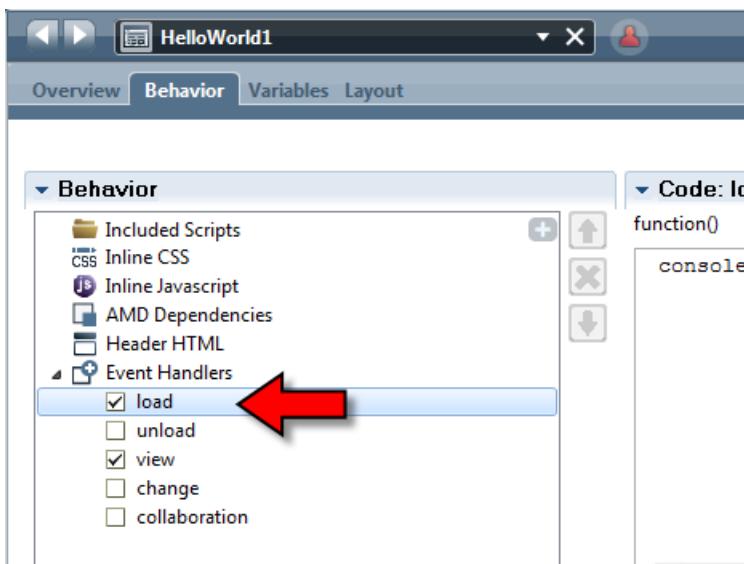
- load
- unload
- view
- change

- collaboration
- validate

Within the Behavior tab of the Coach View editor, we see a section called Event Handlers. Within there are entries for each of the six different types of event handler. Selecting any one of them gives us an editor area into which we can insert JavaScript code that will be executed when that event occurs.



When code is associated with an event handler, the event handler appears with a check-box mark to indicate that it will "do something":



Load event handler

The load function is called when the Coach View is initially loaded. This can be used to perform initialization. Experience has shown that this is the most prevalently used handler in Coach View construction.

View event handler

The view function is called just before the Coach View is to be displayed.

Unload event handler

The unload function is called before the Coach View is disposed and allows it to perform any final saving or cleanup of resources.

Change event handler

The change event handler is called when data associated with the Coach View changes. This data is commonly either the single data item bound to the Coach or any of the data items set in the Configuration section.

The data structure is available through the variable called "event" which is passed into the event handler. This data structure contains the following properties:

Property name	Data type	Description
type	String	An event can occur because of either a change in binding or a change in configuration. If the change occurred because of a configuration change then the type property will have the value "config".
property	String	The name of the property that changed. Note that many of IBM's configuration properties appear to show up as "_metadata.visibility". There is a belief that this is not an object called "_metadata" with a property called "visibility" but is instead a property actually called "_metadata.visibility".
oldVal	N/A	The bound value before the change
newVal	N/A	The bound value after the change
insertedInto	Integer	
removedFrom	Integer	

Validate event handler

This event handler is called when a validation service has been called. The input parameter to the event is either an error or clear object. An error object is used to indicate that the Coach View should show an error while a clear object is used to indicate that a previous error indication is to be removed.

An "error" object has a property called "errors" that is an array of records containing:

Property Name	Data Type	Description
binding	String	Path to the variable in error.
message	String	A description of the error.
view_dom_ids[]	String[]	

A "clear" object has no properties and is used simply to signal that there are no longer any errors.

How a Coach View should indicate to the user that a validation error has been encountered is a decision left to the designer of that Coach View. It is probably wise to follow the same styles as used by IBM for their supplied views. If we look at the Text view as an illustration of how IBM approached the task, we find that the HTML template for the view contains the following:

```
<div class="coachValidationDiv">
    <img class="coachValidationImg CoachView_hidden" role="alert"/>
```

```
<span class="coachValidationText" style="visibility: hidden;">!</span>
</div>
```

This basically means that we have an area of the control into which error information can be shown. We see it has two tags. An image tag to show an icon and a span tag into which the error message will be shown.

The high level logic for view update is if the type of the event is "error"

Coach View Context

From a programming perspective, the Coach View context is probably the single most important notion. It provides access to a broad variety of information. When JavaScript executes within the browser environment, that JavaScript code can access an object called "`this.context`" which is a reference to the Coach View context.

We will start by looking at the functions provided by an instance of object.

`context.bindingType()`

A function which returns the data type of the bound variable.

`context.bpm.system`

This variable holds a large number of properties that are useful when building Coach Views.

● <code>user_id</code> - String
● <code>user_loginName</code> - String
● <code>user_fullName</code> - String
● <code>user_localeCountry</code> - String
● <code>user_localeLanguage</code> - String
● <code>paShortName</code> - String
● <code>snapshotId</code> - String
● <code>taskId</code> - String
● <code>baseTextDirection</code> - String
● <code>instanceId</code> - String
● <code>pald</code> - String

`context.bpm.team`

● <code>manager</code> - *BPMTeamMemberManagerNamespace
● <code>member</code> - *BPMTeamMemberManagerNamespace

This item lists the teams that the user is a member of and which teams he is a manager of. The values are arrays. The objects contains within the arrays contain:

- `name` – The name of the team.
- `tkShortName` – The acronym of the toolkit that the team belongs to. If the value is null, then the team belongs to the current process app.
- `id` – A unique id for the team.

context.broadcastMessage()

context.cancelBoundaryEvents()

context.containDiffLoopingContext()

context.createView(domNode, index, parentView)

context.deleteView(domNode)

context.element

This is the DOM element representing the root of the view in the DOM tree.

context.getDomNode()

context.getInheritedVisibility()

context.getSubview(viewId, requiredOrder)

Get the child Subview with the identity of "viewId". Only one level deep is examined.

context.htmlEscape(<string>)

A function which escapes HTML content and makes it a simple text string.

context.isTrustedSite()

context.parentView()

Returns a reference to the object describing the parent Coach View. If the current Coach View is already on the Coach page itself then it has no parent and null is returned. This method should not be called in the load() function as the environment has not yet been fully built.

context.refreshView()

context.setDisplay(isDisplay, domNode)

This is used to set the visibility of a DOM node. The `isDisplay` parameter is a boolean. The value `true` will show the node while `false` will hide it. The `domNode` is optional. If not supplied, the root of the DOM tree for the view will be used.

context.setUniqueId()

context.subscribeValidation()

context.subview[viewid]

Retrieve the JavaScript object representing the Coach View instance of the immediate child called "viewId". The viewid parameter is the Control Id supplied in the Process Designer editor. Be careful with the objects returned. You really aren't "meant" to do anything with them. About the best you can do is enumerate them to find the viewIds and then call getSubview() to get the actual object.

The following will search the subviews looking for a named view:

```
function findView(context, viewId) {  
    for (var childName in context.subview) {  
        if (childName == viewId) {  
            return context.getSubview(viewId)[0];  
        }  
        var result = findView(context.getSubview(childName)[0].context, viewId);  
        if (result != null) {  
            return result;  
        }  
    }  
    return null;  
}
```

See also:

- Navigating and access to Coach Views on a Coach page

context.trigger(callback)

When invoked, this method produces the publication of a boundary event. The *callback* function is optional. If a callback function is provided, it will be invoked with no parameters. The callback occurs when the coach regains control after the asynchronous processing of the event. If we wish to use this API, ensure that the check-box called "Can fire a boundary event" is checked.

See also:

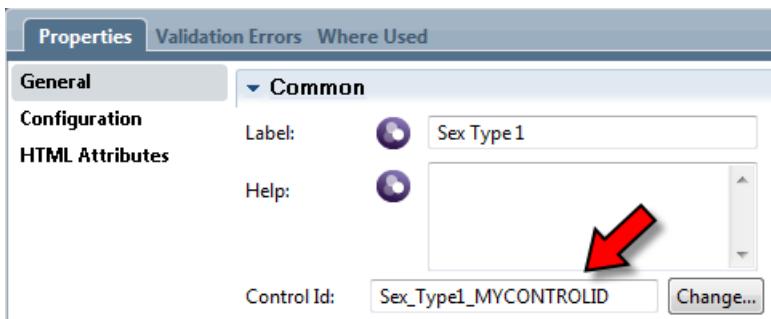
- Boundary Events

context.triggerCollaboration(payload)

context.unsubscribeValidation()

context.viewid

This is the value of the Control Id in the properties view of the Coach Designer where the Coach View is defined:



This can be used as a key value in a number of algorithms. Take care with tables. It appears that the Control Id/context.viewid may be the same value in each row.

Coach View global data

Within a Coach View, we can also access information about other Coach Views also present in the window. The variable `com_ibm_bpm_global` provides access to this.

`com_ibm_bpm_global.topLevelCoachViews`

This is a property which is an object (**not** a list) which has a property corresponding to each of the Coach Views on the top level canvas. A little extra care appears to be needed when using this variable. It appears that when access from a Coach View's `load()` function, the variable has **not** yet been fully populated. This means that even though other Coach Views may be found on the page, they have not yet been added as properties of the page. It appears that by the time the `view()` method is called, the variable is properly and fully populated.

The following will list all the top level Coach Views:

```
var x = com_ibm_bpm_global.topLevelCoachViews;
for (var prop in x) {
    if (x.hasOwnProperty(prop) == true) {
        console.log("Property: " + prop);
    }
}
```

See also:

- Navigating and access to Coach Views on a Coach page

`com_ibm_bpm_global.coachView.byIdControlId()`

`com_ibm_bpm_global.coachView.byIdDomId()`

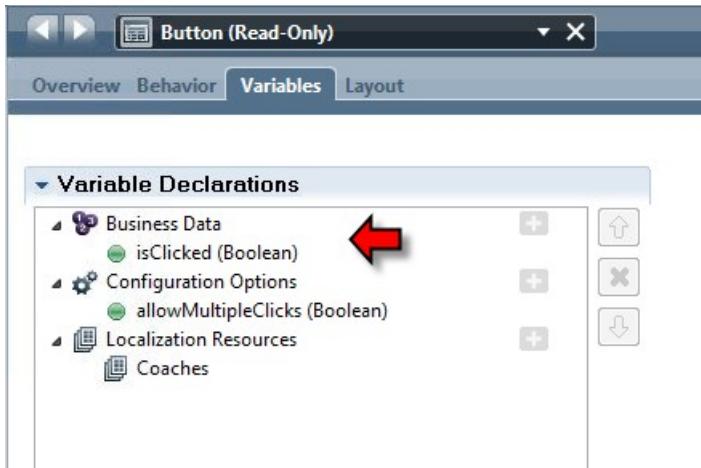
Coach View data binding and configuration

When a Coach View is added to a Coach, that Coach View can be associated with a piece of data defined as a variable in the Human Service. The association of the coach view to a variable is called data binding. It is highly likely that data binding will be used extensively with most custom Coach Views ... after all, a Coach View that doesn't display some configurable data probably isn't that useful.

When a Coach View is displayed in the browser, a *copy* of the value of that variable is brought from the server and stored as data associated with the Coach View. It is this copy that is used with getters and setters when the Coach View code accesses the data. When a boundary event trigger occurs, the current value of the variable *in the Coach View* is then sent back to the server and stored

in the original variable from which the copy was taken in the first place. This is key. If one uses a Coach View, changes some data and then closes the browser window without first submitting a boundary trigger, the data is not sent back to the run-time environment. In addition, once a Coach View is shown, should there be any changes (somehow) to the values of the variables in the Human Service in the run-time, these changes will not be reflected in the Coach View as it will already have loaded a copy of the original values when it was displayed.

The declaration that a coach view can be bound to data us defined in the Business Data entry of the Variables section:



The value of the data can be obtained from within the code of a Coach View with the call:

```
var myValue = this.context.binding.get("value");
```

Note that the "value" parameter is a special keyword and is the actual phrase "value".

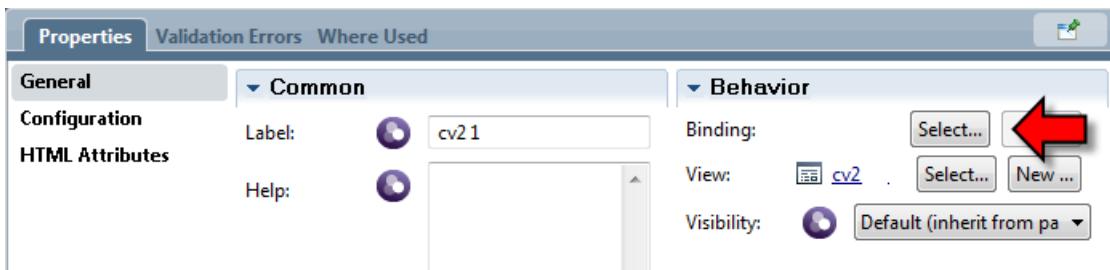
When the value of a variable that is bound to a view changes, the view's `change()` event handler function is invoked to inform it of the change. This is true for simple values such as String and Integer. For Business Objects, the view is notified only if the Business Object itself changes and not if any of its contained properties change. To bind to lower level of a Business Object, we can use the `bindAll` function.

There is also a `bind()` function that can name a particular property in the object.

The `context.binding` JavaScript object has the following properties:

Name	Description
property	The name of the variable
get	A function to get the value
set	A function to set the value

Note that a binding does **not** have to be applied to a Coach View. In the following, no binding has been provided:



What this means is that the `context.binding` property is not defined. The repercussion of this is that an attempt to access a `get()` method through this undefined variable/property will cause a failure. As such, it is essential to always check that `context.binding` is actually defined before attempting to use it. Here is an example:

```
if (this.context.binding != undefined)
{
    ...
}
```

Working with bound list values provides yet another challenge. A change to an entry in the list if not the same as a change to the *list* itself and hence no change event is fired. If "myList" is a list variable, then we can provide a callback to be called if an element within the list is updated by using:

```
var handle = myList.bindAll(function(param), scope);
```

The "scope" parameter is an optional scope in which the callback function will be executed. The "param" is a JavaScript object that looks as follows:

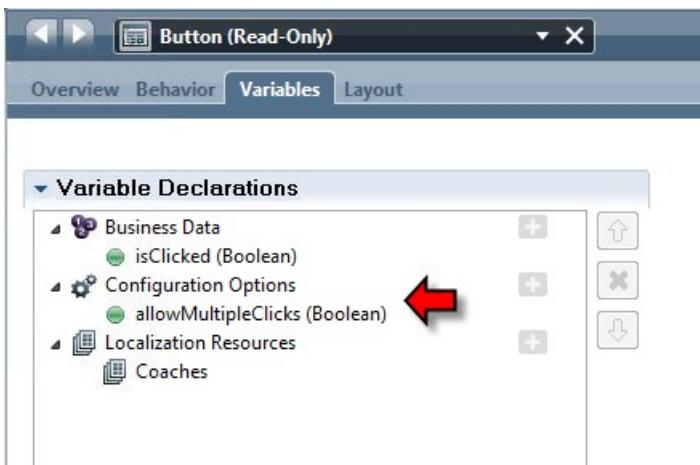
- `newVal` – For an insert, the new value being inserted
- `oldVal` – For a removal, the old value being removed
- `insertedInto` – The list index into which the item is being inserted for an insert. If no insert or removal has occurred, this property will not be present.
- `removedFrom` – The list index from which the item is being removed for a removal

When using binding, one should clean up the binding when finished by calling

```
handle.unbind();
```

where `handle` is the handle returned from a bind call.

In addition to the binding data for variables, a second set of data is also available. Configuration options are options set on the Coach View during development. These are set in the Configuration Options section. Multiple configuration options can be added.



Within the code of the Coach View, the general form for getting a configuration property is:

```
this.context.options.myOption.get("value");
```

and for setting a configuration property is:

```
this.context.options.myOption.set("value", newValue);
```

The IBM BPM product provides a number of predefined options that are commonly present for all coach views. These include:

Option	Data type	Description
_metadata.label	String	The value of the label if one has been provided.
_metadata.visibility	String	<ul style="list-style-type: none"> • DEFAULT • EDITABLE • REQUIRED • READONLY • NONE • HIDDEN
_metadata.labelVisibility	String	<ul style="list-style-type: none"> • SHOW • HIDE
_metadata.helpText	String	
_metadata.className	String	
_metadata.htmlOverrides	Map	

Data types found in IBM BPM are mapped to data types in JavaScript found in the Coach browser environment

Server data type	Browser data type
Date/TWDate	JavaScript Date

Working with Coach View list data

If a variable is bound to a list, the story changes somewhat. Using the `get("value")` method returns an object that *represents* the list. This object has properties and methods upon it which can be used to work with the content of that list. Assume that the variable "list" in the following examples is what is returned from a call to `this.context.binding.get("value")` of a list bound variable. Do **not** think of the list object as a JavaScript array. Rather, think of it is a

logical container for indexed objects. Do **not** assume you can perform arbitrary JavaScript like list work against it.

When working with index values, "0" is considered the first element in the list.

Function	Description
list.add(item)	Adds a new item at the end of the list.
list.remove(index)	Removes an item from the list at the specified position.
list.put(index, item)	Replaces the item at the specified item in the list. The index must already exist.
list.length()	Returns the current length of the list.
list.get(index)	Return the item at the index into the list
list.items	This is a property of the list object that returns a JavaScript array of the items in the list. The content of the list should not be modified.
list.createListElement()	This is an undocumented item that <i>appears</i> to create a new list element but does not add it to the list.

Example: Remove all elements from a list

```
var myList = this.context.options myList.get("value");
while(myList.length() > 0) {
    myList.remove(0);
}
```

In addition to the content of the list, there is also the concept of the selected items in the list. These can be accessed through "gettable" properties of the list object. Make sure that you realize that these are properties of the list object and not of the binding itself.

Property	Type	Description
listAllSelectedIndices	Array of integers	An array of integer values. Each integer corresponds to the index of an item in the list which is considered selected.
listAllSelected	Array of Objects (read only)	An array of objects. Each object corresponds to an object in the list that is considered selected.
listSelectedIndex	Integer (read only)	The index of the <i>first</i> selected item in the list.
listSelected	Object (read only)	The object that is the <i>first</i> selected item in the list.

All of the above properties can be read but only the `listAllSelectedIndices` property may be changed. Setting this to a new array results in a new concept of selected items in the list.

```
myList.set("listAllSelectedIndices", [<array of index values>]);
```

If we want to know if these properties change, use the "bind" function. For example:

```
myList.bind("listAllSelectedIndices", function() {....});
```

Data Store

The Dojo toolkit has the concept of a "Store". This is a generic container for a list of data. The store concept provides an abstraction to sourcing and sinking data. Since the store has an architected specification, a program should be able to work on the store without knowledge of where the data came from or where it is going. The list data returned in a Coach View also has a "store" associated with it. It can be accessed with the property "dataStore".

Before we get too excited, as of 2013/02, this feature is not documented and hence may not be supported.

Dis-allowed Coach Binding data types

Not all IBM BPM data types may be bound to Coach Views, the following are data types that are prevented:

- XMLDocument
- XMLElement
- Map
- Record

In fact, it is suspected that only the basic data types such as String, Integer, Decimal and Date plus Business Objects that contain these are actually allowed.

IBM supplied Coach View utility functions

IBM has supplied a few utility functions which can be leveraged while developing Coach Views.

Getting URL of managed assets – getManagedAssetUrl

A global function called `com_ibm_bpm_coach.getManagedAssetUrl` is available which will return the URL of Process App managed assets. This can be used to access assets stored with the Process App. Common examples of usage for this would include accessing images, script files, CSS or data.

The syntax is:

```
com_ibm_bpm_coach.getManagedAssetUrl(  
    assetName,  
    assetType,  
    projectShortName,  
    returnWithoutAssetName)
```

The return from this function is the String value of the URL for the asset.

Name	Type	Description
assetName	String	The name of the managed asset.
assetType	String	The type of the managed asset. This must be one of the following values: <ul style="list-style-type: none">• <code>com_ibm_bpm_coach.assetType_WEB</code>• <code>com_ibm_bpm_coach.assetType_SERVER</code>• <code>com_ibm_bpm_coach.assetType DESIGN</code>
projectShortName	String	Optional. The short name (acronym) of the process app or toolkit which owns this managed file.
returnWithoutAssetName	Boolean	Optional, default is false. Determines whether the returned URL should include the name of the asset or just the path to the asset.

Generated HTML

When a Coach View is inserted into a Coach within the Human Service editor, the HTML tags that are generated look as follows:

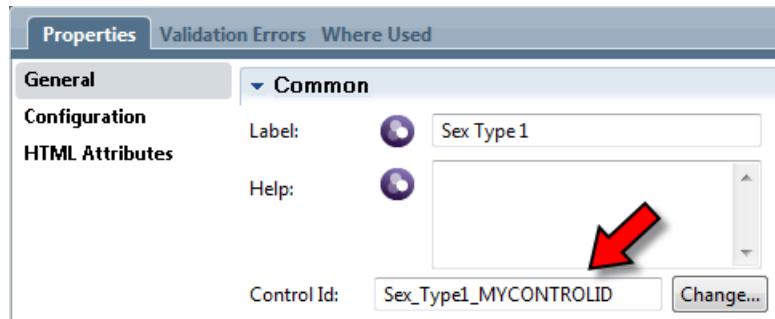
```
<div id="random_generated"  
    class="??? topLevel CoachView"  
    data-eventid=""  
    data-viewid="controlId"  
    data-config=""  
    data-bindingtype=""
```

```

data-binding=""
data-type="com.ibm.bpm.coach.<UUID>">
... content ...
</div>

```

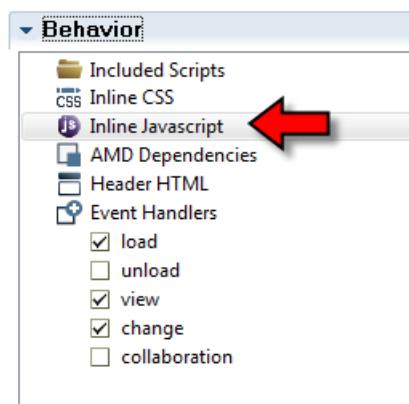
The attributes `id` and `data-viewid` seem to be the same and both appear to be the value supplied inside Process Designer within the "Control Id" field:



Experimentation seems to show that `data-binding` contains the name of the variable in the server but without the "`tw`" suffix. For example, if the variable bound at the server were "`tw.local.xyz`" then the value of this HTML attribute would appear to be "`local.xyz`". This may be the first sign that "`tw`" (which historically was used for the acronym for TeamWorks) is being removed.

Custom JavaScript in Coach Views

In some cases, you will wish to insert custom JavaScript in your Coach Views. You can insert the code in any of the event handler or in the "In-line JavaScript" section:



It is not uncommon to want to execute the same segment of code as the result of a load event, change event or view event. Replicating the actual code is inefficient. A solution is to create a function that contains that code and hang that function off the View Object which is available as "this" in an "In-line Javascript".

Using JavaScript libraries other than Dojo

The IBM Coach technology is a framework for handling task interactions. When a task is created by a process, the Coach technology (as found in a Human Service) can describe a screen to be shown to the user. The screen is composed of multiple instances of "Coach Views" which are "widgets" or "controls" in other UI languages. The internal architecture of the Coach framework is obviously a combination of HTML, CSS and JavaScript as it runs within the browser. When the browser is asked to "display a coach", the framework for loading and managing the Coach Views as well as providing them data is IBM written logic. Because that logic is written in JavaScript and

because browsers are not yet "consistent" in their operations, a JavaScript library is leveraged by the IBM code in order to achieve its task. The library chosen by IBM is "Dojo".

What this means is that the IBM Coach framework has internals for managing the Coach that are written to use Dojo. This requires that some small portions of the Dojo library are required to be loaded into the browser for the Coach to function. There is no customer exposed mechanism for disabling this nor are there any options to replace that code with another JavaScript library.

The inclusion of Dojo for use solely by the Coach framework should not interfere at all with the use of other libraries (eg. YUI) for custom Coach View implementation. The inclusion of Dojo should be transparent.

Generated content in a control

The DOM element in the browser that represents the control can be found by:

```
var element = dojo.query("*[data-viewid=\"" + this.context.viewid + "\"]")[0];
```

however, the reference to this DOM node is already set in the variable:

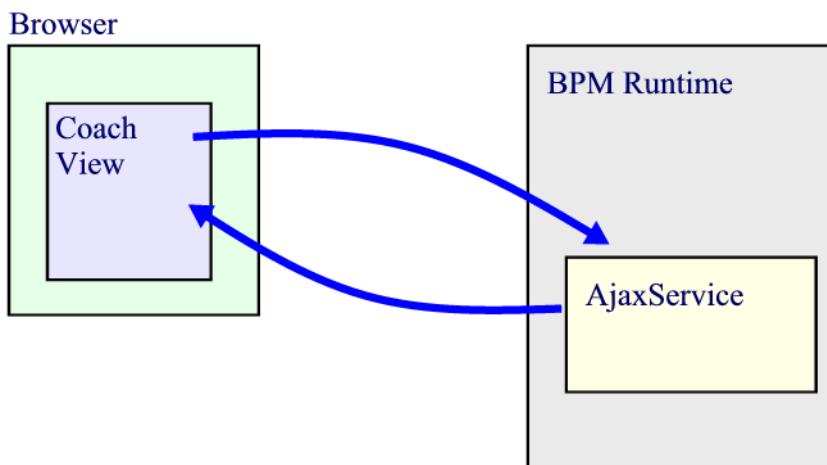
```
this.context.element
```

When a Context Box is added to a Coach View, it will be found in the DOM as:

```
<div ... this Coach View instance ... >
  <div ... class="ContentBox" ...>
    ... Nested Coach Views ...
    ... Nested Coach Views ...
  </div>
</div>
```

Making Ajax and REST requests from a Coach View

A Coach View runs within the browser and doesn't have direct access to any server side data. However, IBPM exposes a type of service called an "Ajax Service". An Ajax Service, just like other service types exposes inputs and outputs as either simple data types or as Business Objects. An Ajax Service can be invoked remotely by the Coach View.

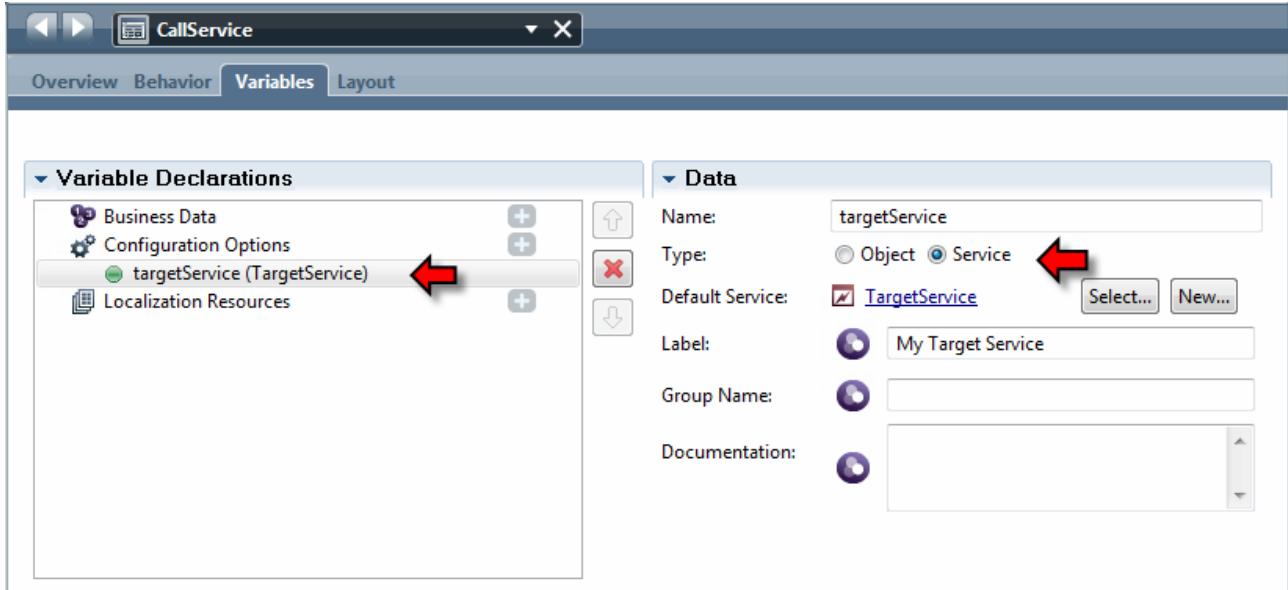


Invoking an Ajax Service involves performing a REST request from the coach view to the run-time naming the Ajax Service to be called and passing in the parameters. IBM BPM provides a REST API to invoke Ajax Services and these can be used directly.

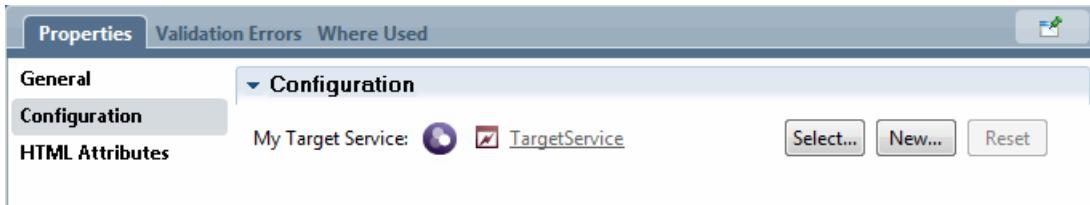
An easier solution though is to create a service based configuration parameter on the Coach View.

This can have a default Ajax Service named or else can be changed by the user of the Coach View within the Coach editor.

To illustrate how to achieve this effect, here is an example of a new Coach View containing an Ajax call. In the Configuration Options for the Coach View, we define a new option called "targetService". We then define the type of this option to be "Service". Finally, we have the option of defining a default service from the list of services available:



When an instance of this Coach View is added to a Coach within Process Designer, we will find that we have an option available corresponding to the option defined in the Coach View:



Ok, so far so good. We have now defined a new option on a Coach View which is basically a "reference" to a run-time Ajax Service. Now we need to consider how to invoke an instance of this service.

Within the Coach View implementation, we have the "context" object which has the bound data and configurations associated with it. For each configuration parameter defined, we have:

```
this.context.options.<optionName>
```

available to us. Because we have defined an option called "targetService" (in our example), we will find that:

```
this.context.options.targetService
```

is a property. We will also find that this is an instance of a function. If we invoke this function, the result will be an invocation of the Ajax Service on the server. That was pretty easy.

SO:

```
this.context.options.targetService();
```

will invoke the Ajax Service.

We should now consider passing data into and getting data from the Ajax service. Because we are

working in JavaScript here, it is assumed that you understand the concept of JavaScript objects. Each Coach View configuration property that is a service takes a single JavaScript object as a parameter that is passed when the Coach View JavaScript code calls the service. The following fields are defined for this object:

- `params` – A JavaScript object with a property for each of the input variables expected by the service that will be called.

Note: As of 2013-11-24 – We seem to have a problem passing in data of type "Date".

- `load` – A function which will be called when the Ajax Service returns. It has a parameter which contains the data returned from the Ajax Service.
- `error` – A function which will be called if the Ajax Service returns with an error.

Here is a snippet of code which shows invoking an Ajax Service from within a Coach View:

```
this.context.options.targetService = {
    params: {
        bol: {"a": "A Value", "b": "B Value"}
    },
    load: function(ret) {
        console.log("Returned");
    },
    error: function(ret) {
        debugger;
    }
};
```

As an alternative to making an Ajax call as shown above, we can also make explicit REST requests using the Dojo XHR technology.

REST requests can be made from a Coach View quite easily. The trick is to use the Dojo XHR functions.

Within the code of your Coach View, code the following:

```
var args = {
    url: "<target URL>",
    handleAs: "json",
    load: lang.hitch(this, function(data) {
        // Code to process response goes here
    })
}
xhr.get(args);
```

Two Dojo AMD bindings are used:

- `dojo/_base/xhr` → `xhr`
- `dojo/_base/lang` → `lang`

Remember that REST requests are asynchronous and that the `load` function is called at some time later in the processing.

Navigating and access to Coach Views on a Coach page

Consider a Coach shown to the user. The Coach is composed of Coach Views on the page. These Coach Views exist in tree oriented structures.

Let us start with the page itself. At the root of the page, there will be one or more top-level Coach Views. These are the ones added directly to the canvas of the Coach itself. A variable called

`com_ibm_bpm_global.topLevelCoachViews` is an object which contains a property for each canvas top level Coach. Note that `com_ibm_bpm_global.topLevelCoachViews` is a JavaScript object and NOT a list.

Given a particular Coach View object, the `context.viewid` property names the view ID for that Coach View instance.

For a given Coach View, the variable called `contextSubview` is an object that contains a property for each of the Coach Views that it contains. However, be careful. The corresponding property in `subview` is **not** the root of the corresponding Coach View. It is somehow related to an HTML DIV in a mysterious way. If we want the Coach View object for the view ID then we should use the method `context.getSubview(viewid)` which will return an array of real Coach View objects corresponding to the `viewid`. It is not yet known how there can be more than one element returned since it was believed that at a given level, all Coach View instances had to be unique.

A method called `context.parentView()` will return us the parent of our current Coach View, assuming that our current Coach View is not already on the base Coach in which case it has no parent.

Here is some example code for walking all the Coach Views in a page:

```
tvar showChild = function(parentCoachView, indent) {
    for (var child in parentCoachView.contextSubview) {
        if (parentCoachView.contextSubview.hasOwnProperty(child)) {
            var childList = parentCoachView.contextSubview[child];
            for (var j=0; j<childList.length; j++) {
                var currentCoach = childList[j];
                var pad = "";
                for (i=0; i<indent; i++) {
                    pad = pad + " ";
                }
                console.log(pad + "> " + currentCoach.context.viewid);
                showChild(currentCoach, indent + 2);
            }
        }
    };
};

for (var child in com_ibm_bpm_global.topLevelCoachViews) {
    if (com_ibm_bpm_global.topLevelCoachViews.hasOwnProperty(child)) {
        var currentCoach = com_ibm_bpm_global.topLevelCoachViews[child];
        console.log("Canvas > " + currentCoach.context.viewid);
        showChild(currentCoach, 2);
    }
}
```

See also:

- `com_ibm_bpm_global.topLevelCoachViews`
- `context.parentView()`
- `contextSubview[viewid]`
- `context.viewid`

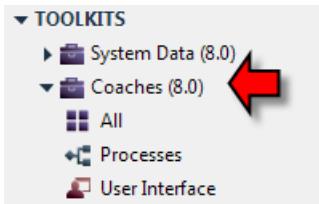
Learning resources to build new Coach Views

The skills required to build new or customize existing Coach Views *can* be quite broad and varied and includes:

- HTML
- JavaScript
- Dojo

- CSS
- IBM BPM Coach View Framework

The source of the controls supplied with the product can be examined to understand how they function and how they were put together by IBM. These controls can be found in the Coaches toolkit.



In addition, there are a growing wealth of "as-is" Coach Views available at the IBM BPM Wiki site.

See also:

- [BPM Community Wiki – Working with Coach Views](#)
- [BPM v8 Tutorial Video – Building a custom Coach View](#) - 2012-06-06

Debugging Coach Views

While developing Coach Views, it can be useful to use debuggers to see what is going on. Most of the browsers available today have debugging tools built into them including FireFox, IE and Chrome. Using these debuggers we can set break points and examine the content of the DOM tree and Java Scripts.

As an example, if we insert the statement:

```
debugger;
```

into a `load()` function callback and view the coach in Fire Fox with Fire Bug installed, we see the following:

A screenshot of the Firebug developer toolbar in Firefox. The 'Script' tab is selected, showing a code editor with a red circle indicating a breakpoint at line 67. The code is a snippet of JavaScript related to a CoachView. To the right, the 'Watch' tab is open, displaying the current state of variables. It shows 'this' pointing to a 'CoachView' object with properties like '_bindingHandles', '_contentBoxTemplates', '_counter', 'context', '_parent', 'parentView', 'binding', 'bpm', 'element', 'options', '_metadata', and 'myOption1'. The value for 'myOption1' is shown as an object with a 'value' property set to 'undefined'. Other variables like 'readonly', 'subview', and 'viewid' are also listed.

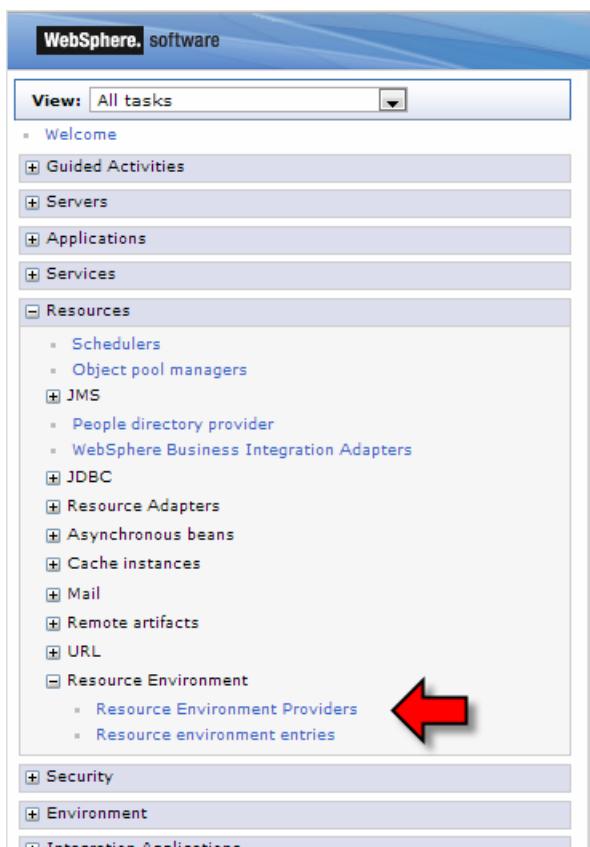
Looking on the right hand side, we can see the value of bindings as well as of configuration options. At the time of writing, my weapon of choice for browser/client side debugging is Google Chrome. Chrome has a built in development tools so no 3rd party plug-ins are needed.

When debugging a Coach View, you may find that the code shown in the debugger appears "compressed" and un-readable. The IBM supplied code, including Dojo is what is called "minified". What this means is that the interpreted JavaScript that comprises the IBM code and Dojo code has been "compressed" by shrinking variable names and whitespace (including line breaks). The reason for this is that by being compressed, it is smaller in size and hence there is less data to be moved over the network when the libraries are being used in the browser. This is a standard technique. However when debugging your own code, it can be confusing. Fortunately,

IBM has provided a solution. There is a debugging flag that can be enabled which causes a different version of the JavaScript libraries to be used. That alternate version is not minified and hence does not show compressed in the debuggers.

Here is the recipe to enable the debugging:

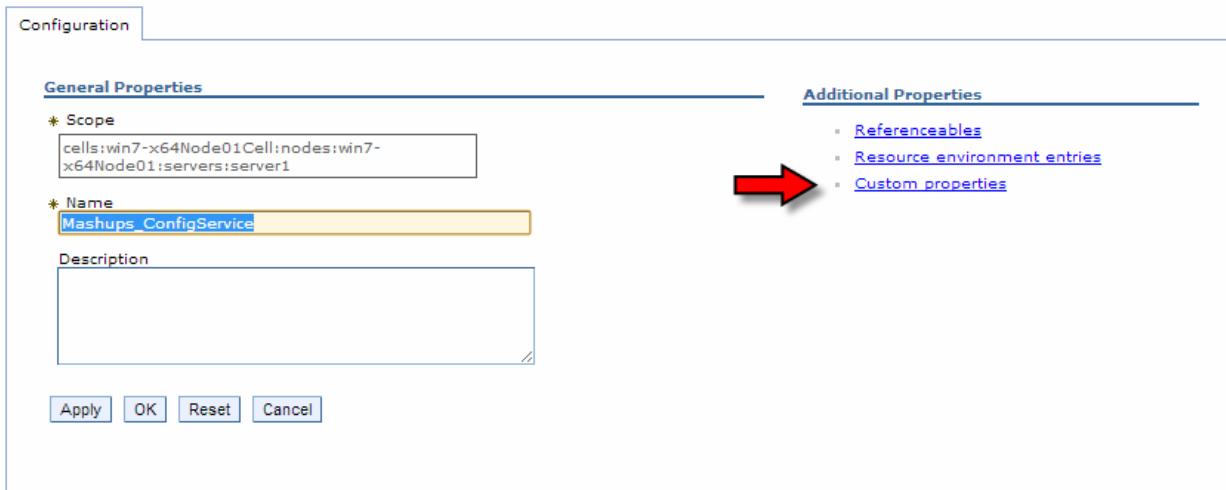
1. Open up the WAS admin console.
2. Expand Resources > Resource Environment > Resource Environment Providers



3. Select the provider called "Mashups_ConfigService"

	Name	Scope
You can administer the following resources:		
<input type="checkbox"/>	Mashups_BlobConfigService	Node=win
<input type="checkbox"/>	Mashups_CacheService	Node=win
<input type="checkbox"/>	Mashups_ConfigService	Node=win
<input type="checkbox"/>	Mashups_ContextMenu_Provider	Node=win
<input type="checkbox"/>	Mashups_DynamicSpotMapping	Node=win
<input type="checkbox"/>	Mashups_Endpoints	Node=win
<input type="checkbox"/>	Mashups_SpaceTemplates	Node=win
<input type="checkbox"/>	Mashups_Spaces	Node=win
Total 8		

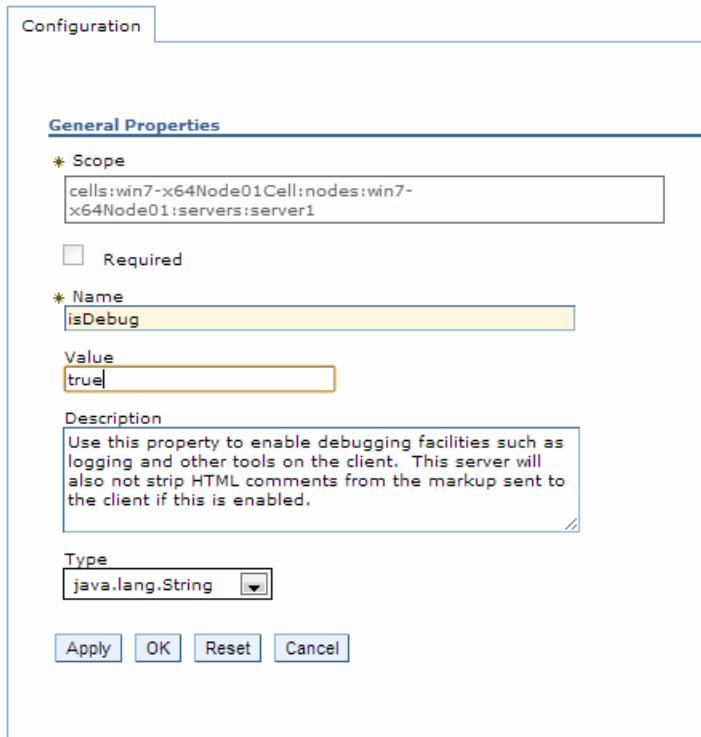
4. Select Custom Properties



5. Select the "isDebug" property. The easiest way to find this is to use the filter

The screenshot shows a filtered list interface. At the top, there are buttons for New..., Delete, and a search/filter icon. Below is a table with columns: Select, Name, Value, Description, and Required. A filter bar at the top of the table shows 'Filter: isDebug'. Below the table, a message says 'To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *, ?, %).'. A 'Filter' dropdown shows 'Name' and a search term 'isDebug'. A 'Go' button is next to it. Below the table, a message says 'You can administer the following resources:' followed by a table row. This row contains a checkbox, the value 'isDebug' (with a red arrow pointing to it), 'true', a detailed description of the isDebug property, and the value 'false'. At the bottom, it says 'Total 154 Filtered total: 1'.

6. Change the value from false to true (or true to false to disable again) and click OK



- With the resource definition changed, save/commit the changes in the WAS admin console and restart the WAS server. Your debug changes will not start happening until after a server restart. Once the server has been restarted, code will now show up non-minified. Realize that this may have an impact on overall performance but a small price to play during Coach View debugging. If you are using a shared environment with other developers, be sensitive to their needs as well as your own.

Coach View Construction Tips

DOM root access in a Coach View

It is common to include a "Custom HTML" entry in a Coach View so that content may be added to it, a typical example might be:

```
<div class="MyClass">  
</div>
```

Within the `load()` event, we would then create DOM content and add it into the DOM model. To find the appropriate entry in the DOM tree, the following code fragment will work:

```
var myNode = query(".MyClass", this.context.element)[0];
```

Editing JavaScript for Coach Views

Unless the edits are small, it is not recommended to build the code of the callbacks for Coach Views in the Process Designer editors. There is very little assistance here and mistakes can easily happen. Instead, use a professional JavaScript editor with entry assist and copy/paste the code from the editor into Process Designer. You will forget to perform this copy/paste a few times during development but the trade between making that mistake and the errors you will introduce by not using a proper JavaScript editor will more than compensate. I also recommend prefixing your JavaScript code (in the editor) with some dummy definitions for the commonly used functions that

will be used in your own code. This will provide the ability to have entry assist.

Adding a label

Many IBM supplied controls have a label associated with them. This label can be shown above the new Coach View.

```
<div class="textLabel">
    <label class="controlLabel"></label>
</div>
```

In the load code, we can set the text of the label with:

```
var labelDom = query("label", this.context.element)[0];
if (this.context.options._metadata.label != undefined &&
    this.context.options._metadata.label.get("value") != "") {
    labelDom.innerHTML = this.context.htmlEscape(this.context.options._metadata.label.get("value"));
}
```

In addition, we will want to handle the visibility settings of the label:

```
var labelDivDomNode = this.context.element.querySelector(".textLabel");
if (this.context.options._metadata.label == undefined ||
    this.context.options._metadata.label.get("value") == "" ||
    (this.context.options._metadata.labelVisibility != undefined &&
     this.context.options._metadata.labelVisibility.get("value") == "HIDE")) {
    // hide the label div
    this.context.setDisplay(false, labelDivDomNode);
} else {
    // show the label div
    this.context.setDisplay(true, labelDivDomNode);
}
```

Working with visibility

The Coach View framework makes available a Visibility concept.



This is set on a Coach View by Coach View instance basis. The values are:

- Default (inherit from parent) – DEFAULT – Inherit the value from the parent coach view
- Required – REQUIRED – The Coach View is editable and MUST have a value
- Editable – EDITABLE – The Coach View is editable
- Read Only – READONLY – The Coach View is read only
- None (hide or disable) – NONE – The Coach View is hidden

It is up to each Coach View implementation to implement code to honor these settings. One way is to create a handler function in the load:

```
this._handleVisibility = function() {
    var visibility = "DEFAULT";
    if (this.context.options._metadata.visibility != undefined) {
        visibility = this.context.options._metadata.visibility.get("value");
    }

    if (visibility == "NONE") {
```

```

        this.context.setDisplay(false);
    } else {
        this.context.setDisplay(true);
    }
}

```

Then we will call this handler in the view function and also when the `_metadata.visibility` property changes as seen in the change function.

Using a Dijit Widget in a Coach View

When using a Dijit Widget in a Coach View, each widget is expected to have a unique "id" value. Unfortunately, when we place a Coach View in a table, the same "`context.viewid`" is used for each row in the column and hence the widget can end up with the same id which won't work. Fortunately, Dojo provides an answer. The method

```
dijit.getUniqueId("<widgetType>")
```

will generate a unique id. The AMD object to load is:

`dijit → dijit`

Including a custom Dijit Widget in a Coach View

There are times when you might want to create your own custom Dijit Widget and use that when creating a new coach view. Here is the recipe for achieving that. At a high level, the story progresses by:

- Writing and testing our new Dijit Widget
- Packaging that widget in a ZIP file with a directory structure mapping to the module
- Adding the ZIP containing the Widget as a Web managed file
- Creating a "requires" JavaScript fragment mapping the module name to the ZIP file
- Adding the "requires" JavaScript fragment as a managed Web file
- Defining the "requires" JavaScript as an included JavaScript fragment
- Defining an AMD definition for our new widget

We will illustrate by example. Let us imagine that we are creating a new Widget called "DateProgress" that exists in the module called:

`kolban/widget/DateProgress`

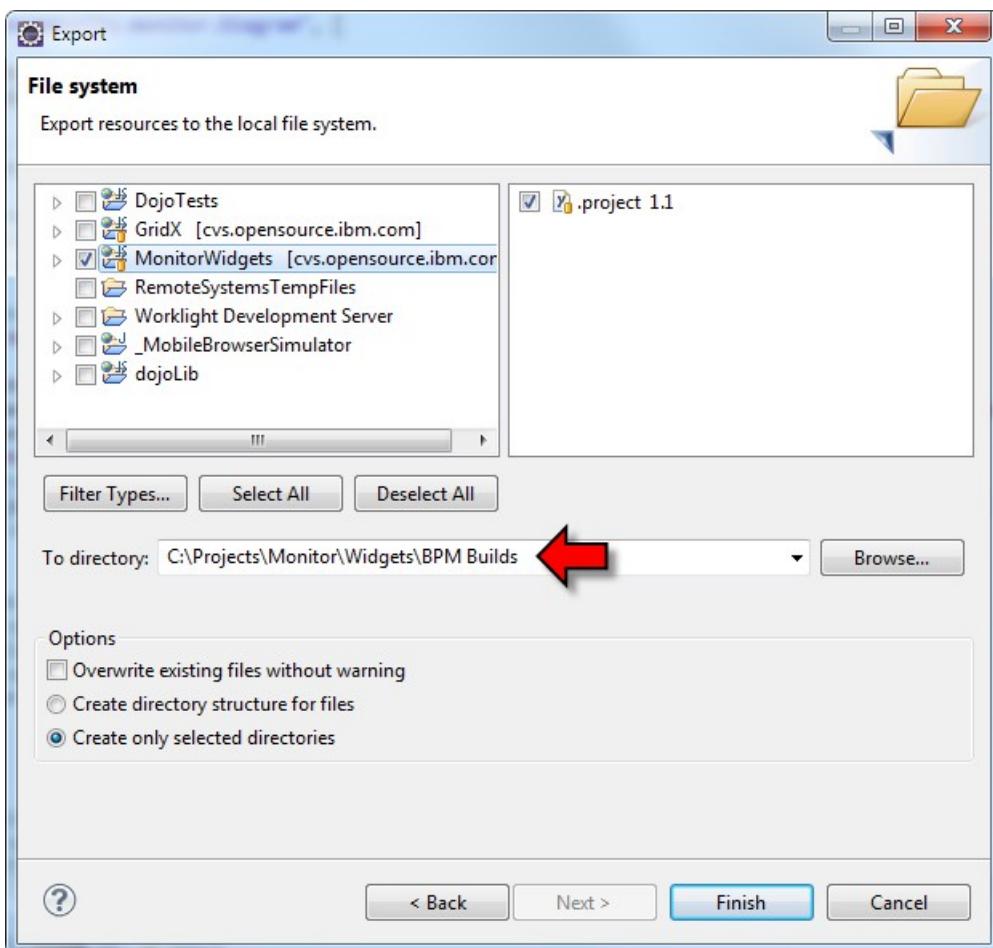
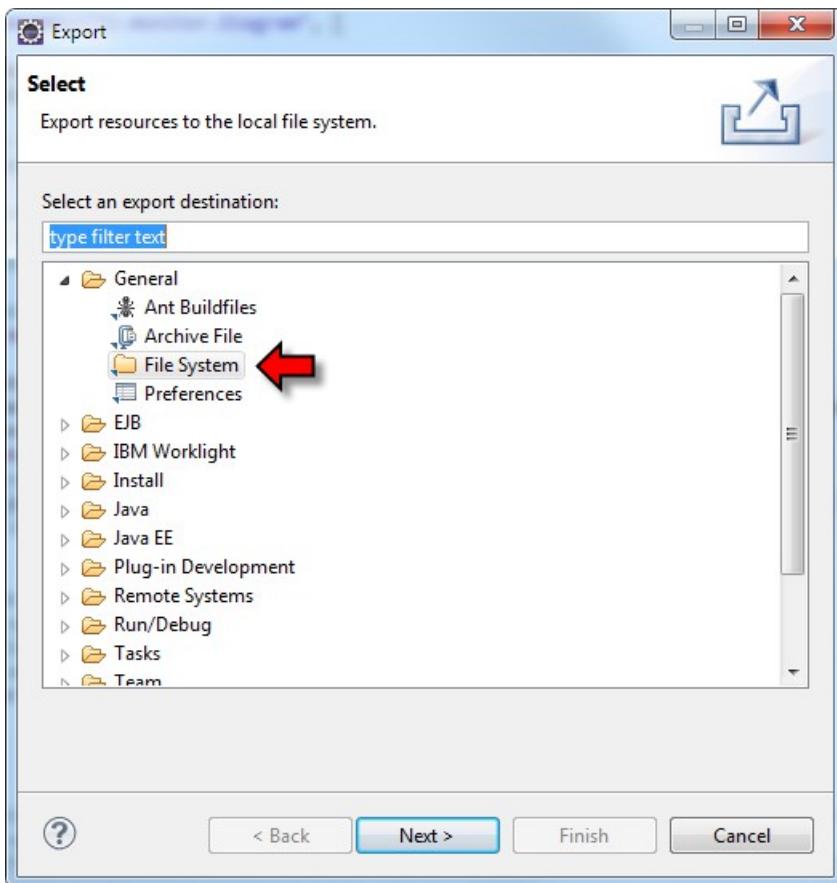
we will assume the following source level tree structure.

```

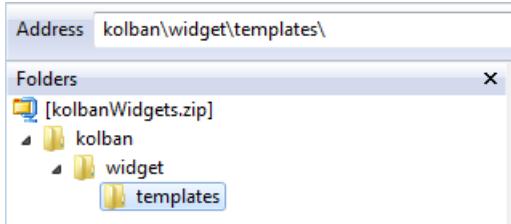
kolban
  widget
    DateProgress.js
    templates
      DateProgress.htm

```

We can export from an Eclipse based environment using the Export to File System:



We now zip this structure up

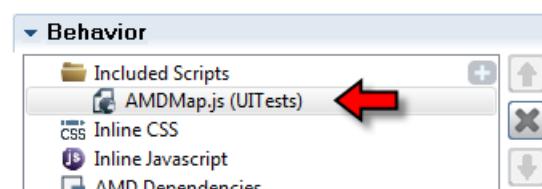


We add this zip (kolbanWidgets.zip) as a Web managed file to our BPM Process App.

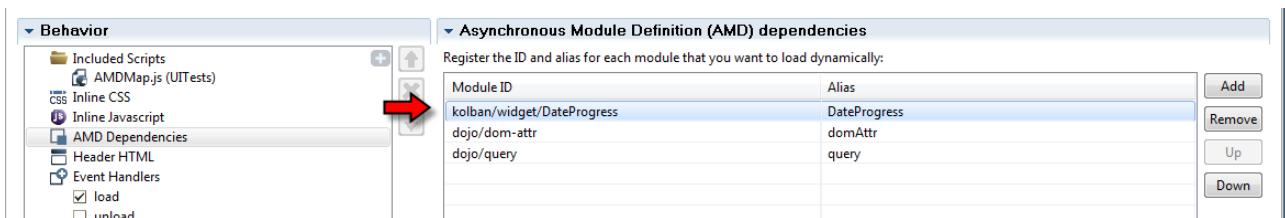
Next we create a JavaScript file (AMDMap.js) that contains:

```
require({
    packages:
    [
        {
            name: "kolban",
            location:
                com_ibm_bpm_coach.getManagedAssetUrl('kolbanWidgets.zip',
                    com_ibm_bpm_coach.assetType_WEB, '<ToolkitAcronym>') + "/kolban"
        }
    ]
});
```

This JavaScript source file then also gets added as a Web managed file. Now we have two managed files. One is the ZIP that contains our Dijit widgets and the other is the JavaScript AMD mapper. In the Coach View included scripts, we reference the AMDMap.js file:



Finally, we can now create a AMD dependency on our new module and map it to a variable.



Describing a Dojo based page declaratively

Within Dojo, we can describe the Dojo widgets to be used either programmatically or declaratively. Programmatically means that we build the page layout within JavaScript by creating instances of widgets and placing them at the appropriate DOM locations within the DOM tree. This gives us optimal control over placement and parameters but has the significant down-side that we can't use any visual editor to build them out. In addition, modification becomes complex as we have to remember the layout at the program level. All in all, except for the simplest of pages, it is a slow and cumbersome process.

The alternative is known as declarative layout. In this instance, we describe our screen through the use of HTML which can be constructed in a suitable Dojo aware HTML editor such as Worklight Studio or Maquette. The HTML that describes the Coach View can then be placed in a Custom

HTML component within the layout section of the Coach View definition. When the Coach View is loaded, the HTML contained within that area is parsed looking for Dojo declarations. Those Dojo declarations then cause the creation of the associated widgets. At the conclusion, we have a Coach View which contains widgets and looks great.

The power of Coach Views is that they are not just static HTML but contain BPM data and respond to events. We want to then augment the Coach View with JavaScript code that will handle interactions with those widgets. When we created the widgets programatically, we had access to the object that represented those widgets and could add handler and other logic. When we create the description of the page declaratively, we need a little bit more work to get those widget object references.

Dojo provides a class called "dijit/registry" which has a method upon it called "byId(<name>)". This method returns a reference to the Dojo widget on the page that has the corresponding id name. So for example, if we have a button on the Coach View that has an id of "mySubmit", we can gain access to the Dojo widget corresponding to that button with:

```
var myButton = registry.byId("mySubmit");
```

There is one final snag with this though and that is the rule that each Widget on the page must have a unique id value. Defining:

```
<div id="mySubmit"
      data-dojo-type="dijit.form.Button"
      data-dojo-props="label:'Submit'>
</div>
```

Has a problem. If we have two instances of the Coach View on the page then both buttons will be given the same id and things will break. IBM BPM has a solution to this. If instead we code:

```
<div id="$$viewDOMID$$_mySubmit" data-dojo-type="dijit.form.Button"
      data-dojo-props="label:'Submit'>
</div>
```

The special code of " \$\$viewDOMID\$\$ " is replaced with the DOM ID of the Coach View itself. So if we want to retrieve the button widget, we can now code:

```
var myButton = registry.byId(domAttr.get(this.context.element, "id") + "_mySubmit");
```

For any Dojo widgets used in a declarative style, always ensure that you have also used the AMD loader to load the corresponding Dojo implementation of those widgets.

See also:

- Error: Reference source not found

Accessing images by CSS

It is not uncommon to use CSS to access an image. For example, instead of using the tag to show an image, one can use CSS to set the image in a <div>. For example:

```
.myClass {
    width: 24px;
    height: 24px;
    background-image: url(A URL to an image);
}
```

will set the image. The "url" function in the CSS defines where the image is loaded. Unfortunately, there appears to be a puzzle when we build Coach Views. What should the URL be to the image file? We desire that the image be packaged with the Coach View so the first notion of adding the image as a managed Web File seems like it might be sound ... however there is a problem. The URL to the image can only be determined programatically. It doesn't have a fixed value. With CSS

styling, we don't have access to the JavaScript and hence can't dynamically define the location.

One solution to this puzzle is really quiet elegant. It works as follows:

1. Build the CSS you want to include in the Coach View. When you need reference to images, supply the URL for those images as file names relative to where the CSS file can be found.
2. Place the images relative to the CSS file on the file system.
3. ZIP up the CSS file and the images maintaining the relative directory structure.
4. Add the resulting ZIP file as a managed web file.
5. Include the CSS file in the Coach View in the Coach View editor naming the ZIP file and the CSS file contained within.

What happens is that when the CSS is loaded, its reference to the image files will be local to the ZIP file and the server will satisfy those requests from the content of the ZIP.

If you need to update the CSS or image, you must repeat the process which includes packaging up the file system into the ZIP and refreshing the managed web file that corresponds to that ZIP.

See also:

- Error: Reference source not found

Commonly used AMD loaders

When building Coach Views, there are AMD functions that simply keep coming up over and over. Here we capture a cheat sheet for some of the most common:

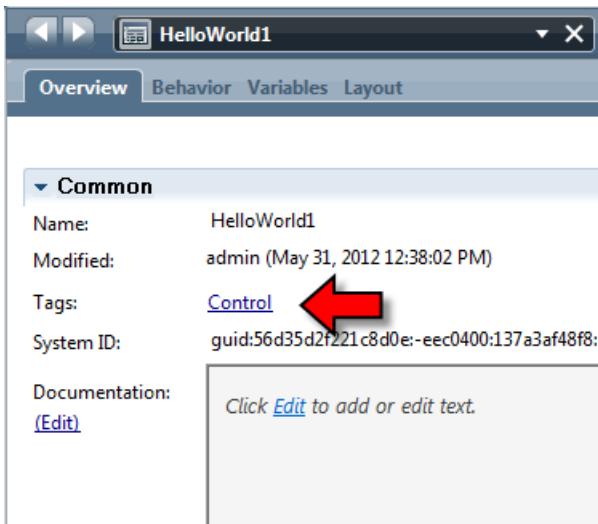
AMD	Alias	Example Functions
dojo/query	query	query
dojo/_base/lang	lang	lang.hitch
dojo/dom-attr	domAttr	domAttr.get, domAttr.set

Sample Coach Views

Sample Coach View – trivial "hello world"

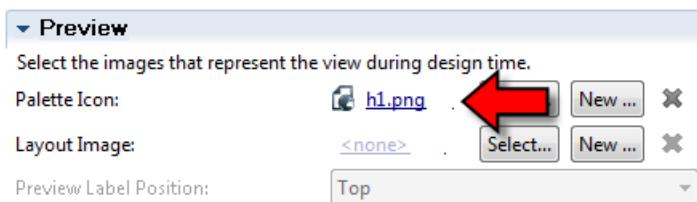
We start by creating a new Coach View. I called this one "HelloWorld1".

I defined the new Coach View as having a tag of "Control":

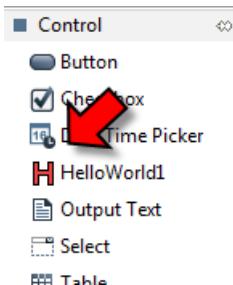


This causes the new Coach View to appear in the Controls section of the Coach Editor.

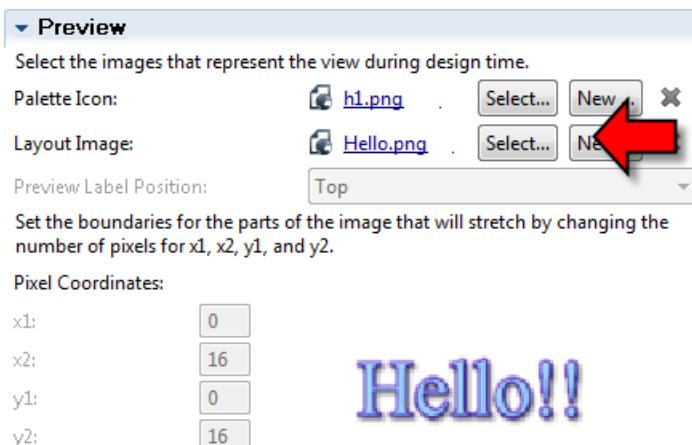
Next, I built an icon that was 16x16 pixels. I used GIMP to create it but you can find many free icons on the Internet. I added this icon into the Files of my Process App as a Web managed file.



By doing this, when the new Coach View appears in a Coach editor, it will have that icon in the palette of available Coach Views. Looking ahead, it will look like:



Next an image was defined for the layout in the Coach Designer editor:

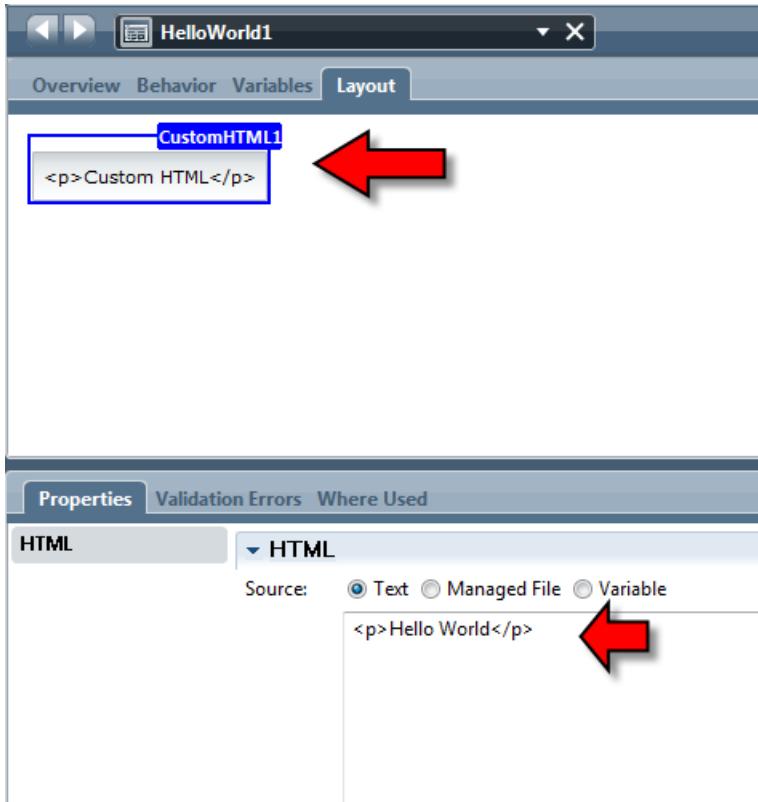


By defining an image here as the layout image, when the Coach Editor is shown, an appropriate

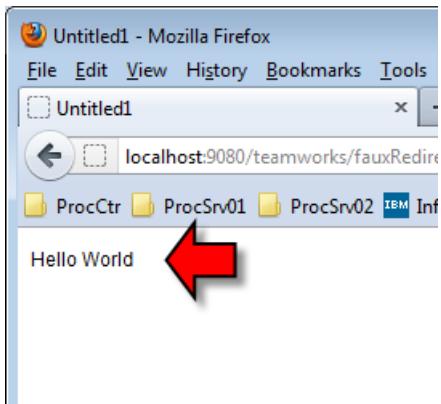
place holder will be seen in the editor canvas area when an instance of the Coach View is added to the canvas.



In the Layout tab, a CustomHTML component was added.



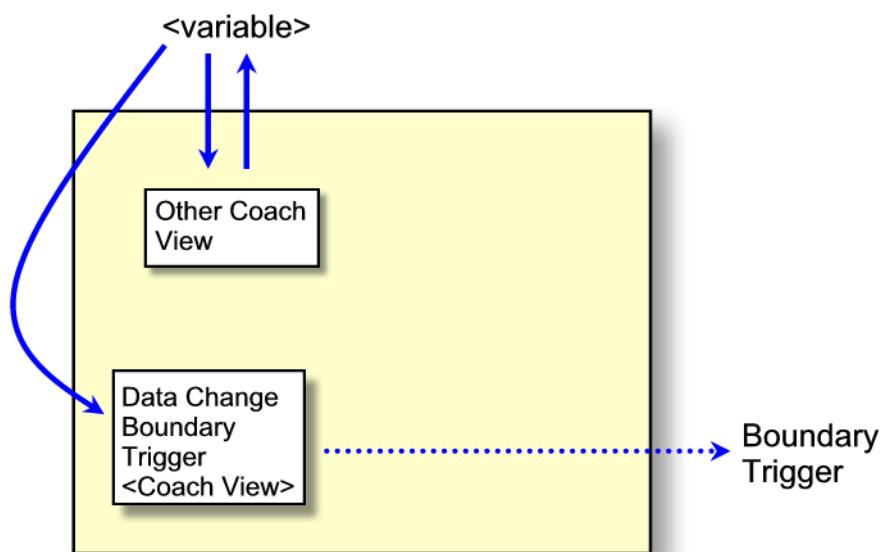
In the HTML properties, a small fragment of HTML was added to provide some visualization. When I now create a Coach and add the Hello World Coach View component into the body of the coach and have the Coach display, we end up with an HTML output as desired/expected.



Sample Coach View – Data Change Boundary Trigger

When data changes in a coach view, unless that causes a boundary trigger, no other coach view on the page will be informed of the change. This is not always a good thing. Consider a table that shows a list of customers by region. We might also wish to have a combo box in which we can select the region. If we use the Select coach view, changes to the selection causes nothing else to happen. We can conceivably edit the Select coach view to cause a trigger to occur but this would result in a second coach view for this customized select. Not a good thing. What we want to happen is for a boundary event to fire when the Select changes its data.

Here we examine a way to make that happen. What we will do is build a new Coach View called Data Change Boundary Trigger (a long name ... but very descriptive of what it does). This Coach View will have **no** visual appearance. Instead, what it will do is monitor a data value and, if that data value changes, cause a Boundary Trigger event to fire.



The Coach View is configured to fire a boundary event.

Screenshot of the Data Change Boundary Trigger configuration screen. The window title is 'Data Change Boundary Trigger'. The tabs at the top are 'Overview', 'Behavior', 'Variables', and 'Layout'. The 'Overview' tab is selected.

Common		Usage	
Name:	Data Change Boundary Trigger	Can Fire a Boundary Event:	<input checked="" type="checkbox"/>
Modified:	admin (Jul 11, 2012 6:43:53 PM)	Use as a Template:	<input type="checkbox"/>
Tags:	Kolban	Supports a Label:	<input type="checkbox"/>
System ID:	guid:56d35d2f221c8d0e:-481bf8ca:138593edfb3:-7e9a	Preview	
Documentation:	Monitor a data value and, when it changes, fire a corresponding boundary trigger event. Author: Neil Kolban (kolban@us.ibm.com) Support: As-is/Best effort V0.1 - 2012-07-11		
(Edit)	Select the images that represent the view during design time. Palette Icon: <none> . Select... New... X Layout Image: <none> . Select... New... X Preview Label Position: Top		

The Coach View has a handler for the "change" event. When the local change event fires, we trigger a boundary condition.

Behavior

- Included Scripts
- Inline CSS
- Inline Javascript
- AMD Dependencies
- Header HTML
- Event Handlers
 - load
 - unload
 - view
 - change** (checked)
 - collaboration

Code: change

```
function(event)
  // Fire a boundary condition when the associated data has changed.
  //console.log("CV: OnChange: My OnChange fired!");
  this.context.trigger(function() {
    // Do Nothing
    // console.log("Trigger completed!")
  });
}

(this == the view object)
```

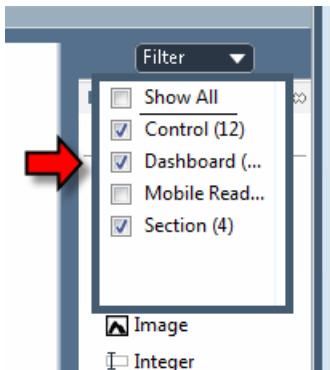
and ... that's it. No more complexity than that.

Coach View Templates

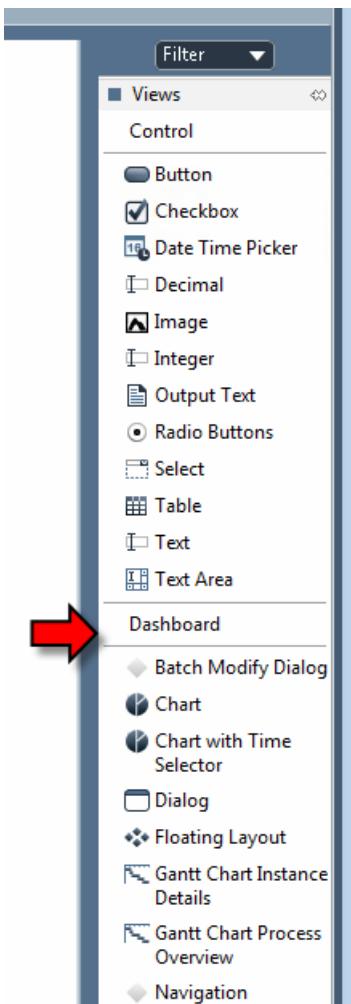
When a new Coach View is created, it can be flagged as a Template. What this means is that the new Coach View will not be available for inclusion in Coaches but instead, when future new Coach Views are created they can use the Coach Views flagged as templates as starting places for the new Coach Views.

Dashboard Coach Views

A toolkit is supplied with the product named "Dashboards". When added to a Process App project an additional set of Coach Views are available. By default, these are not shown in the Coach editor and must be explicitly enabled within the filter menu:



Once done, a new set of building blocks will be found underneath the Dashboard grouping:



With the controls now available to be added to our Coaches by the coach designer, we will turn our attention to descriptions of each of them.

See also:

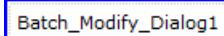
- Team Performance
- Process Performance

Batch Modify Dialog Control

Palette icon:



Canvas appearance:



Configuration parameters:

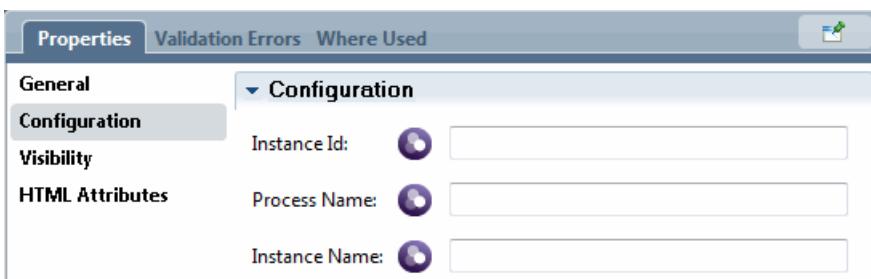


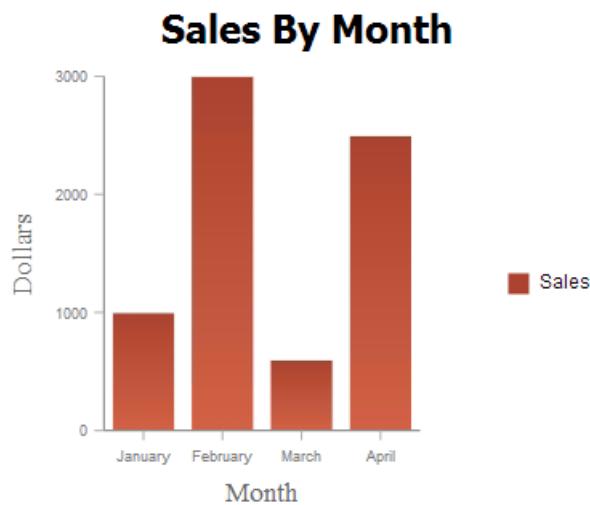
Chart Control

The Chart control is used to create and display charts that show data to the end user. The types of charts available include pie, bar, column and line. The data type used for binding this control is called "ChartData". These charts are based upon the Dojo charting controls.

Before we get into the details of a Chart Control, let us first ensure that we are grounded in what we want to achieve. A chart is used to graphically display some data. Commonly, we can visualize that data as a table:

Month	Sales
January	\$1,000.00
February	\$3,000.00
March	\$600.00
April	\$2,500.00

If we now think of the Month values as the X-Axis and the Y-Axis as the values, we can achieve a chart like:

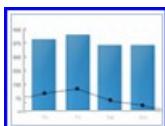


It is important to visualize in ones mind the relationship between table data and a potential chart to show that data. If necessary, write down on paper some sample table data and ensure that you solidly comprehend how your would take that table data and create a chart.

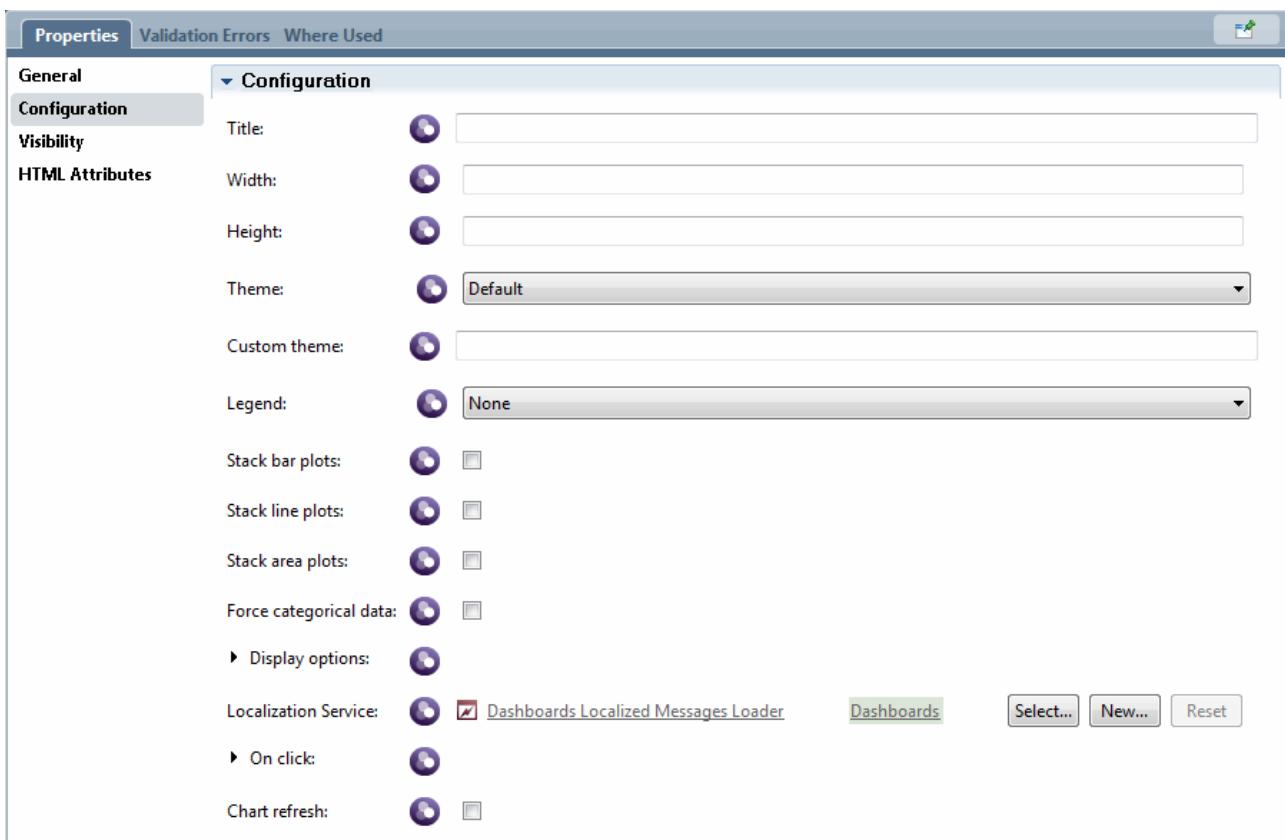
Palette icon:



Canvas appearance:



Configuration parameters:



The primary chart configuration is its data binding which is of type `ChartData`. It is a highly nested structure which is described next bottom up.

The `ChartDataPoint` object contains two fields:

- `name (String)` – The name of the data point. This contributes to the point's label.
- `value (Decimal)` – The value of the data point.

The `ChartDataSeries` object contains two fields:

- `label (String)` – The label shown in the legend
- `data (List of ChartDataPoint)` – The list of `ChartDataPoint` objects that comprise this series of data items.

A `ChartDataSeries` contains the data to be shown in the chart. For different chart types, the `ChartDataPoints` represents different entities:

- Pie – A pie chart
- Bar – A single horizontal bar
- Column – A single vertical column
- Line – A single point included in the line

A `ChartDataPlot` object contains the following:

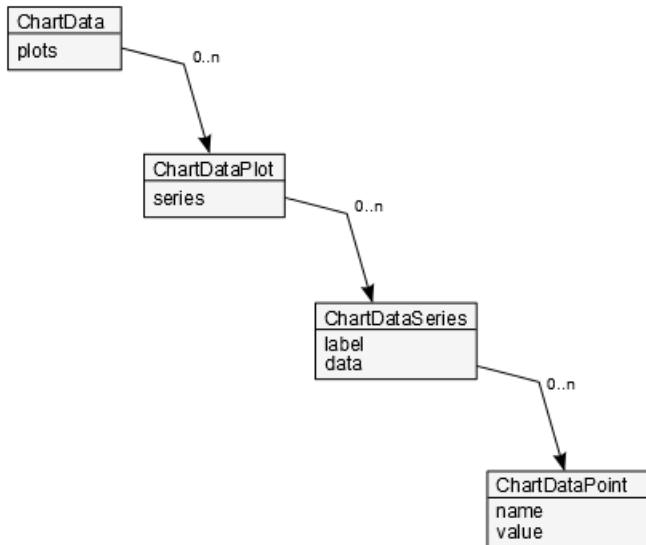
- `series (List of ChartDataSeries)`

The `ChartDataPlot` contains multiple `ChartDataSeries`.

A `ChartData` object contains the following:

- plots (List of ChartDataPlots)

The resulting structure then looks like:



Here is an example of a template of a variable to hold a ChartData object:

```

tw.local.myChartData = {
  plots: [
    {
      series: [
        {
          label: "My Label",
          data: [
            { name: "name1", value: 1 },
            { name: "name2", value: 2 }
          ]
        }
      ]
    }
  ];
}
  
```

So, with this interesting series of data structures out of the way, let us look at some simple plots of data.

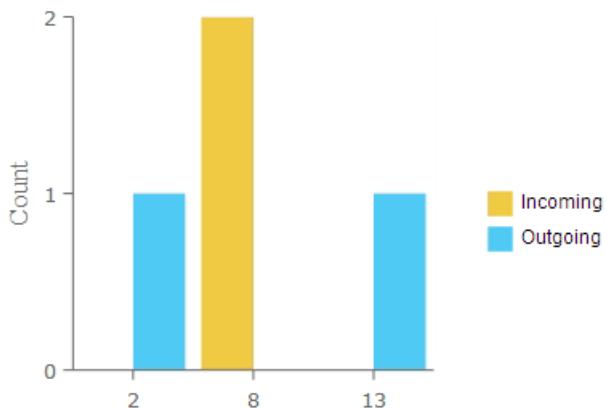
Horizontal Bar	Vertical Bar	Line	Area Line	Pie

The primary configuration of a chart is through the chart properties page which contains:

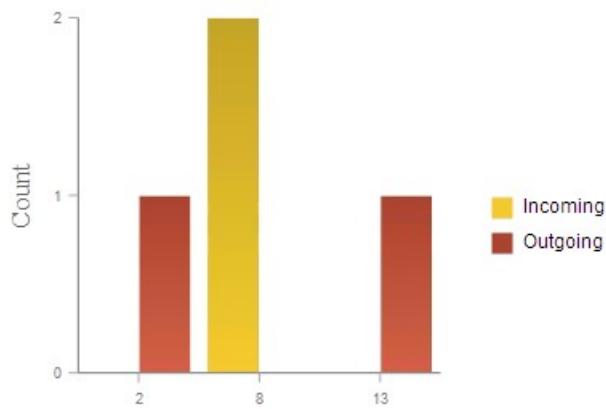
- Title (*String*) – The title of the chart. This appears to be an HTML Head Level 2 tag.
- Width (*Decimal*) – The width of the chart in pixels.
- Height (*Decimal*) – The height of the chart in pixels.

- Theme – The theme used to describe the visuals of the chart. Values include:

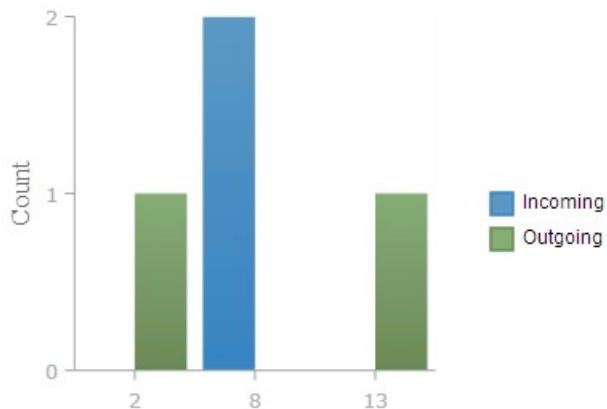
- Default



- At risk and overdue



- Turnover trend



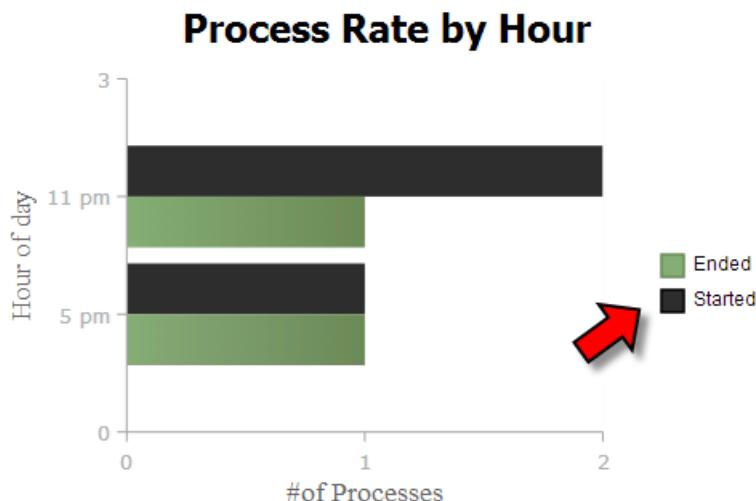
- Use custom theme

These themes are supplied by IBM BPM.

- Custom Theme – If the Theme setting is "Use custom theme", this property describes the style of theme to use. This will be a JSON version of a `dojox.charting` style object. The JSON data is provided as constructor information to the `Theme()` object.
- Legend (`ChartLegendPositionSelection`) – Where (if anywhere) the chart legend is shown. Choices are:

- None (*none*) – No legend is shown.
- Left (*left*) – The legend is displayed to the left of the chart.
- Bottom (*bottom*) – The legend is displayed on the bottom of the chart.
- Right (*right*) – The legend is displayed to the right of the chart.

In the following image, we see a legend shown to the right:



- Stack bar plots
- Stack line plots
- Stack area plots
- Force categorical data – Forces the scale on the X axis to include space for items not present and scales from the minimum value to the maximum value in width to make the chart "even".
- On Click – This option can be bound to a variable of type `ChartClickEvent`. This data type contains:
 - `seriesLabel (String)` – The series that was clicked
 - `dataPointIndex (Integer)` – The index into the series that was clicked
 - `dataPoint (ChartDataPoint)` – The `ChartDataPoint` object that was clicked

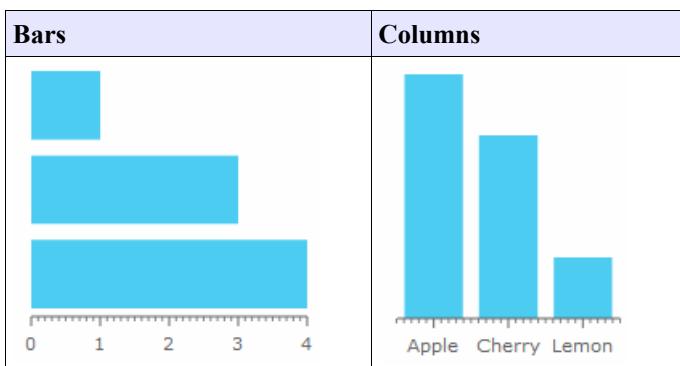
When a section of the chart is clicked on by the user a boundary trigger event is fired and the variable's values changes to reflect the item clicked. This can be used by other Coach Views or functions to cause an update as a result of a user selection. A good example of this might be a drill-down where the user sees something interesting in the chart and wishes to see more about that item.

When one clicks on a chart, we can see it become hi-lighted. This may not always be the desired effect. We can disable this by setting the chart's style attribute to "`outline-style: none`".

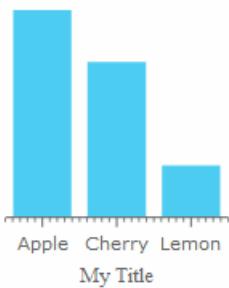
- `Chart refresh (Boolean)` – If set to "true" the chart refreshes itself and then resets the flag variable to false. This causes the chart to redraw itself completely. Necessary if we wish the chart to update itself based upon any configuration property changes.

The way the data is shown is based on the configuration property called "Display options". This is a rich data structure that contains a list of `ChartPlotDisplayOptions`:

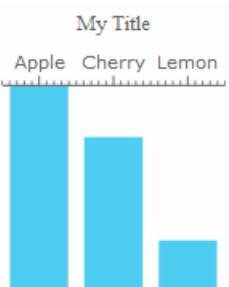
- `plotType (ChartPlotType)` – The type of chart to draw for this plot. The choices are
 - Bars – Draw a horizontal bar
 - Columns – Draw a vertical bar
 - Lines – Draw line
 - Areas – Draw an area
 - Pie – Draw a pie
- `plotTypeFixed (Boolean)`
- `displayHorizontalAxis (Boolean)` – If set to `true`, a horizontal axis is drawn for all chart types but Pie. Pie has no axis. Here are samples for columns and bars:



- `displayHorizontalAxisScrollButtons (Boolean)`
- `horizontalAxisTitle (String)` – Adds a title to the horizontal axis. This is not shown if the display of the horizontal axis is not enabled.



- `flipHorizontalAxisToTop (Boolean)` – By default the horizontal axis is shown beneath the chart. Setting this property to true causes the horizontal axis to appear above the chart.

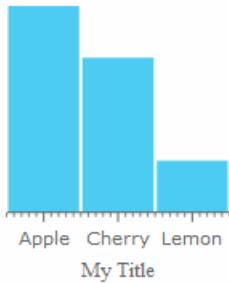


- `horizontalAxisMajorTickMarks (Decimal)` – How often to draw the major tick.

0 disables.

- `horizontalAxisMinorTickMarks (Decimal)` – How often to draw the minor tick.
0 disables.
- `horizontalAxisWindowScale (Decimal)`
- `horizontalAxisWindowOffset (Decimal)`
- `horizontalAxisWindowOffsetIsTrailing (Boolean)`
- `displayVerticalAxis (Boolean)` - If set to true, a horizontal axis is drawn for all chart types but Pie. Pie has no axis. Here are samples for columns and bars:
 - `verticalAxisTitle (String)`
 - `flipVerticalAxisToRight (Boolean)`
 - `verticalAxisMajorTicks (Decimal)`
 - `verticalAxisMinorTicks (Decimal)`
 - `pieLabels (ChartPieLabelsSelection)`
 - `tooltipTemplate (String)`
 - `gap (Integer)` – The gap between bar and column entries in pixels.

In this example, we set the gap to be 1 pixel:



- `minBarSize (Integer)` – The minimum width of a bar or column in pixels. The width of each bar will be scaled as a function of the width of the chart, the number of bars to be shown and the gap requested between bars. This provides guidance on the minimum width of a requested bar.
- `maxBarSize (Integer)`

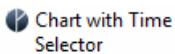
When working with chart data, we also have the opportunity to show corresponding tables of the data. If we look carefully at the Series, we see it is a list of `ChartDataPoint` objects where each of those contains a name and a value property. If we bind a BPM Table Coach View to the list of `ChartDataPoint` and then create two columns ... one of text and bound to "name" and other of decimal bound to "value", we end up with a pleasant table showing the data in the chart:

Month	Procurements
September	1
October	2

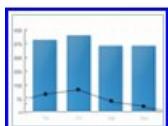
Chart with Time Selector Control

Although this control advertises itself as a chart with time selection, reality seems to show that it is actually a specialized chart that shows bar and column items with a trend line running through it. This seems to "spoil" it for general purpose use.

Palette icon:



Canvas appearance:



Configuration parameters:

Screenshot of the configuration properties for the Chart with Time Selector control in a software interface.

The configuration includes the following parameters:

- General:
 - Title: [Input field]
 - Width: [Input field]
 - Height: [Input field]
 - Theme: [Dropdown menu set to Default]
 - Custom theme: [Input field]
 - Legend: [Dropdown menu set to None]
 - Stack bar plots: [Switch]
 - Stack line plots: [Switch]
 - Stack area plots: [Switch]
 - Force categorical data: [Switch]
- Localization Service: [Service selection dropdown set to Dashboards Localized Messages Loader] with [Select...], [New...], and [Reset] buttons.
- Trend Unit: [Dropdown menu set to Hour]
- Trend unit service: [Service selection dropdown set to ChartDataUnitService] with [Select...], [New...], and [Reset] buttons.
- Trend service: [Service selection dropdown set to ChartDataInstanceTrend] with [Select...], [New...], and [Reset] buttons.
- Display options:
 - [Switch]
- Process ID: [Input field]
- Process Application ID: [Input field]
- Team ID: [Input field]

- Title
- Width
- Height
- Theme

- Custom Theme
- Legend
- Stack bar plots
- Stack line plots
- Stack area plots
- Force categorical data
- Display options
- Localization Service
- Trend Unit
- Trend unit service
- Trend service

This is an Ajax service that has the following inputs:

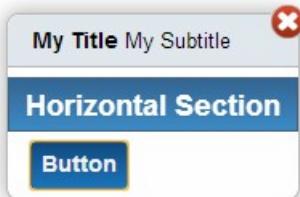
- units (String)
- numPeriods (Integer)
- endPeriod (String)
- timezone (String)
- searchFilter (String)
- processId (String)
- processAppId (String)
- teamId (String)

which returns the following output:

- categoricalData (ChartData)
- Trend service input
- Process ID
- Process Application ID
- Team ID

Dialog Control

The dialog control is used to open a child window within the Coach. Initially, when the Coach appears, the dialog box is not shown. The control can be bound to a Boolean typed variable. When the variable becomes true, the dialog is shown. Here is an example of what the dialog control looks like:

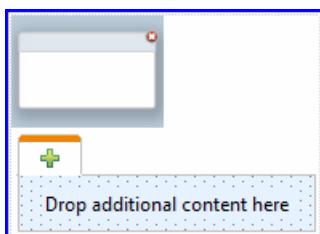


Its top area is composed to two title areas. One defined by the property called "Title" and the other defined by the "Subtitle" property. In the upper right of the dialog is a close button which appears to be the only way to close the dialog.

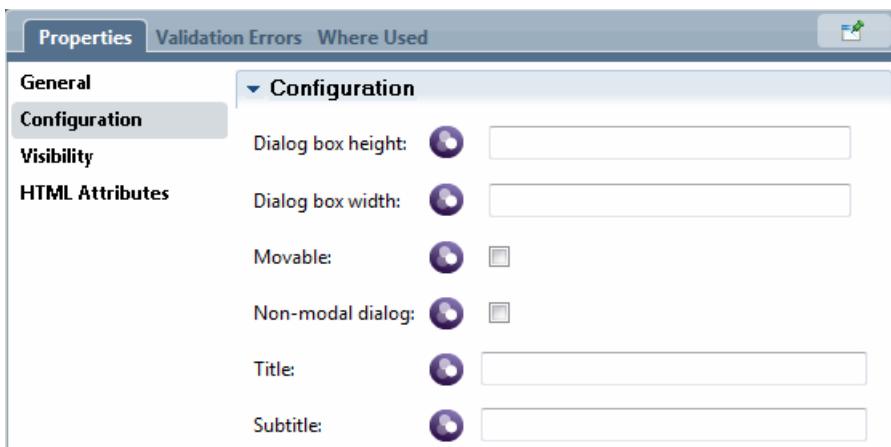
Palette icon:



Canvas appearance:



Configuration parameters:

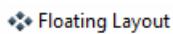


The properties for this Control are:

- Dialog box height (Integer)
- Dialog box width (Integer)
- Movable (Boolean)
- Non-modal dialog (Boolean)
- Title (*String*) – The text shown in the title of the dialog.
- Subtitle (*String*) – The text also shown in the title of the dialog.

Floating Layout Control

Palette icon:



Canvas appearance:



Configuration parameters:

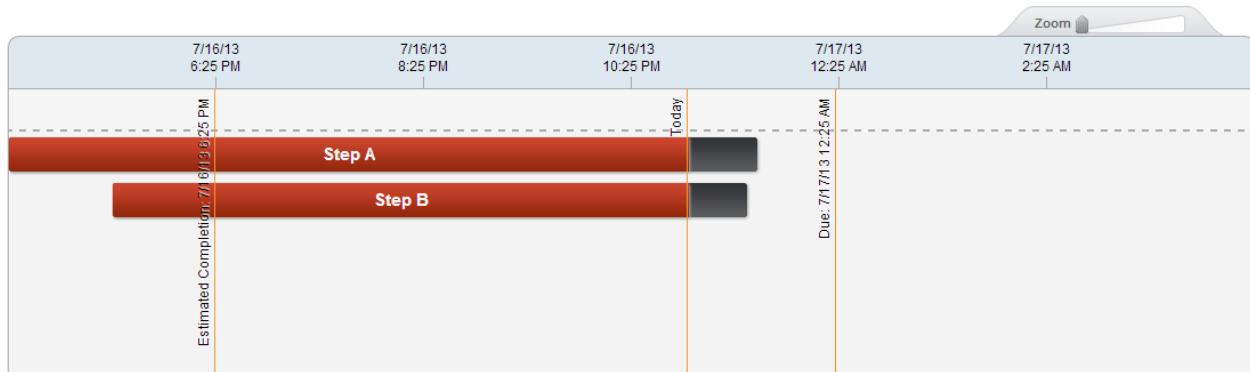


Gantt Chart Instance Details Control

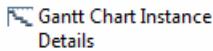
The binding type for this control is an instance of GanttChartInstanceDetailsData. This data structure contains:

- processId
- Phases
- Tasks
- processName
- processStartDate
- processEndDate
- processDueDate
- processEstimatedCompletionDate
- Stream

Here is an example of this control:



Palette icon:



Canvas appearance:



Configuration parameters:

A screenshot of the 'Properties' dialog box. The 'General' tab is selected at the top. Below it, the 'Configuration' tab is selected. The configuration section contains several settings with checkboxes and dropdown menus:

- Show task details:
- Update service: Default Instance Gantt Save Service [Dashboards](#) Select... New... Reset
- Unsaved Changes:
- Save Changes:
- Chart Reset:
- Localization Service: Dashboards Localized Messages Loader [Dashboards](#) Select... New... Reset
- Preview service: Default Instance Gantt Data Service [Dashboards](#) Select... New... Reset
- Calendar Type: Gregorian [Select...](#) Clear

- Show task details
- Update service
- Unsaved Changes
- Chart Reset
- Localization Service
- Preview Service
- Calendar Type
- Available actions

Gantt Chart Process Overview Control

The binding data for this control is an instance of `GanttChartProcessOverviewData`. This contains the following fields:

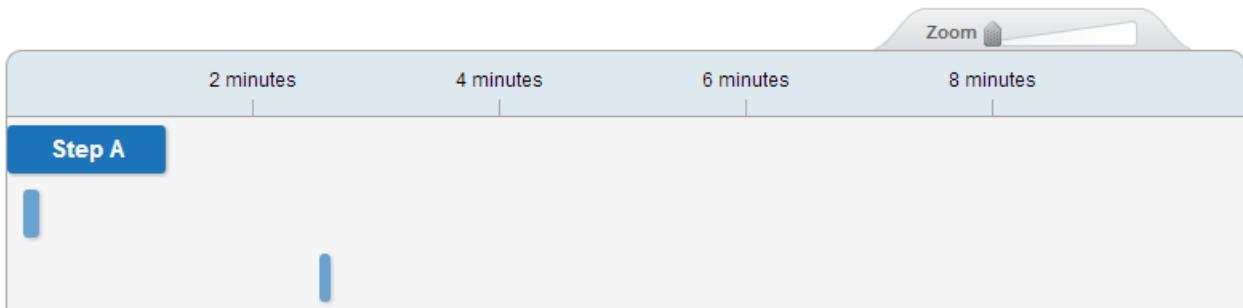
- processId
- processName
- Tasks
- processAverageDuration

A service provided as part of the Dashboards toolkit called "Process Gantt Chart

"Initialization Service" returns an instance of this object when called with:

- ProcessAppId – The ID of the process application containing the process model. An example would be an instance of TWProcessApp.id.
- BPDIId – The ID of a process model. An example would be an instance of TWProcess.id.
- SearchFilterString – Unknown.

Here is an example of what this control looks like:



Palette icon:



Canvas appearance:



Configuration parameters:

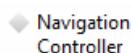
A screenshot of the configuration properties for the Gantt chart control. The "Properties" tab is selected. Under the "Configuration" section, there is a dropdown menu set to "Localization Service". Next to it is a checkbox labeled "Dashboards Localized Messages Loader" which is checked. To the right of the checkbox are several buttons: "Dashboards", "Select...", "New...", and "Reset".

- Localization Service
- Chart refresh

Navigation Controller Control

The Navigation Controller control appears to be an unusual component in the mix. It almost appears as an after-thought. It provides no visual appearance, no parameters and no data-binding. It simply "is". The documentation on it is light ... presumably because there is so little that can be done to configure it. However, from examination and reverse engineering, it seems to be an event subscriber that, when triggered, causes a jump to a different web page or a refresh of the current web page with different content.

Palette icon:



Canvas appearance:

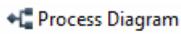
Navigation_Controller1

Configuration parameters:

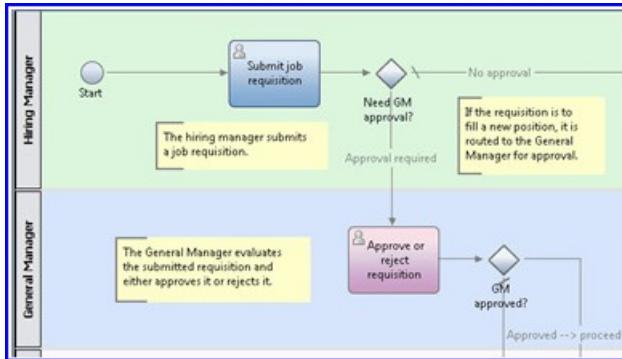
None.

Process Diagram Control

Palette icon:



Canvas appearance:



Configuration parameters:

Properties Validation Errors Where Used

General Configuration

Configuration

Zoom Level:

Process ID:

Snapshot ID:

Branch ID:

Project ID:

Process Instance ID:

Activity Summaries:

Show Activity Summaries:

Risk State:

Step ID:

Traversed Path:

Projected Path:

Custom Path:

Localization Service: Dashboards Localized Messages Loader

Available Actions:

- Zoom Level (Decimal) – The zoom level of the diagram. A value of 1 through 10.
- Process ID

- Snapshot ID
- Branch ID
- Project ID
- Process Instance ID (String) – The process instance ID who's diagram will be shown. This value has precedence over the "Process ID" value.
- Activity Summaries
- Show Activity Summaries
- Risk State
- Traversed Path
- Projected Path
- Custom Path
- Localization Service
- Available Actions

Process Due Date Control

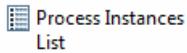
Palette icon:

Canvas appearance:

Configuration parameters:

Process Instances List Control

Palette icon:



Canvas appearance:



Configuration parameters:

Properties Validation Errors Where Used

General

Configuration

Visibility

HTML Attributes

Configuration

List title:

Retrieval service: Default Process Instance List Service

Height:

Number of instances displayed:

Risk state:

Step:

Step risk state:

Search filter:

Process application ID:

Process ID:

Process application name:

Process name:

Estimated completion service: Default Instance Estimated Completion Service

▶ Process instances:

- List title
- Retrieval service
- Height (Integer) – The height of the control in pixels. The default is 600 pixels.
- Number of instances displayed
- Risk state
- Step
- Step risk rate
- Search filter
- Process application ID
- Process ID
- Process application name
- Process name
- Estimated completion service
- Process instances
-

Process Summary Control

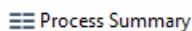
This control provides a summary in terms of numbers of overdue, at risk and on track counts of a given process type.



The binding data for this control is a `ProcessSummary` instance. An example of sourcing data for this might be:

```
var myProcess = tw.system.model.findProcessByName("Process 1");
tw.local.processSummary = myProcess.performanceMetrics.retrieveProcessSummary(null, false);
```

Palette icon:



Canvas appearance:



Configuration parameters:

Properties		Validation Errors	Where Used
General	Configuration		
Configuration	Localization Service: <input checked="" type="button"/> Dashboards Localized Messages Loader <input type="button"/> <input type="button"/> Select... <input type="button"/> New... <input type="button"/> Reset Expanded Mode: <input checked="" type="radio"/> Custom Theme: <input type="button"/>		
Visibility			
HTML Attributes			

- Localization Service
- Expanded Mode
- Custom Theme

See also:

- Data Type – `ProcessSummary`

Search Control

Palette icon:



Canvas appearance:



Configuration parameters:

The screenshot shows the 'Properties' dialog for the Stream Control. The 'Configuration' tab is selected. Under the 'Configuration' section, there are several configuration parameters:

- Localization Service:** A dropdown menu showing "Dashboards Localized Messages Loader" with a checked checkbox, and buttons for "Select...", "New...", and "Reset".
- Auto-completion Service:** A dropdown menu showing "Default Data Label Autocompletion Service" with a checked checkbox, and buttons for "Select...", "New...", and "Reset".
- Fire Boundary Event on change:** A checkbox that is checked.
- Disable Auto-completion:** A checkbox that is checked.
- Auto-completion delay in milliseconds:** An input field containing the value "1000".
- Search scope:** A dropdown menu showing "Team" with a checked checkbox.
- Business process ID:** An input field containing the value "1000".
- Process application ID:** An input field containing the value "1000".

Stream Control

Palette icon:

Canvas appearance:

Configuration parameters:

Task List Control

This control shows the list of tasks available for a user. The control seems to be able to update itself dynamically so as tasks are added or removed, the control dynamically changes. Here is an example of a task list with three tasks:

▼ Due Today (3)

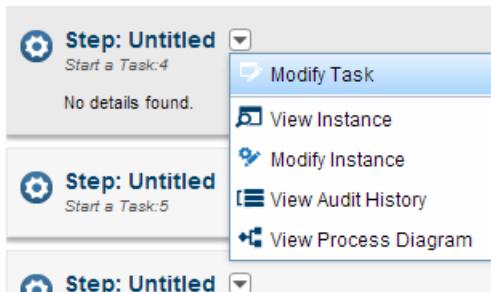
The screenshot shows the Task List Control displaying three tasks:

- Step: Untitled** (Due: June 20, 2013 1:17 PM) - Assigned to All Users
- Step: Untitled** (Due: June 20, 2013 1:18 PM) - Assigned to All Users
- Step: Untitled** (Due: June 20, 2013 1:18 PM) - Assigned to All Users

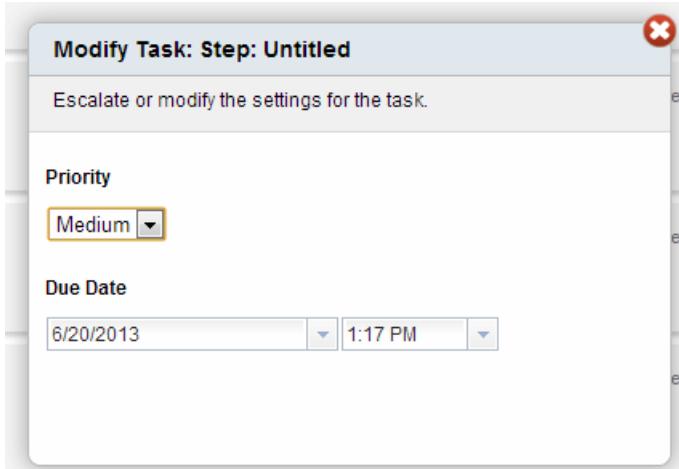
Showing 3 of approximately 3 results

The heritage of this Control models the Process Portal that was built for v8.0 of the product. The Task List basically shows horizontal bars with one bar for each task. If we click on a task, the details of that task are shown. This is the value defined in the "Narrative" section of the task definition.

A pull-down menu on each task allows the user to perform additional actions.



The "Modify Task" option opens a dialog which allows the priority and due date of the task to be changed.



We have the option of only showing a subset of available tasks using the "Initial list size" option. This controls how many items are initially shown. If there are more items to be shown than we have initially asked for, a "Show More" button is displayed:

▼ Due Today (3 of 13)

Step: Untitled ▾ Start a Task:4	Due: June 20, 2013 1:17 PM All Users
Step: Untitled ▾ Start a Task:5	Due: June 20, 2013 1:18 PM All Users
Step: Untitled ▾ Start a Task:6	Due: June 20, 2013 1:18 PM All Users

Showing 3 of approximately 13 results

Show More

Clicking on the description of the task should launch the Coach for that task. At present, it is not known how to achieve this. When clicked and there is no current own

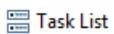
If no tasks are available, the phrase "No tasks were found" is shown:

No tasks were found.

The Task List Control has a relationship to the "Navigation Controller" control.

Unfortunately, it appears that the height of the Task List controller is fixed at 600 pixels. There is no apparent way this can be changed through the use of a style sheet.

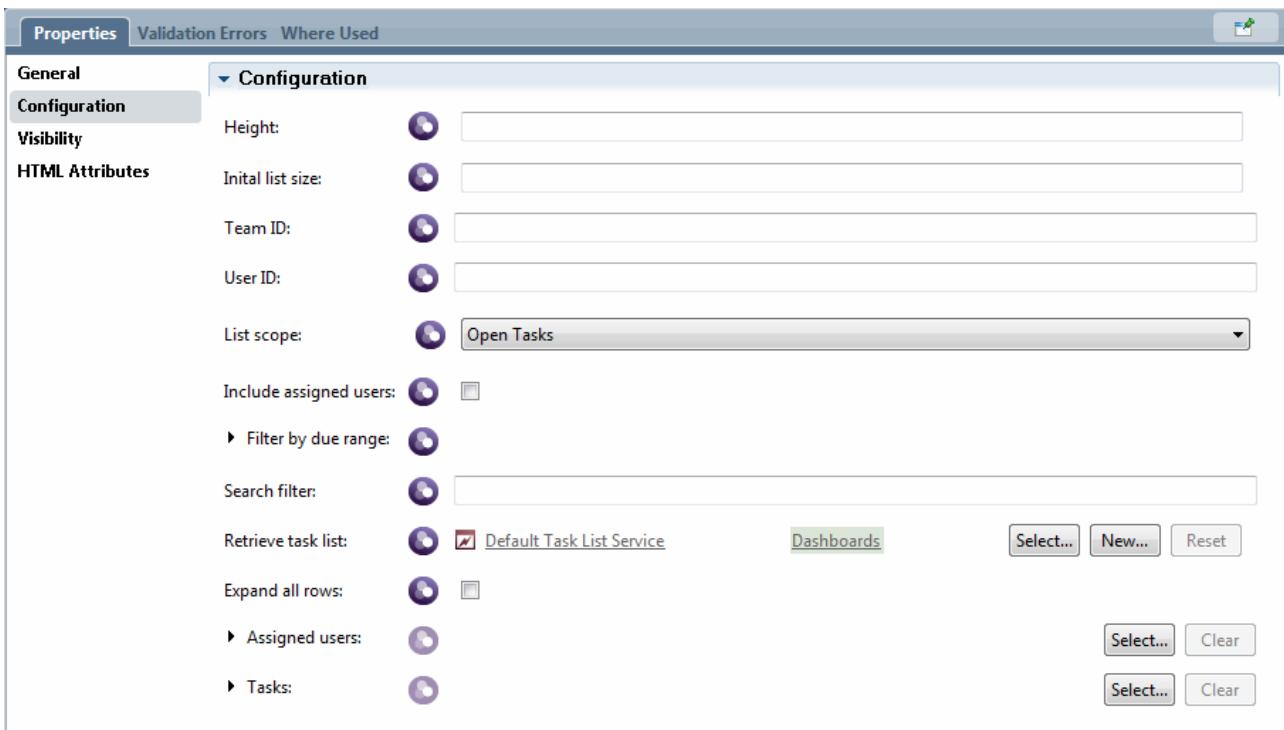
Palette icon:



Canvas appearance:



Configuration parameters:



The screenshot shows the "Properties" dialog for a "Task List" control. The "Configuration" tab is selected. The configuration parameters are as follows:

- Height:** A dropdown menu with a preview window.
- Initial list size:** A dropdown menu with a preview window.
- Team ID:** A dropdown menu with a preview window.
- User ID:** A dropdown menu with a preview window.
- List scope:** A dropdown menu set to "Open Tasks". Other options include "Unassigned Tasks", "Assigned Tasks", and "Completed Tasks".
- Include assigned users:** A checked checkbox.
- Filter by due range:** An unchecked checkbox.
- Search filter:** A dropdown menu with a preview window.
- Retrieve task list:** A dropdown menu set to "Default Task List Service". Other options include "Dashboards", "Select...", "New...", and "Reset".
- Expand all rows:** A checked checkbox.
- Assigned users:** A dropdown menu with a preview window. Buttons for "Select..." and "Clear" are available.
- Tasks:** A dropdown menu with a preview window. Buttons for "Select..." and "Clear" are available.

- **Height (*Integer*)** – The height of the control measured in pixels. If not supplied, the default value is 600 pixels. If more tasks are available to be shown than can be displayed in the supplied height, scroll bars are automatically added. A special height of "0" causes the control to be tall enough to show all the available tasks.
- **Initial list size (*Integer*)** – The initial number of items to show in the list. The default is all of them.
- **Team ID**
- **User ID**
- **List scope** – The options for this property are:
 - Open Tasks
 - Unassigned Tasks
 - Assigned Tasks
 - Completed Tasks
- **Include assigned users**

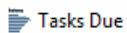
- Filter by due range
- Search filter
- Retrieve task list
- Expand all rows (Boolean) – Show each of the tasks in the list already expanded and hence their details are also shown.
- Assigned users
- Tasks

See also:

- Navigation Controller Control
- Data Type – TaskListData
- Data Type – TaskListProperties

Tasks Due Control

Palette icon:



Canvas appearance:



Configuration parameters:

The screenshot shows the 'Properties' dialog for the 'Tasks Due' control. The 'General' tab is selected. Under the 'Configuration' section, there are several configuration parameters:

- Team ID:** A dropdown menu with options like 'None', 'Team A', 'Team B', etc.
- User ID:** A dropdown menu with options like 'None', 'User A', 'User B', etc.
- List scope:** A dropdown menu set to 'Open Tasks'.
- Search filter:** A dropdown menu.
- Selected range:** A dropdown menu.
- Fire event on selection:** A checkbox.
- Retrieve tasks due data:** A dropdown menu set to 'Default Tasks Due Service'. To its right are three buttons: 'Dashboards' (disabled), 'Select...', 'New...', and 'Reset'.
- Localization service:** A dropdown menu set to 'Dashboards Localized Messages Loader'. To its right are three buttons: 'Dashboards' (disabled), 'Select...', 'New...', and 'Reset'.

Team Roster Control

This control displays a list of users belonging to a team:

Roster

Individual counts are total counts for all teams.



UserA

Assigned Tasks: 1
Tasks Completed Today: 0



UserB

Assigned Tasks: 0
Tasks Completed Today: 0



UserC

Assigned Tasks: 0
Tasks Completed Today: 0

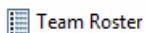
The binding data for this control is a List of TeamRosterEntry. This data type has the following structure:

- userid (String)
- name (String)
- fullName (String)
- jobTitle (String)
- emailAddress (String)
- phoneNumber (String)
- totalOpenTasks (Integer)
- tasksCompletedToday (Integer)

The function called retrieveTeamMemberList() can be used to retrieve a list of TeamRosterEntry structures:

```
var myTeam = tw.system.org.findTeamByName ("Team A");
tw.local.rosterList = myTeam.dashboard.retrieveTeamMemberList();
```

Palette icon:



Canvas appearance:



Configuration parameters:

Properties Validation Errors Where Used

General

Configuration

Visibility

HTML Attributes

Configuration

Title:

Priority team members: [Select...](#) [Clear](#)

Team ID:

Height:

Localization Service: [Dashboards Localized Messages Loader](#) [Dashboards](#) [Select...](#) [New...](#) [Reset](#)

Retrieve team data: [Default Team Member Retrieval Service](#) [Dashboards](#) [Select...](#) [New...](#) [Reset](#)

- Title (String) – The title shown above the roster
 - Priority team members
 - Team ID
 - Height (Integer) – The height of the list. The default is 600 pixels.
 - Localization Service
 - Retrieve team data

See also:

- Data Type – TeamDashboardSupport

Team Summary Control

Palette icon:



Canvas appearance:



Configuration parameters:

Properties Validation Errors Where Used

General

Configuration ▾ Configuration

Localization Service:  Dashboards Localized Messages Loader Dashboards Select... New... Reset

Visibility

HTML Attributes

Expanded Mode: 

Custom Theme: 

Zoom Control

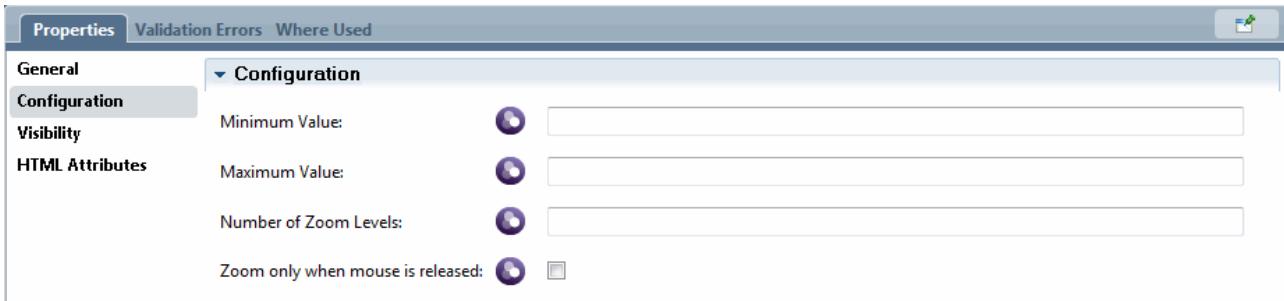
Palette icon:



Canvas appearance:



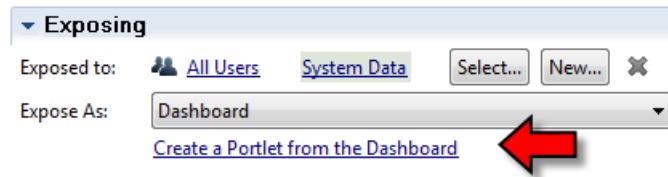
Configuration parameters:



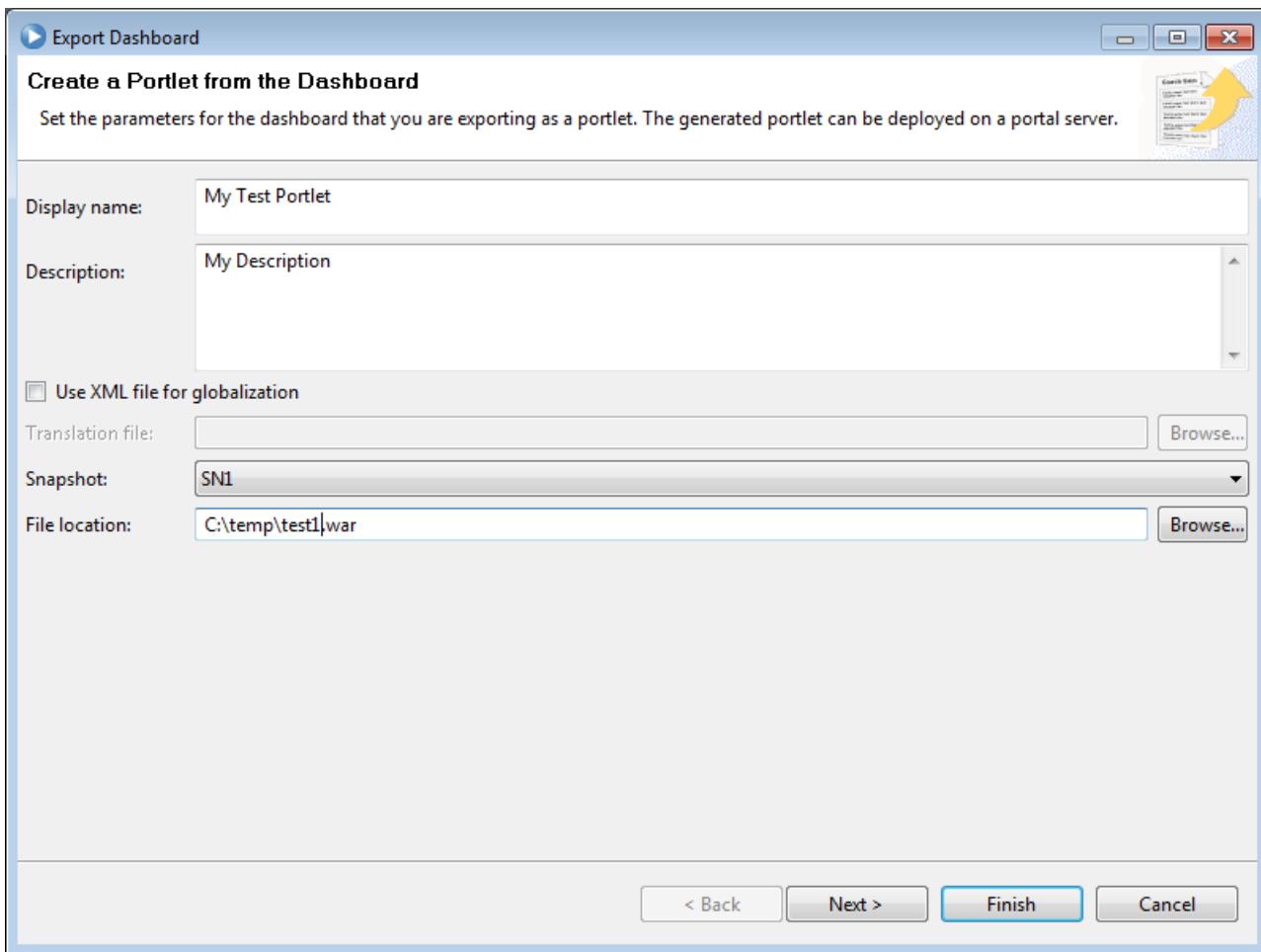
Exposing a Human Service as a portlet

Java provides a standard called JSR-286 which describes how visual components called "Portlets" can be built to be used in products called "Portlet Servers". IBM builds a product called IBM WebSphere Portal. IBM BPM provides the ability to expose certain types of Human Services as JSR-286 compliant portlets for inclusion in such a portal server.

For a Human Service to be used as a portlet, we must define the Human Service as "Exposed As: Dashboard". From there we will find an option to create a portlet:



Before the portlet can be created, the Process App which includes it must be part of a snapshot. Once done, the "Create a Portlet from the Dashboard" link can be clicked. This will launch a wizard used for the creation of the portlet. The screen for the wizard is shown next:



We are asked for the name of the portlet which will be shown to the portal user. We are also asked for a description. Possibly the most important attribute at this stage is the notion of the "File location". A portlet manifests itself as a Java EE Web Archive File (a .war file). We supply the location where the war file will be written.

Once the wizard is complete, we will now find that the WAR file has been generated for us. We are now able to deploy this to our portal server.



LIFERAY

To deploy the WAR to Liferay, we can copy it to the <Liferay>/deploy directory where it will be automatically consumed.

Page Flow/ Screen Flow Solutions

The model of using Coaches with Human Services is that when a Human Service is reached, the BPMN process described by the BPD pauses until the Human Service completes. When the Human Service is reached, a new entry is added to the task list commonly seen in the Process Portal. A user selects the task from the task list and at that point sees the Coach. The user then works on the Coach/Human Service and completes the Coach allowing the process to continue. When the next Human Service is reached, the story repeats including picking the task from the task

list.

This story paints the picture of human interaction as "task" oriented meaning that each Human Service is a discrete tasks that has a beginning and end and, when ended, has no relationship to the next.

In some scenarios, an alternate style of human interaction is desired that is termed "screen-flow". In this style, the desire is for a screen to be shown to the user and, when the user completes the screen, the BPD/BPMN process gets control back, does some further work and then reaches the next Human Service. At this time, a new screen is shown to the user as soon as possible. The effect is a page transition under process control as opposed to a task claim, view, complete, claim, view, complete story.

Unfortunately, this style of process modeling and user interface design is simply not possible with the IBPM product at this time when working with Coaches. A circumvention is to code/design a series of Coaches in a single Human Service such that transition from one Coach to another follows a screen flow. Nested Service calls in the Human Service can then be used to call additional services. The down-side of this approach is that it bypasses all the features of BPDs. This includes:

- Analysis and optimization
- Nested BPDs
- Timer based activities
- Event based activities
- more ...

Let us now drill down to see what would be involved in creating a Human Services style screen flow solution. We will start by assuming that there is a Human Service which contains multiple Coaches wired together perhaps with logic and service calls between them.

To begin, we need a mechanism to start an instance of this Human Service. We will assume the existence of a function called "startInstance". What this will take as input is a description of which service to start. This will be of the form "<ProcessApp>@<ServiceName>".

We can next make a REST call that looks as follows:

```
/rest/bpm/wle/v1/service/<service>?action=start&createTask=true
```

This will start our task instance. The return data from the REST request is also important. It contains a lot of details about the newly started Human Service. Among these, the following fields of information will be necessary to keep around:

- data.step – The name of the first/next Coach to be displayed
- data.key – The TaskId/ServiceId of this instance of the Human Service
- data.data – Data available in the Human Service
- data.actions – The Control Ids of the buttons on the first/next Coach. Sending these in as "action" requests causes the corresponding Coach to believe that the button has been pressed and navigate onwards as appropriate
- data.serviceStatus – The status of the Human Service. While in the middle of a Human Service, this will have the value "coach" while at the end when the Human Service has completed it will have the value "end".

In a screen flow solution, we will be instructed on which screen to show next. We retrieve that

from the “data.step” data. We should then assume that the UI knows how to display the information on such a screen.

When the user has completed the current screen, the UI code will invoke a function I have called “finishUI”. This function takes two parameters:

- The data that should be returned from this screen (if any)
- The name of the “button” that should be assumed to have been pressed

The design of “finishUI” looks as follows:

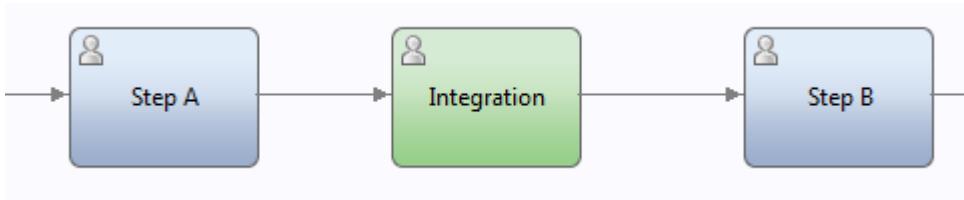
```
if we have return Data then
    REST /rest/bpm/wle/v1/service/<serviceId>?action=setData&params=<jsonParams>
Complete the page using
    REST /rest/bpm/wle/v1/service/<serviceId>?action=<action>
Save the response data
```

Screen Flow using BPD

Let us take a second look at Screen Flow techniques. This time we will look at how this might be achieved at the BPD level. The first thing we have to do is read the section on:

- Automatic Flow

One of the challenges of this techniques has been that virtually anything between two Human Activities on the BPD diagram will cause the auto-flow to break. However, an intriguing opportunity has been noticed. Imagine we have two BPD Human Activities (lets call them A and B) and we wish to perform other work between them. If we place yet another Human Activity between A and B but the Human Service that implements this activity has no Coach UI within it, then screen flow progresses. This means that we can include a variety of service integration calls as nested services.



The activity labeled “Integration” is defined as a Human Activity but its content can include a variety of different integration functions.

User Interface fragments

As experience with IBPM grows, a number of useful fragments and techniques can be captured. Here are a sample of such:

Embedding another HTML page

We can embed an HTML page from another location within the coach. We can achieve this through the use of the HTML iframe tag. Create a Custom HTML component and define its body to looks as follows:

```
<iframe src="http://www.ibm.com" width="100%" height="620px"></iframe>
```

The src attribute names the web page to be shown. The width and height should also be supplied to

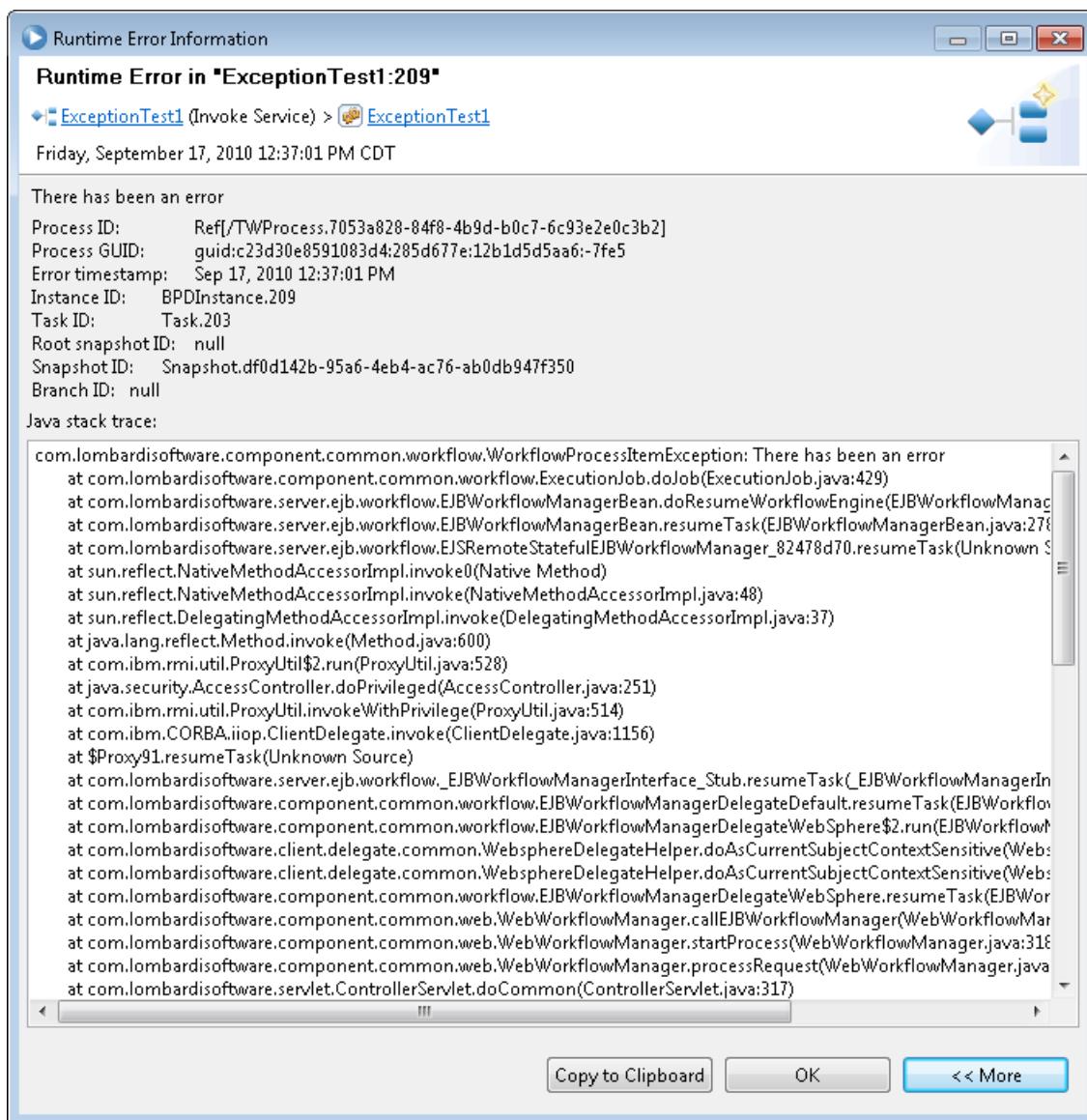
describe the size of the internal frame used to display the content. Here is an example:

The screenshot shows a modal dialog box titled "Lombardi Coach" with a "Section Title" field containing "Section Title". Below it is a "Checkbox:" field with an unchecked checkbox. The main content area displays the IBM homepage. The homepage features the IBM logo, a search bar, and a navigation menu with links like "Home", "Solutions", "Services", "Products", "Support & downloads", and "My IBM". A banner at the top says "Explore real-world change in a virtual game" and includes a link to "Play CityOne now". Below the banner, there are three promotional cards: "Help reinvent tomorrow by playing a game today", "October is National Cybersecurity Awareness Month", and "Introducing premium storage at a midrange price". A news item about EnergyAustralia selecting IBM for a smart grid initiative is also visible. The footer contains links for "Learn about", "Shop for", "Develop & deploy", "Get support", "My IBM links", and "About IBM". On the right side of the footer, there's a graphic for "Find out what HR leaders are saying" with a link to "Register for the study". The bottom of the page has standard footer links for "About IBM", "Privacy", "Contact", "Terms of use", "Accessibility", "IBM Feeds", and "Jobs".

Error Handling

Building a solution where one expects everything to behave correctly is only part of the story. A solution should accommodate error conditions as well.

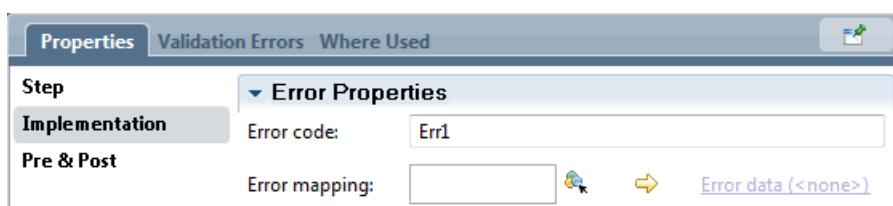
When an error is caught, the `XMLElement` at `tw.system.error` contains some data.



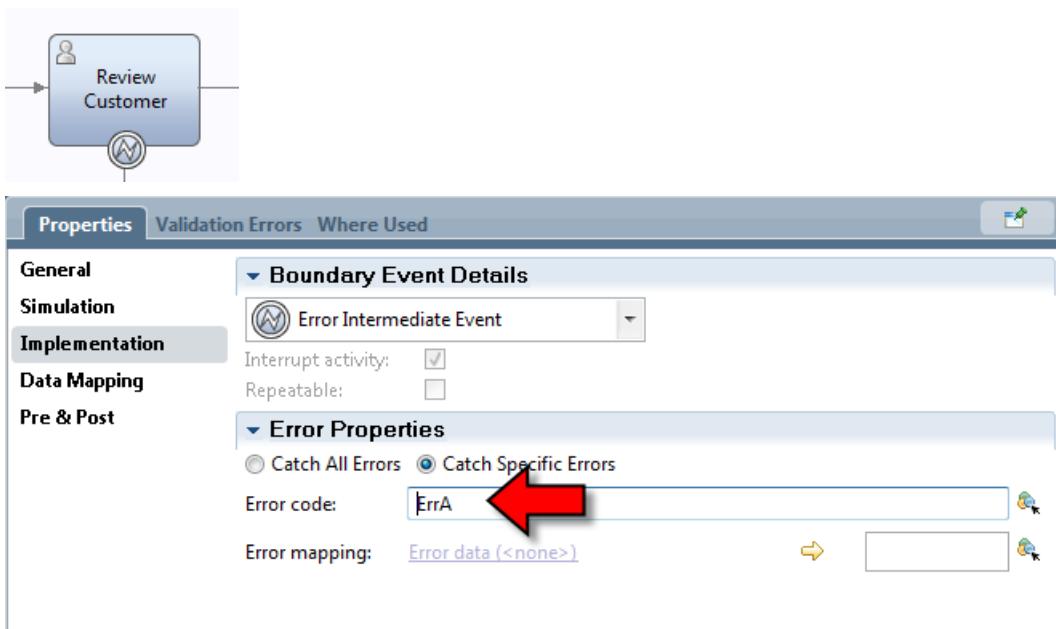
Within a service, we can define an Error End Event. When this is reached, the service comes to an end and an error condition is raised.



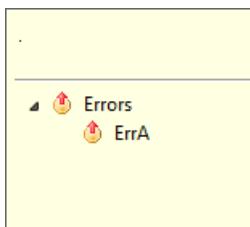
The name or identity of the error condition is supplied in the Implementation tab in the "Error Code" field.



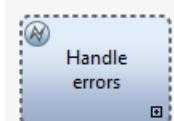
In a BPD where a call might be made to such a service, an Error Intermediate Event can be attached to handle such an error:



A handler can be added to catch a particular named error or a handler can be added to catch all errors. If we can't remember the names of all the errors, we can be provided with a list of possible errors from the selection entry on the far right.



An additional way to handle errors in our environment is to provide an "Event Subprocess". This is a sub-process which will be invoked when an error condition (general or specific) is detected.



See also:

- Error End Event
- Error End Event
- Catch Exception

Error Handling in JavaScript

Code within services that is written in JavaScript can include the try/catch mechanism to surround code that may fail. For example:

```
try
{
    // Some code here
}
catch (myException)
{
```

```
    log.error("Caught an exception: " + myException.message);  
}
```

BPD Events

Events are occurrences that happen associated to the process and which the process should be notified about. There are various types of events associated with the solution. Events are usually triggered by some occurrence. The general occurrence types are:

- An external message arrives
- A time period has elapsed
- An exception has been detected



Start Event – see Start Event



End Event – see End Event



Timer Event – see Timer Events



Start Message Event – see Timer Events



Intermediate Message Event – see Error: Reference source not found



Intermediate Tracking Event – see Tracking Intermediate Event



Start Ad-hoc Event – see Ad-hoc Start Event



Intermediate Exception Event – see Error: Reference source not found



End Exception Event – see Error End Event



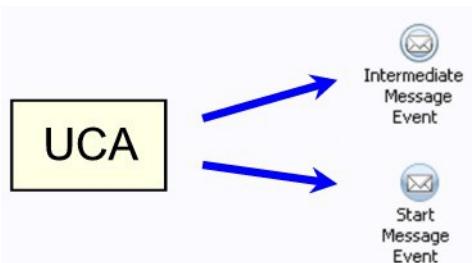
Terminate Event – see Terminate Event

Undercover Agents (UCAs)

An Undercover Agent (UCA) listens for triggers from external systems, Invoke UCA service components or for scheduled time based events. UCAs are also responsible for firing either a Message Start Event or for firing an Message Intermediate Event that is contained within a BPD.

See also:

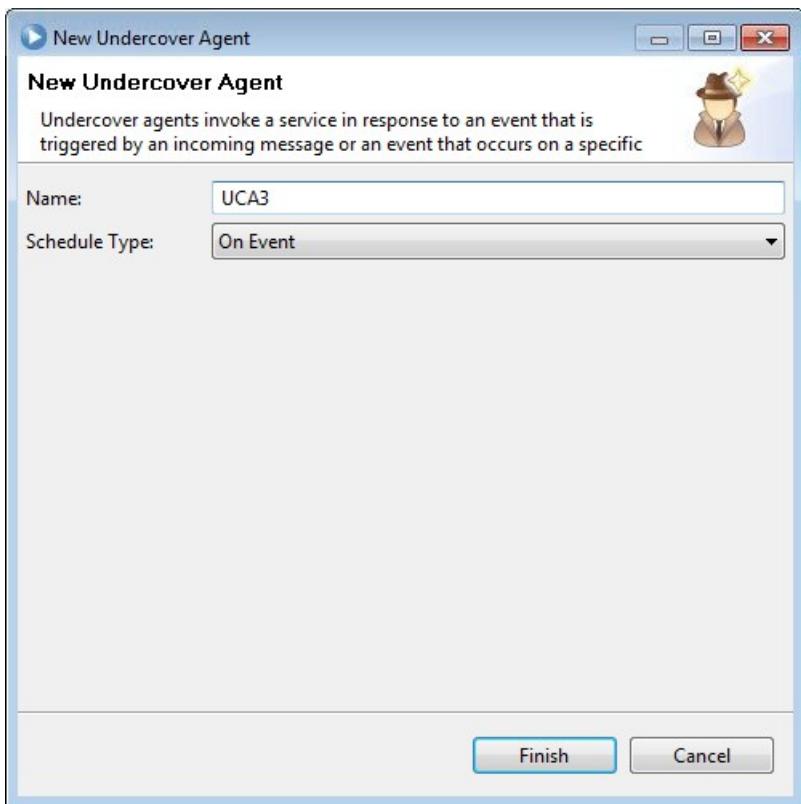
- Message Start Event
- Invoke UCA
- Starting a UCA from REST



A UCA definition is created from the implementation category within Process Designer.



Once selected, a wizard will appear. In this wizard we get to give the new UCA a name and define how it is started.



The "Schedule Type" attribute has two possible options:

- On Event – Fired as a result of an explicit Invoke UCA in a service
- Time Elapsed – Fired on schedule from the run-time

Event Initiated UCAs

When an Event Initiated UCA is defined, it looks as follows:



When a UCA is activated, it will basically represent an event. That event can carry payload data with it. IBPM provides two distinct mechanisms to define the payload data that may be carried with a UCA event. The simplest way is to select "variable".

Details

Queue Name: Async Queue

Implement: Variable Service

Variable Type: NameValuePair System Data Select... New...

Enabled:

Parameter Mapping

Use default [Input \(NameValuePair\)](#)

With that radio button checked, we get to supply a single variable data type. When a UCA is invoked, it will expect an instance of that type of data to be passed along with it. The second option is available when Service is selected:

Details

Queue Name: Async Queue

Implementation: Variable Service ←

Attached Service: [Default...](#) [Event](#) [Syst...Data](#) [Select...](#) [New...](#)

Enabled:

Parameter Mapping

Use default → correlationKey (String)

Use default → value (String)

Now we get to supply the reference to a service (commonly a general service definition). The input data to the general service is then used to build some output data and the output data is what is passed along with the UCA. It is extremely common to have a general service map directly from the input parameters to the output parameters without modification. This allows a UCA to pass more than one parameter.

Once created, the UCA definition will look as follows:

Undercover Agent

Common

Name: UCA2
Modified: admin (Nov 30, 2012 7:09:24 PM)
System ID: guid:56d35d2f221c8d0e:-3cac44a9:13b35102749:-7e9f
Documentation: [\(Edit\)](#)

Scheduler

Schedule Type: On Event
Event Marker: [Message](#) Select...
[Run now](#) Add Event Subscription...

Details

Queue Name: Async Queue
Implementation: Variable Service
Attached Service: [Default...](#) [Event](#) [Syst...Data](#) [Select...](#) [New...](#)
Enabled:

Parameter Mapping

Use default → correlationKey (String)

Use default → value (String)

Event

Event Message: d7b999b6-3b8d-4720-b4f0-d02086c005b9

The Enabled check box can be used to enable or disable the execution of the associated service when an event arrives.

In the Event section, we see a correlation value called Event Message that is used to associate an incoming event with the instance of a UCA that is watching for that kind of events. The majority of times this can be ignored. This is of particular importance when using JMS to initiate a UCA. See: Java Message Service – JMS.

When an event arrives and triggers a UCA, the UCA in turn will execute an IBPM service. This is

specified by the `Attached Service` attribute and is commonly an IBPM General Service. The way that the UCA causes the service to be called is not by invoking that service explicitly, instead, the UCA queues the request to start the service on a logical queue. It must be stressed that this is indeed a logical queue and has no relationship to JMS or SI Bus. The queuing mechanism here is used to serialize the execution of UCA originated events. When the UCA places a request on a queue, any further requests placed on that queue are not processed until the previous events have completed which means that the associated services have completed. One exception to this is the special queue called the `Async Queue`. If the UCA start service request is placed on this queue, further requests to start services will **not** wait for previous services to complete.

The number and names of these queues can be changed in the IBPM Process Admin Console.

Schedule initiated UCAs

A schedule (time based) UCA has an additional set of attributes:

The Time Schedule defines recurring periods of when the UCA should fire. The Time Schedule is quite flexible in its configuration. Each time the schedule is reached, the UCA fires.

The icon for a Timer Event UCA looks as follows:



Within Process Center there is a Process Server used for testing. It doesn't make much (if any) sense for a UCA to fire for each of the schedules held in the repository in the Process Center Server. As such, they simply don't. For testing, the Run now button can be used to explicitly fire an instance of the UCA.

When a UCA is scheduled, we can see it in the Process Admin Console under Event Manager > Monitor:

Event Manager > Monitor

Scheduler ID	Status	Connect expiration	# Jobs Executing
win7-x64		Jul 10, 2011 11:42:55 AM	0

Total Jobs Executing: 0 Total Jobs: 2

Scheduler	Process App / Toolkit	Snapshot	Job Name	Job Queue	Scheduled Time	Last Scheduled Time	Last Execution Time	Next Scheduled Time	Job Status
	Process Portal	7.5.0	Execute UCA Periodic SLA Update, on set schedule	SYNC_QUEUE_1	7/10/11 11:45:00 AM	7/10/11 11:30:00 AM	7/10/11 11:30:00 AM	7/10/11 12:00:00 PM	Scheduled
	UCATests	SN7	Execute UCA ScheduledUCA, on set schedule	Async queue	7/10/11 11:45:00 AM	7/10/11 11:40:00 AM	7/10/11 11:40:00 AM	7/10/11 11:50:00 AM	Scheduled

See also:

- Process Scheduling with Job Scheduler

UCAs and queued events

Consider a UCA firing which results in an event being generated for processing by either a Start Message Event or an Intermediate Message Event.

Disabling UCA processing

Within the Process Admin Console under the Deployed Apps section, we can see lists of the UCAs associated with an application.

From here we can selectively enable or disable their processing.

UCAs and Toolkits

One common usage pattern for UCAs is to have them used in a publish/subscribe model. This means that a publisher publishes an event by invoking a UCA. Any interested subscribers then listen on that UCA to be triggered if and when a UCA is invoked. The implication of this is that both the publisher and subscriber must share the UCA definition. If the publisher and subscriber

are in different Process Applications then the UCA definition must be stored in a toolkit that is common to both Process Applications. Care should be taken that UCAs added to Toolkits be **only** event processing UCAs and should **not** be time based UCAs. This technique is a powerful solution to having one BPD invoke another BPD when both BPDs are in different process apps. There is no known alternate solution to that puzzle.

Token Management

As a process instance executes, it can be thought of as having a current step that is being processed. If we were to draw out our process on a sheet of paper, we could imagine placing a pebble on the current step being executed. We would think of this pebble as a "token" to indicate which step has control. The IBPM product exposes the concept of tokens to indicate where within the process the current step may be found. The REST API exposed by the product provides a set of commands that can be used to interrogate the process to find the current token, a command to move a token from one step to another and a command to delete a token from the instance. When we combine these capabilities we have the notion of being able to achieve some interesting effects.

By moving a token from one step to another, we effectively have fine grained control over the flow of the process. Specifically, we can jump backwards or forwards within the process. Using this technique we can effectively jump to a previous step and have execution continue from there. Alternatively, we can jump forward skipping current steps and end up at some future step.

Before we go one to discuss these ideas further, let us introduce a note of caution. A process instance is characterized by a number of item. Most important of these are the states of variables contained within the process. Steps within a process change or modify the values of these variables. The output of a previous step may be the population of a variable which is essential for the correct operation of a subsequent step. What this means is that if we move a token forward, steps which would have been responsible for variable population are skipped with the result that the variables are not properly set. When a step is reached which relies on these values, the step may fail to operate as we expected.

In addition, jumping around within a process subverts the design intent of the process designer. A process is a sequence of activities that when executed in order, achieve a well defined business outcome. Jumping around within a process effectively breaks that design. This could result in technical failure of the process or, worse, it could result in erroneous outcomes. For example, if we re-run a step which bills a customer's credit card, the result would indeed be a double billing. If we skip over a billing step, then the customer may not be billed at all. Neither are acceptable outcomes for the process as a whole.

With these notions in mind, the idea of moving a token within a process has to be carefully considered. It should rarely be considered a "go-to" practice and should only be used by expert process designers and even then with caution.

With those disclaimers and caveats in place, let us now look at how this technology can be used.

At the core there is a REST API called "Move Token" which has the following syntax:

```
POST /rest/bpm/wle/v1/process/{processInstanceId}?action=moveToken&tokenId={tokenId}&target={stepId}
```

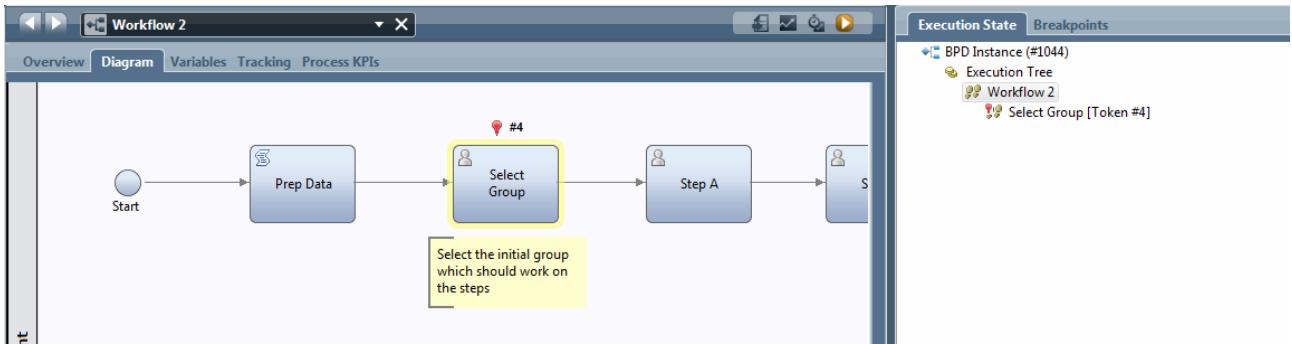
This is quite a technical looking command so let us break it down.

In effect, it is a function call that takes three parameters:

- `processInstanceId` – This is the identifier of the process instance within which a token will be moved. Obviously if we are jumping from one step in the process to another, there has to be an instance of the process within which we are jumping.
- `tokenId` – This is the identifier of the token that is going to be moved. More will be said on this soon.
- `target` – This is the step within the process to which the token will be moved. Again, more will be said on this shortly.

When we run a process within Process Designer and look at it within the Inspector view, we see

diagrams that look as follows:



Notice the marker. This is a visual indication of the token within the process. Each token has an identifier associated with it that we call the tokenId. It should be noted that this is not a computable nor fixed value. When a token is at one step it will have a certain tokenId value but as soon as it moves to another step, the token will have a completely new tokenId. It appears in fact to be an illusion that we are "moving tokens". Instead a token is the marker showing which step we are at and when we say we are moving a token, what we really appear to be doing is "ending" the current token and creating a "new"token (with a new tokenId) at the next step. Despite this subtlety, it appears that we can simply think of the token as having moved and still hold our model cleanly. Just ensure that you realize that a "token" doesn't have the same identifier throughout its life.

How then can we find the current token within a process instance?

Again, another REST command comes to our aid.

```
GET /rest/bpm/wle/v1/process/{processInstanceId}
```

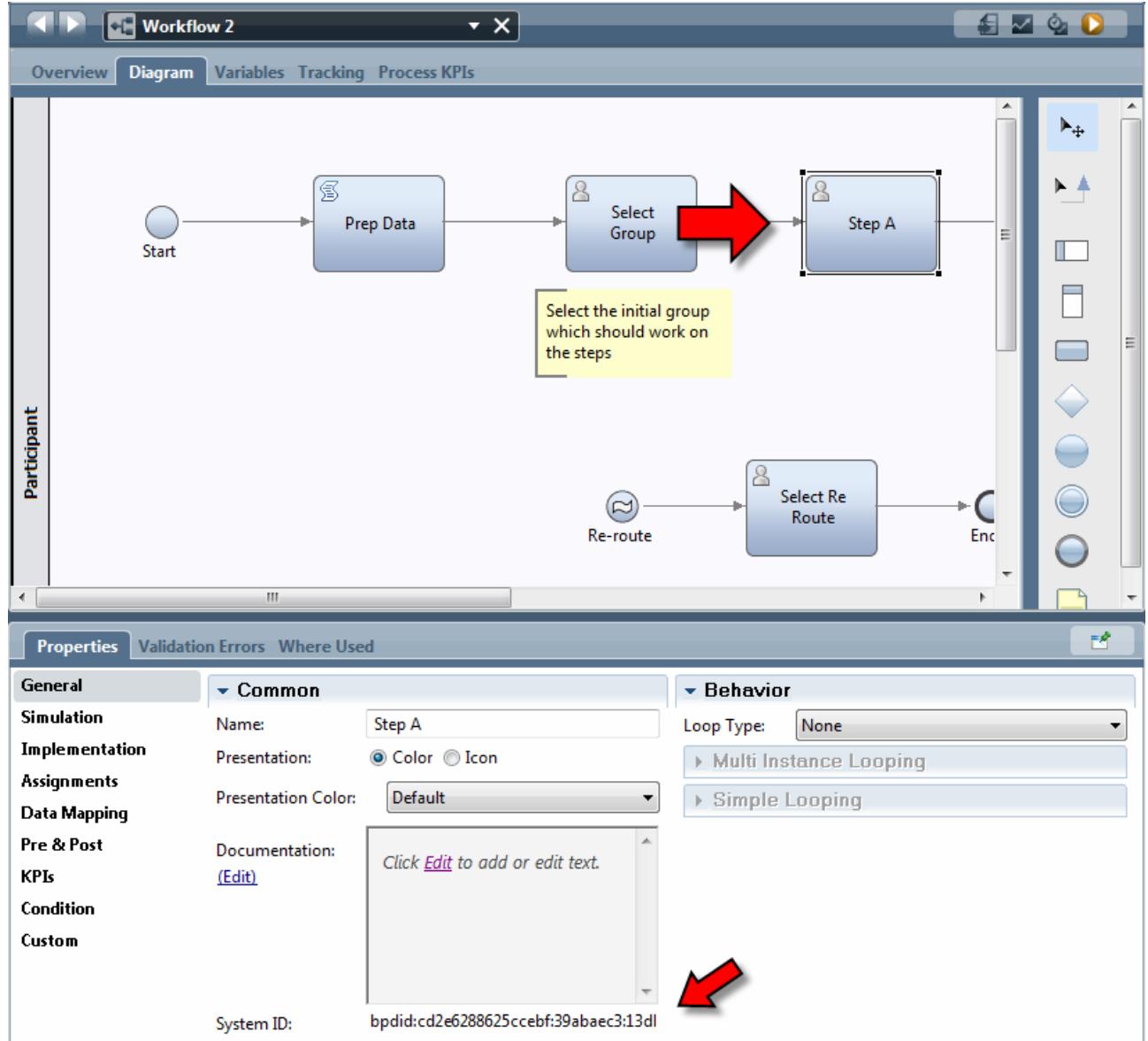
This function returns a detailed data structure describing the process supplied by the `processInstanceId`. Contained within this data structure is a sub-structure called the "`executionTree`". This structure lists all the steps within the process instance which currently have tokens associated with them. The details of the structure will not be described here and can be found elsewhere. However, the astute reader will have noticed that I attempted to sneak something into our story. I suddenly started to talk about token existence within a process in plural form.

Yes ... this is correct and was not a mistake. Within any given process instance, multiple steps within it may indeed be available to be executed concurrently and hence each of those steps have their own associated token. Because of this, when we are working with process instances and are looking for the identity of a particular token to move, we need to be cautious that a process instance may have multiple tokens and not be fooled into simply taking the first token we come across.

Let us recap. In order for us to move a token within a process, we need to know the `processInstanceId` of the process instance that is going to be manipulated. Knowing the `processInstanceId` we can then ask for the `tokenId` for the token that we wish to move. What remains is to define the step within the process which should be the target of the move. At the conclusion of the move, the token will now reside at that step.

When we look at a process diagram, we see that steps have nice human-readable labels associated with them. In principle, we could use those labels as the identity of a step but the reality is that is a bad idea. If we coupled our logic for moving processes to what a designer chose to name a step on a particular day, we will quickly become broken if someone should simply rename the step. Putting it simply, we don't want the process designers to break our solution if they change a step name nor should we wish to constrain them from changing those names. What we need is some unique identity for a step in a process that isn't going to be related to its label. Fortunately, there is exactly such a thing. If we look in Process Designer at a particular step, we see that in its General

Properties tab, there is an entry called "System ID". This weird and wonderful code is assured to be unique and distinguishes this step in this process model from any other step in any other process model. Note that to actually see this field in the tooling, you have to enable the "advanced options" within the tool.



With this notion in mind, we can return to the last parameter of our Move Token function. The last parameter which is the target of the token move is exactly an instance of this kind of identifier for a step. Note that if we wish, there is another REST API that will allow us to list all the steps in a process model. This list contains their names as well as their stepIds. From this information, we can present the user a list of possible targets instead of hard-coding.

Now we have all the piece parts we need to achieve the token move functions.

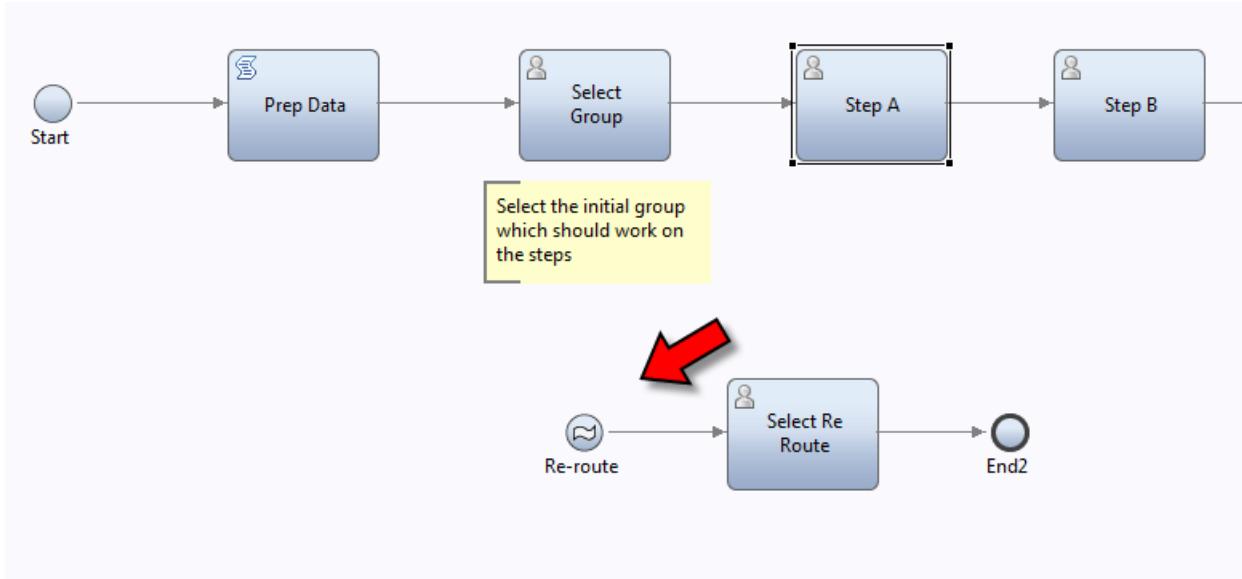
Let us now turn our attention to the semantics of moving tokens. If we move a token from a human activity, that human activity is canceled. If we move a token to an activity, it is as though we had just reached that activity through normal flow means and it is executed appropriately.

Having now given a brief overview of how to move tokens within a process, let us now look at when and why we would use such a technique.

When we build processes, there are a variety of patterns that we would like to accommodate. Once

such pattern is that we may wish to correct an in-flight process instance. Perhaps we have executed a sequence of steps only to realize that a change has been made in our assumptions. Perhaps a customer has called in to amend an existing order or an application for a mortgage has to be reworked because the individual's circumstances have changed. We could always abandon the process and start it again but in certain cases, that may not be the best result.

Within a process we can model an "ad-hoc" entry point. This means that while the process is running, we also have the ability to asynchronously execute steps within its context as a result of an external event. Here we see a process which has an ad-hoc entry point.



When this entry is called, a human service is presented to the user. Within that human service we can design the logic to choose the current token within the process as well as ask choose the step in the process that the token will be moved to. Example screens for this might look like:

Select source token and target step

Selected process is the template of "Workflow 2" width instance id of 1044

Now we pick the token within the process to be moved.

Token selection

Step Name	Token Id (Information only)
Select Group	4

Now we pick the target step in the process

Potential Target Steps

Step Name	Step Id (Information only)
Prep Data	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:70f
Select Group	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:718
Step A	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:6dd
Step B	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:6e8
Step C	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:6f3
Step D	bpdid:cd2e6288625ccebf:39abaec3:13db831b987:6fc

Here is a list of the previous tasks executed by this process.

Previous Tasks

Name	Status	Task Id	Completion Time	Step Id
Select Group	Received	1709		bpdid:cd2e6288625ccebf:39abaec3:13db831b987:718
Total: 1	1			

[« Back](#) [Refresh](#) [Execute](#)

Here are shown the list of current tokens and the set of potential target steps. Selecting these appropriately and clicking execute will move the process accordingly.

See also:

- Moving a token within a process using REST

Security

IBPM leverages both its own model of users and groups as well as leveraging the underlying WebSphere Application Server security sub-systems. User administration of the IBPM private security is performed through the IBPM Process Admin Console in the User Management section (see: Process Admin - User Management).

Unless you are running a pure sand-box development environment or simply are not concerned about security, one of the first tasks that should be performed against IBPM after installation is changing the passwords for the default userids. IBPM documents the default passwords and until those are changed from their default values, the system must be considered completely in-secure. An IBPM TechNote was written on the subject:

- [1448216](#) - How to Change the Default User Passwords (including tw_admin) on WebSphere Lombardi Edition

Some of the passwords needed for operation are stored in encoded formats in files on the file system within the sub directories associated with the IBPM product configuration. These passwords are not encrypted but rather "re-arranged" so that a simple eyeball of the content of the file will not expose their values. If an administrator needs to know the value of the encoded password, a web site on the Internet can be used to enter the value and return the plain text representation. Note that nothing is known about this web site other than its existence so users beware:

<http://www.sysman.nl/wasdecoder/>

See also:

- TechNote - [WebSphere Lombardi Edition Access Control Settings Overview](#) – 2011-04-20

Security Groups

Group management of the IBPM supplied private group system is performed in the IBPM Process Admin Console under the User Management > Group Management section.



A new group can be created by clicking upon the New Group link.

User Management > Group Management

Select Group to Modify: []

New Group Remove

A screenshot of a web-based application interface titled "User Management > Group Management". At the top left, there is a text input field labeled "Select Group to Modify:" with a small dropdown arrow icon. Below it is a button labeled "New Group" with a red arrow pointing to it from the left. To the right of the "New Group" button is a "Remove" button. The main area shows a list of groups. The first group in the list is highlighted with a green background and has a small user icon next to its name. The name "Test Group 1" is visible in the list. To the right of the list, there is a "Remove" button.

A dialog appears in which the name of the new group can be entered:



Once created, user members can be added to the group:

User Management > Group Management

Select Group to Modify: Tes

New Group Remove

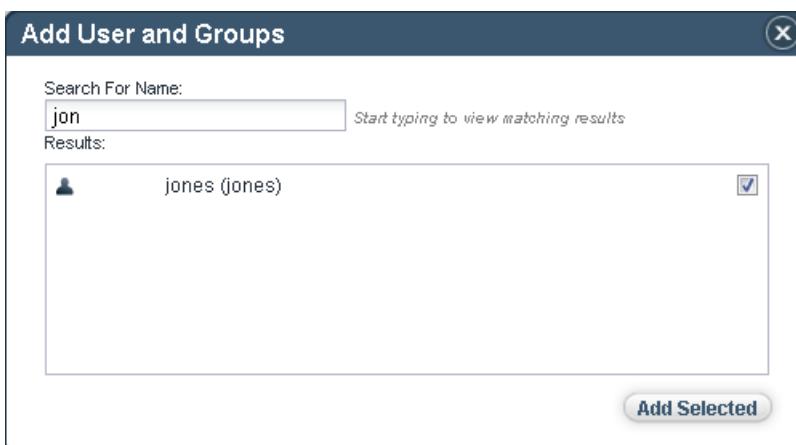
Test Group 1

Team Manager Group: No Team Manager Group

Add Members Remove

A screenshot of the "User Management > Group Management" interface. On the left, there is a list of groups. One group, "Test Group 1", is selected and highlighted with a green background. To the right of the list, there are two input fields: "Team Manager Group" with the value "No Team Manager Group" and "Add Members". A red arrow points to the "Add Members" button.

A dialog appears into which users can be selected and added:



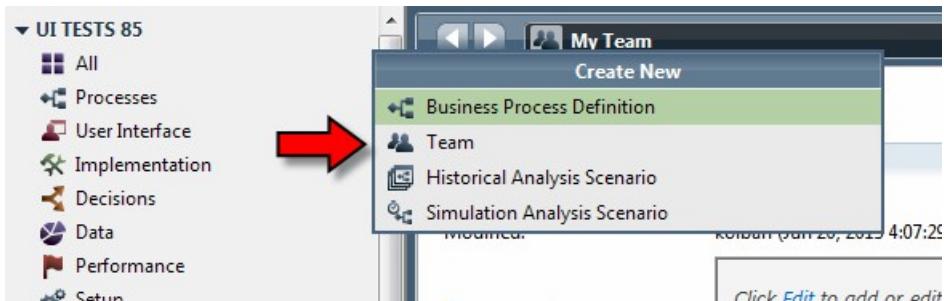
Existing groups can be listed by typing in "spaces" or the prefix of the names to be shown.

See also:

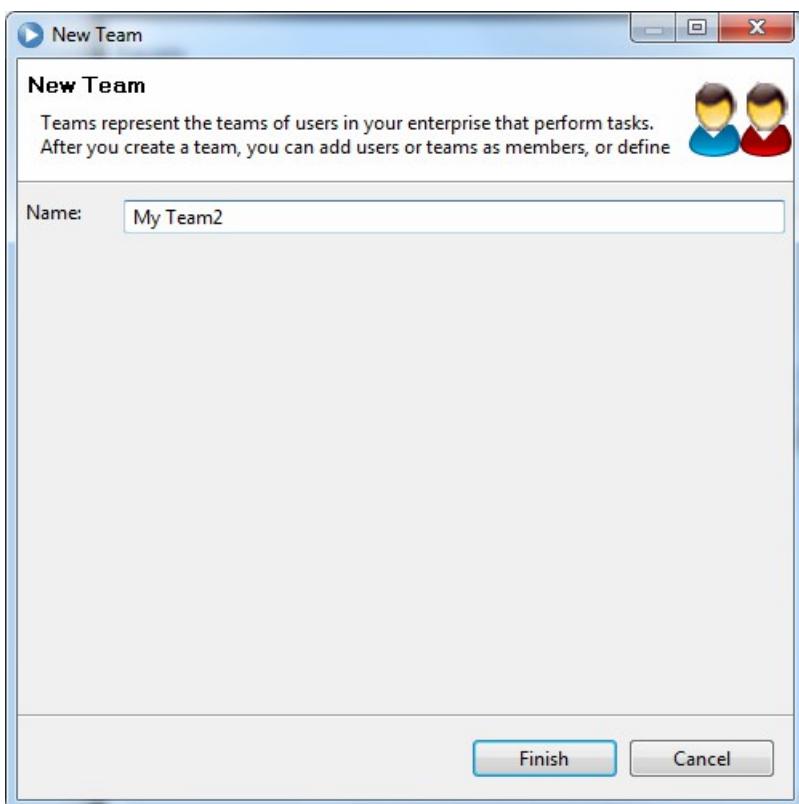
- Data Type – TWRole

Teams

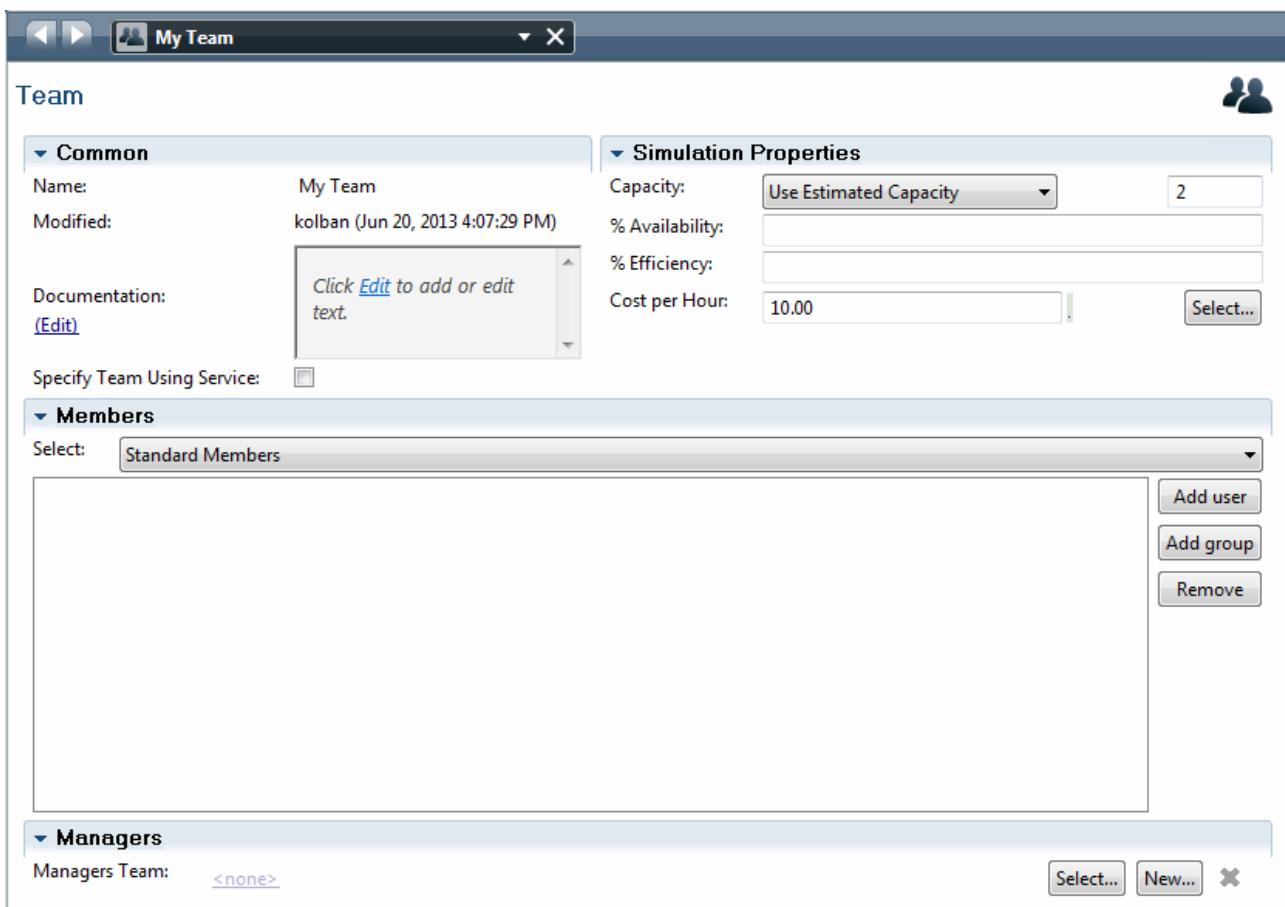
A team is used to represent a set of users in your organization or enterprise locally within an application. Teams can be found in the Processes category within PD. The members of a Team are local to the application in which it is defined. Security Groups provide a global mechanism to define groups. It is key to understand that a Team is a logical group that does not exist outside of the concept of the Process App in which it is defined.



During the creation of a new Team, the name desired for the new group is prompted:



Once created, the details of the new Team can be seen. When newly created, a Team has no members associated with it.



Within the Members section is where we define who belongs to the team. The Members selection criteria can be one of three types:

- Standard Members
- Using Expression (Deprecated)
- System (Deprecated)

Using the Standard Members selection, the members of the participant group are taken from the users and groups defined in the Process Admin Console or from the underlying WebSphere Application Server security system.

Since the other two types of Member selection are deprecated, we will no longer discuss these.

An alternative mechanism to define who are the members of a team is to click on the "Specify Team Using Service" check box. If selected, then the result of executing a BPM service will be used to determine who the members of the team are. The service has to have a specific signature and can be built from the "Team Retrieval Service Template".

This service has an input called "name" which is the name of the team who's members are to be determined. The service returns an output called "team" which is an instance of the Team business object. This defines who will be members of the team.

As well as supplying the members of a team, you can also determine who the manager or managers of the team will be. This is itself achieved by defining another team that can contain one or more members who will be considered the managers. From there you can then name that management team.

The screenshot shows the 'Team' properties dialog box. The 'Common' tab is selected, displaying details like Name (My Team), Modified (kolban (Jun 20, 2013 6:17:28 PM)), and Documentation. The 'Simulation Properties' tab shows Capacity set to 'Use Estimated Capacity' (2). The 'Members' tab lists 'Standard Members' (kolban) with buttons for Add user, Add group, and Remove. The 'Managers' tab shows 'Manager Team' (Manager Team) with buttons for Select..., New..., and X. A red arrow points to the 'Manager Team' link.

When a Task is added to a BPD, that task is added to a lane within the diagram. A default Team can be associated with that lane. Unless otherwise specified, that task will be associated with the members of the associated Team at run-time. Selecting the lane and viewing the properties allows us to select a Team:

The screenshot shows the 'Properties' dialog box for a lane. The 'General' tab has 'Name' set to 'Team' and 'Presentation Color' to 'Default'. The 'Behavior' tab shows 'Default Lane Team' set to 'All Users' and 'System Data' with buttons for Select... and New... A red arrow points to the 'Is System Lane' checkbox.

From within the Process Admin Console, the set of users associated with a Team can be changed at run-time. To achieve this start the Process Admin Console and select Installed Apps:

The screenshot shows the IBM Process Admin Console interface. The top navigation bar includes 'Server Admin', 'Process Inspector', and 'Installed Apps' (with a red arrow pointing to it). The top right shows the user is logged in as kolban@us.ibm.com with links for Preferences and Log out. The main content area has a sidebar with links for IBM BPM Admin, User Management, Monitoring, Event Manager, Admin Tools, and Saved Search Admin. The central area contains text about the Process Admin console and a 'Process Status Summary' dashboard. The dashboard shows a summary for Today: 0 Started, 0 Completed, 0 Late, 0 Executing, and 0 Failed. Below the dashboard, it says 'Process Applications <none>'.

A list of the installed application is shown. Select the application that contains the Participant Groups that are to be changed:

Lombardi | Process Admin Console

Server Admin **Installed Apps** Logged in as

Sort Snapshots By: Application Name ▾ All | Active | Default

TEST1 (TEST1) - SN1 ← Active
BPD2 - 0 instances

From there, a list of the Participant Groups can be found with their associated members within the Role Bindings section. Members can be added or removed.

Lombardi | Process Admin Console

Server Admin **Installed Apps** Logged in as tw_admin | Log out ?

TEST1 (TEST1) - SN1 (Active) Exposing **Role Bindings** ← Event Vars

Participant Group	Members	Action
All Users	1 Members, 0 Tasks	Add Users and Groups
pg_jones	1 Members, 0 Tasks	Add Users and Groups
pg_kolban	1 Members, 0 Tasks	Add Users and Groups

Deactivate Application
Migrate Inflight Data
Sync Settings
Make Default Version
Send Tracking Definitions

About Role Bindings...
During development, process authors designate the participant groups for exposed items. After a snapshot is installed on a Process Server in a different environment (test or production), you may need to add or remove users in those groups. In this page you can manage the runtime members of each participant group. You can also see each exposed item to which a group has access as well as the type of access.
[Configuring runtime participant groups](#)

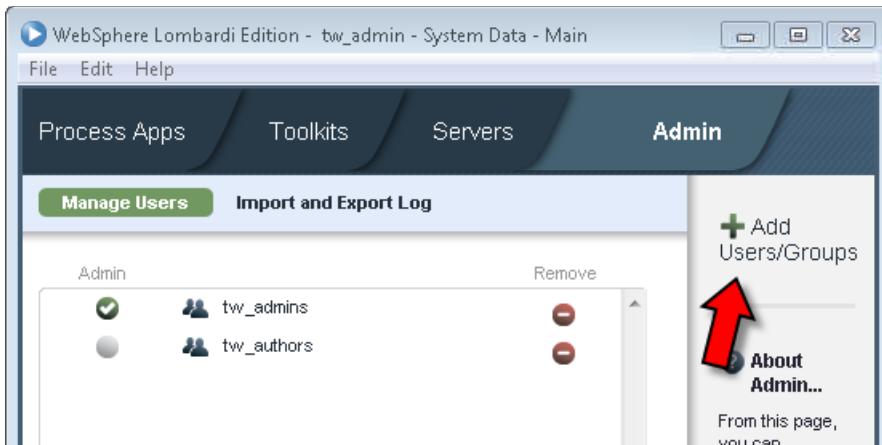
IBM.

See also:

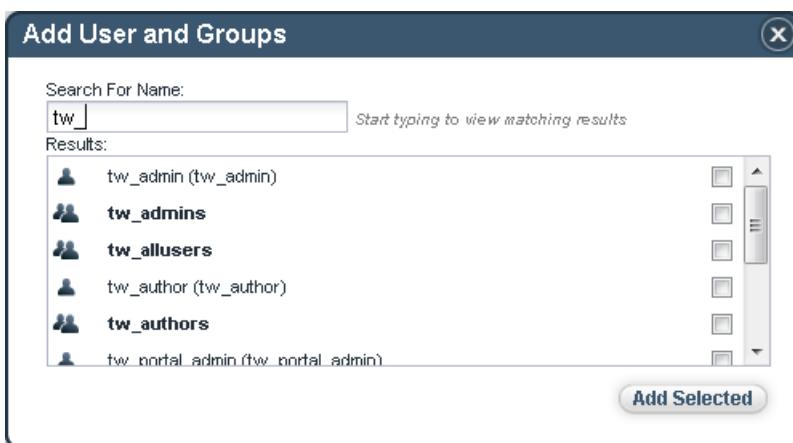
- Pools and Lanes
- Security Groups
- Data Type – TWParticipantGroup
- Data Type – Team
- Assigning Activities – Mapping staff to work
- Vimeo - [IBM BPM 8.5 Teams](#)

Securing Access to the repository

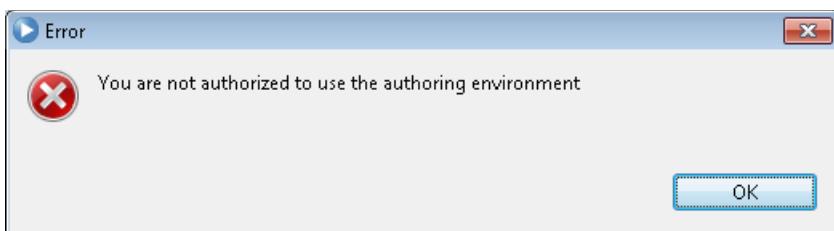
The Process Center repository is secured as it hosts the projects within the enterprise. We typically do not want just anybody to either see or make changes to our Process Applications. In the Admin section of the Process Center console, we can select to Manage Users. From here we can add new users and groups to the list of authorized users.



When the Add button is clicked, a dialog appears into which the additional users and groups can be added:



If a user wishes to login through PD, the user must be authorized to login or else the following error will be produced:



Notice that users and groups of users can also be removed from Process Center access. It is **vital** that this be done with care. If no users are defined to have Admin rights to Process Center then you have effectively locked yourself out. Simply put, don't do this. Take great care that this does not happen.

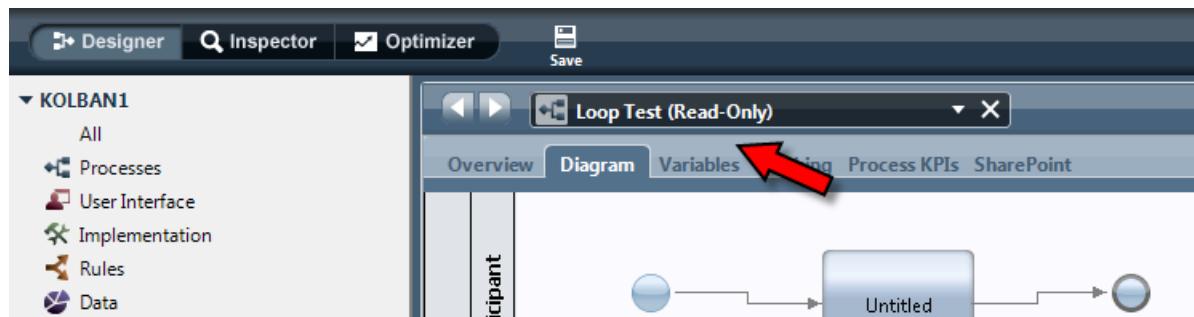
Securing development of a Process Application

Granting a user authority to access the Process Center allows them to login through the PD but this does not mean that they now have privileges to work on or even see **all** the Process Apps in the environment. Process Apps and Toolkits are individually controlled with their own access control lists. From within the Process Apps section of the PD or the Process Center Console, an application can be selected and the Manage tab selected. In there we have a section into which users and groups may be associated. These define the permissions for those entities.

	Read	Write	Admin	
tw_admin (tw_admin)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="Remove"/>
kolban (kolban)	<input checked="" type="checkbox"/>	<input type="radio"/>	<input type="radio"/>	<input type="button" value="Remove"/>

There are three roles a user or group can have:

- Read – This allows a user or group to see the project and see the artifacts within it. The read role can not be removed without removing the user or group association completely. If a user or group is **not** associated with the a Process App then the user or group has no authorities on that application. If an artifact is opened and the user only has Read authority, the artifact is flagged as Read-Only in the editor.



- Write – This role allows the user or group to update or add artifacts into the Process Application.

- Admin – This role allows the user or group to administer the Process Application.

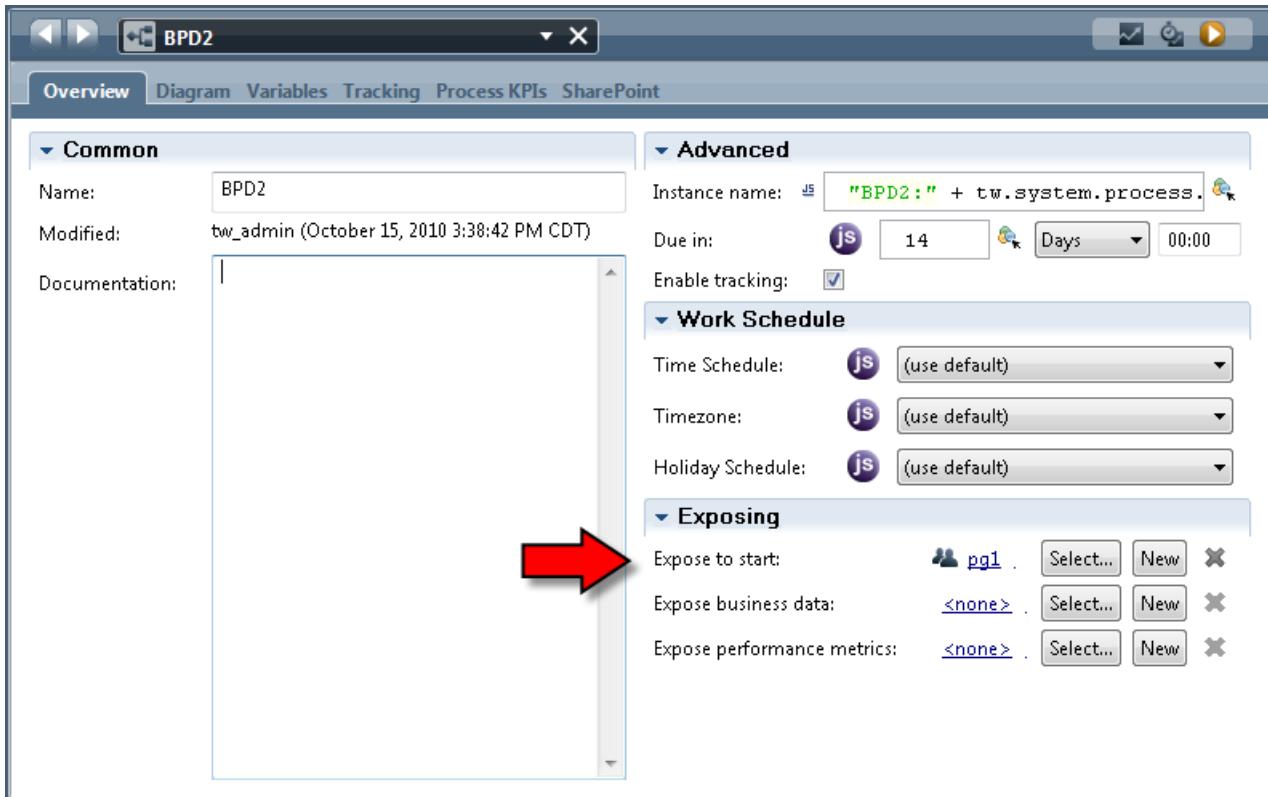
See also:

- Process Admin - User Management

Securing the ability to start an application

Typically, we do not wish just any arbitrary user to be able to start a new instance of a process application. Instead, we want to allow only sufficiently authorized users. In the Overview page of a BPD we can find a section called Exposing.

In there is an attribute called "Expose to start". This can be set to the name of a participant group. Only the members of this participant group will be allowed to start a process. If a user is not in this group, they will not be allowed to start the process:



Securing ability to work with tasks (Human Services)

An activity in the BPD has a Routing section. Within there, the names of the potential staff members who can work on the task are defined. If a user is in this list, then the task will appear in their process portal page and they can work upon it. If they don't appear in this list, then they can't work upon it.

Securing Process Portal capabilities

Many of the tasks of interacting with IBPM are performed through the Process Portal. We wish to control who can do what through this interface.

IBPM exposes a set of configuration parameters that define the role a user must have in order to be able to perform a function. The different functions available for configuration are:

Capability	Description
ACTION_ABORT_INSTANCE	Terminate an instance of a process.

ACTION_SUSPEND_INSTANCE	Suspend an instance of a process.
ACTION_RESUME_INSTANCE	Resume a previously suspended instance of a process.
ACTION_ADD_COMMENT	Add a comment to an instance of a process.
ACTION_ADD_HELP_REQUEST	Create a help request associated with this instance of a process.
ACTION RESPOND_HELP_REQUEST	Respond to a help request previously raised against a process.
ACTION_ASSIGN_TASK	Assign the task to the current user making the request.
ACTION_ASSIGN_AND_RUN_TASK	Run a task and assign it to the current user.
ACTION_REASSIGN_TASK	Return the task to the original group to which it was originally assigned.
ACTION_REASSIGN_TASK_USER_ROLE	Assign the task to a different user or role.
ACTION_CHANGE_TASK_DUEDATE	Change the due date on a task.
ACTION_CHANGE_INSTANCE_DUEDATE	Change the due date on a process instance.
ACTION_CHANGE_TASK_PRIORITY	Change the priority of a task.
ACTION_MOVE_TOKEN	Skip the current task in the BPD and move to the next one.
ACTION_DELETE_TOKEN	Delete token.
ACTION_INJECT_TOKEN	Start an ad-hoc event. Note that by default, only admin users can start ad-hoc events. This trips up a lot of folks.
ACTION_VIEW_PROCESS_DIAGRAM	Ability to view the process diagram.
ACTION_VIEW_PROCESS_AUDIT	Ability to view historical data about process variables.
ACTION_CHANGE_CRITICAL_PATH	
ACTION_ADD_DOCUMENT	
ACTION_UPDATE_DOCUMENT	
ACTION_DELETE_DOCUMENT	
ACTION_DELETE_INSTANCE	
ACTION_FIRE_TIMER	
ACTION_RETRY_INSTANCE	
ACTION_SEND_EVENT	

The configuration for these parameters used to be achieved via XML customizations and the instructions for that are shown later. However from 8.5 onwards, wsadmin scripting can be used to achieve this task. There is an AdminConfig object called "BPMPolicyActions" that has methods on it to list, set and get values. This will probably be used to provide scripted or programmatic access. Fortunately, IBM has supplied a sample script to work with these options. The script can be found at <ROOT>/util/Security/BPMSecurityConfig_sample.py. Asking it for its usage shows:

Usage: Use this script to get/modify the configured security properties.

```
-E|--de DE_name -option
      -g|--get property_name
      -s|--set property_name , new_value
      -a|--add console_property_name , constraint_value
          |action_policy_name , role to be added
      -r|--remove console_property_name , constraint_value
          |action_policy_name , role to be removed
```

An example of running it would be:

```
C:\IBM\BPM\v8.5\util\Security> wsadmin -f BPMSecurityConfig_sample.py -lang jython -user kolban
```

```

-password password -E Del -g ACTION_ABORT_INSTANCE
WASX7209I: Connected to process "dmgr" on node Dmgr using SOAP connector; The type of process is:
DeploymentManager
WASX7303I: The following options are passed to the scripting environment and are available as
arguments that are stored in the argv variable: "[-E, Del, -g, ACTION_ABORT_INSTANCE]"

Current value for property ACTION_ABORT_INSTANCE in DE Del is:
tw_admins

```

The configuration settings for these capabilities are made in the 100Custom.xml file. The default settings can be found in the 99Local.xml file found in:

```
<Install>/profiles/<profileName>/config/cells/<cellName>/nodes/<nodeName>/servers/<serverName>/process-center/config/system/99Local.xml
```

The default section in this XML file looks as follows:

```

<portal>
  <default-action-policy>
    <action type="ACTION_ABORT_INSTANCE">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_SUSPEND_INSTANCE">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_RESUME_INSTANCE">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_ADD_COMMENT">
    </action>
    <action type="ACTION_ADD_HELP_REQUEST">
    </action>
    <action type="ACTION RESPOND_HELP_REQUEST">
    </action>
    <action type="ACTION_ASSIGN_TASK">
    </action>
    <action type="ACTION_ASSIGN_AND_RUN_TASK">
    </action>
    <action type="ACTION_REASSIGN_TASK">
    </action>
    <action type="ACTION_REASSIGN_TASK_USER_ROLE">
    </action>
    <action type="ACTION_CHANGE_TASK_DUEDATE">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_CHANGE_INSTANCE_DUEDATE">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_CHANGE_TASK_PRIORITY">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_MOVE_TOKEN">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_INJECT_TOKEN">
      <role>tw_admins</role>
    </action>
    <action type="ACTION_VIEW_PROCESS_DIAGRAM">
    </action>
    <action type="ACTION_VIEW_PROCESS_AUDIT">
      <role>tw_admins</role>
    </action>
  </default-action-policy>
</portal>

```

The <role> entry names a security group (see: Security Groups) which is a global group with defined members. If no <role> is defined within an action tag then all users are allowed to perform that action.

The following is an example of an entry that can be made in the 100Custom.xml file to change the authorities:

```
<portal>
```

```

<default-action-policy>
  <action type="ACTION_VIEW_PROCESS_DIAGRAM">
    <role>Test Group 1</role>
  </action>
</default-action-policy>
</portal>

```

It is strongly recommended **not** to edit the content of the 99Local.xml file. Instead, edit the 100Custom.xml file and add/replace all changes there. When Process Server starts, it will merge the 99Local.xml with the 100Custom.xml producing a combined result. Editing only 100Custom.xml keeps all the changes in one place and aids in maintainability. Changes made to these configuration files don't take effect until after the next restart of the Process Server/Process Center.

Securing access to publish Integration Designer projects

In order to publish modules from Integration Designer, the connected user must have the WAS Administrative permissions of "deployer" or "configurator". These can be set from the WAS admin console under Security > Global Security > Administrative Security:

The screenshot shows the 'Global security' administrative interface. At the top, there are two tabs: 'Security Configuration Wizard' and 'Security Configuration Report'. Below these tabs, there are two main sections: 'Administrative security' and 'Application security'. In the 'Administrative security' section, there is a checkbox labeled 'Enable administrative security' which is checked. To its right, there is a list of three items: 'Administrative user roles', 'Administrative group roles', and 'Administrative authentication'. The 'Administrative authentication' item has a sub-menu with several options: 'LT' (Local Trust), 'Kerberos', 'SW', 'Authenticode', and 'Web'. A large red arrow points to the 'LT' option in the 'Administrative authentication' menu. In the 'Application security' section, there is also a checked checkbox labeled 'Enable application security'.

Global security

[Global security](#) > [Administrative user roles](#) > User

Use this page to add, update or to remove administrative roles to users. administer application servers through the administrative console or thro

* Role(s)

Administrator
Auditor
Configurator
Deployer

Search and Select Users

Decide how many results to display, enter a search string (use * for wildc add them to the Mapped to role list. Users which have already been map

Search string Search

Maximum results to display

Available

coRequestor3
resAdmin
resDeployer
resMonitor
tw_portal_admin
tw_runtime_server
tw_user
tw_webservice
user2
user3

Mapped to role

user1

Select All Deselect All

OK Reset Cancel

The screenshot shows a web-based administrative interface for managing user roles. At the top, there's a header bar with the title 'Global security'. Below it, a breadcrumb navigation shows 'Global security > Administrative user roles > User'. A descriptive text block explains the purpose of the page: 'Use this page to add, update or to remove administrative roles to users. administer application servers through the administrative console or thro'. There's a section titled 'Role(s)' containing a list of roles: 'Administrator', 'Auditor', 'Configurator', and 'Deployer', with 'Deployer' currently selected. Below this is a section titled 'Search and Select Users' with a search string input field containing '*' and a 'Search' button. It also includes a 'Maximum results to display' input field set to '20'. Two lists are shown: 'Available' on the left and 'Mapped to role' on the right. The 'Available' list contains several user and system identifiers like 'coRequestor3', 'resAdmin', etc. The 'Mapped to role' list contains one item, 'user1', which is highlighted in blue. At the bottom are 'Select All' and 'Deselect All' buttons for both lists, and standard 'OK', 'Reset', and 'Cancel' buttons.

Lightweight Directory Access Protocol - LDAP

LDAP is a commonly used registry of information and is typically used to hold organizational information including the definitions of users, passwords and group membership. WAS can leverage an LDAP server for management of users and passwords. IBM's implementation of LDAP is called IBM Tivoli Directory Server.

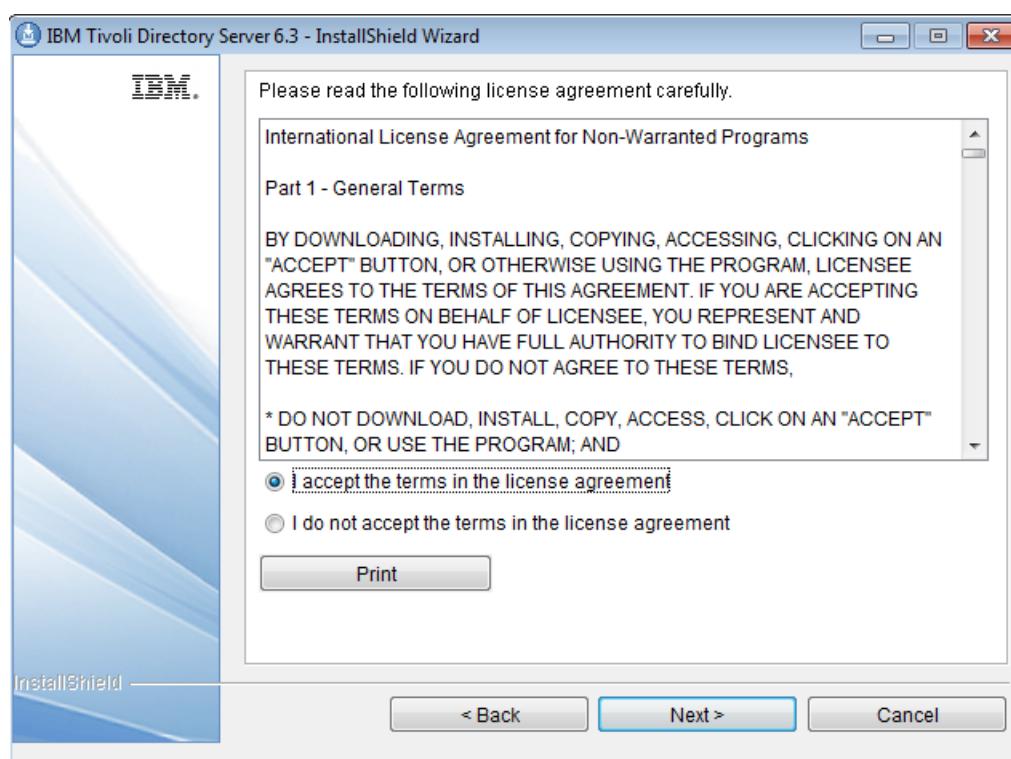
See also:

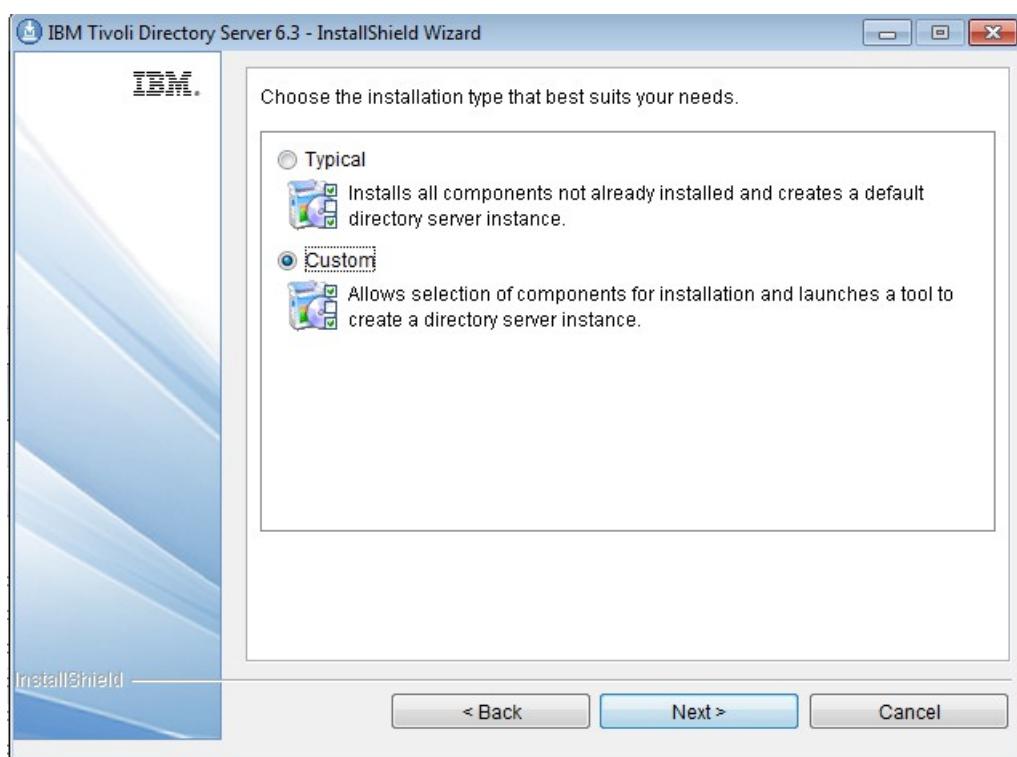
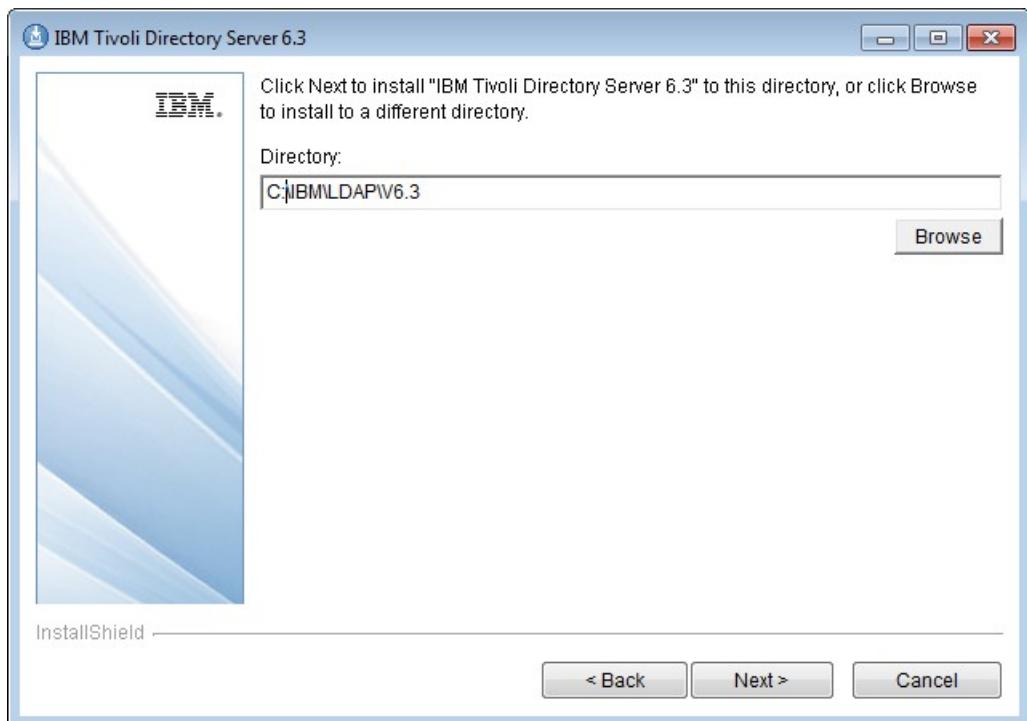
- [Redbook - IBM Business Process Manager V7.5 Production Topologies](#) – Chapter 5.6
- DeveloperWorks - [Associating WebSphere Lombardi V7.2 attributes with users defined in OpenLDAP and routing tasks](#) - 2012-01-18
- DeveloperWorks - [Integrate an LDAP user registry into a WebSphere Lombardi Edition business process](#) - 2011-10-31
- DeveloperWorks - [Secure integration of an LDAP user registry with WebSphere Lombardi Edition](#) - 2011-06-29

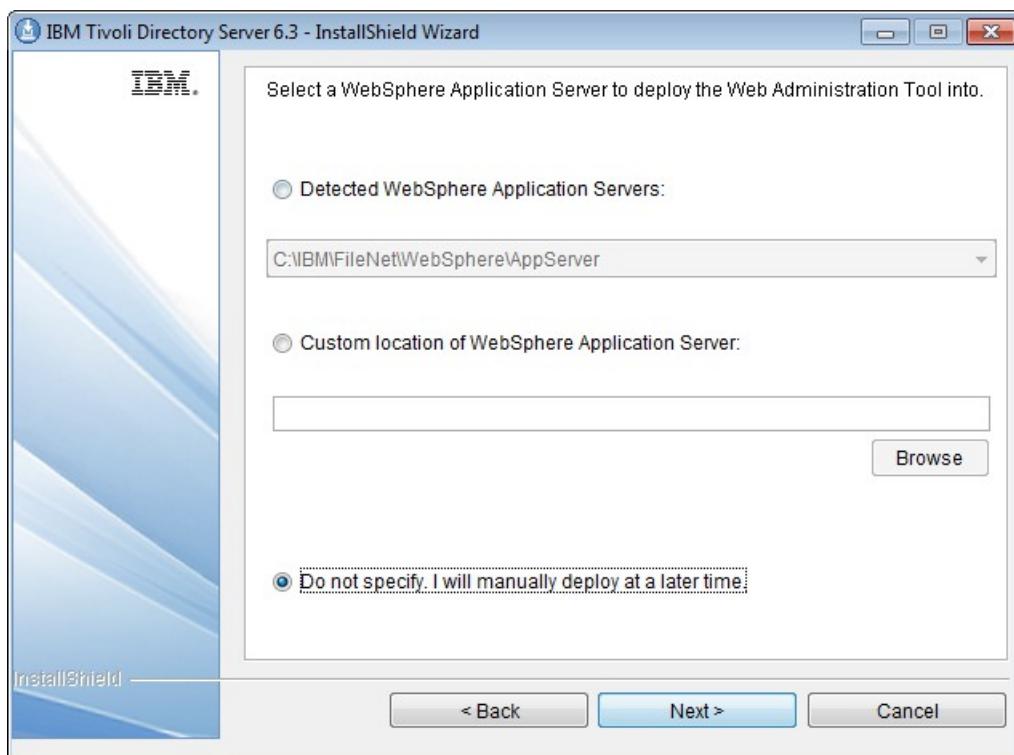
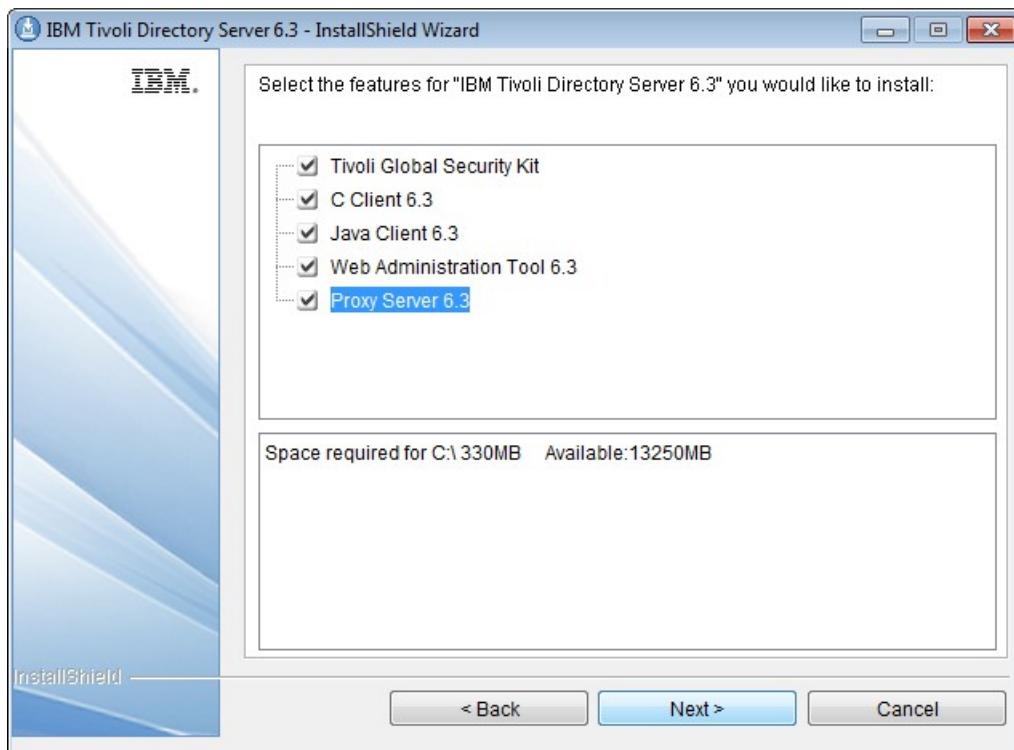
Tivoli Directoy Server

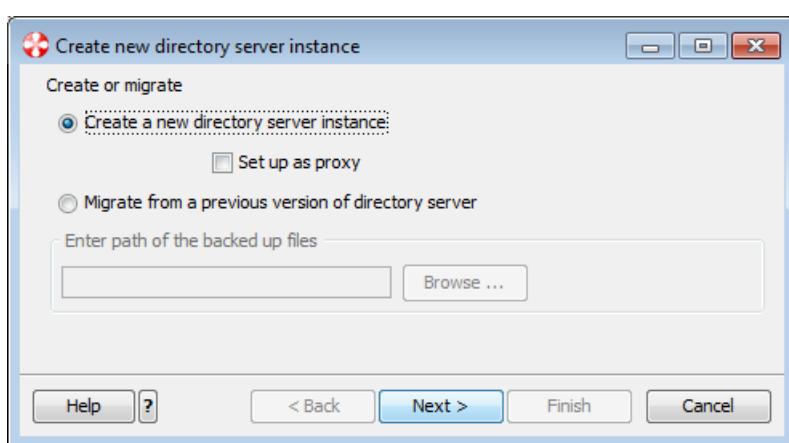
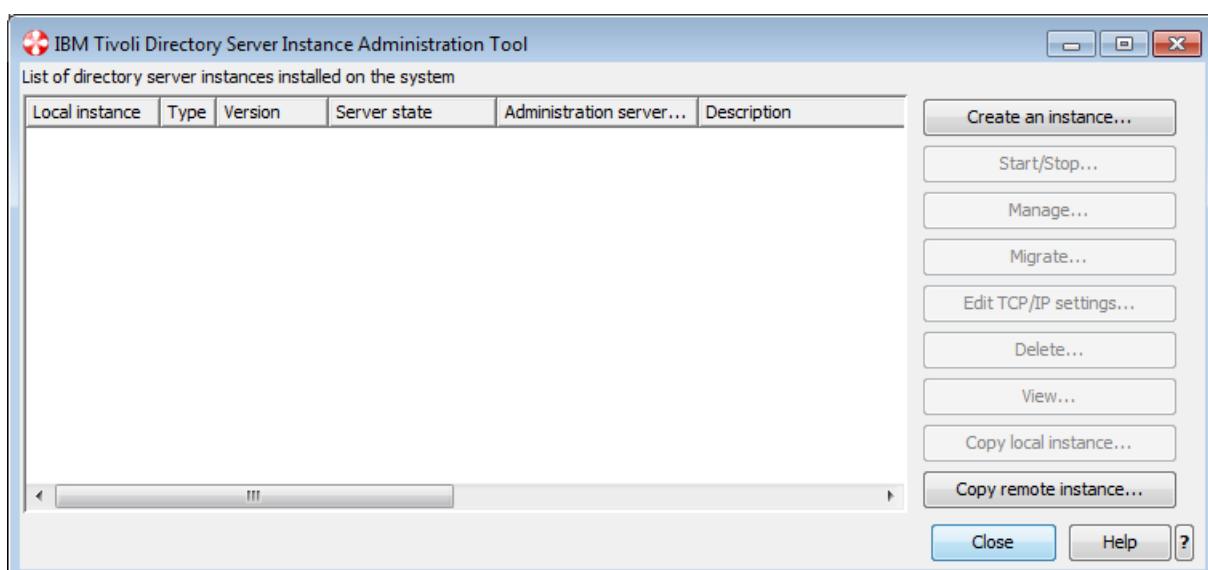
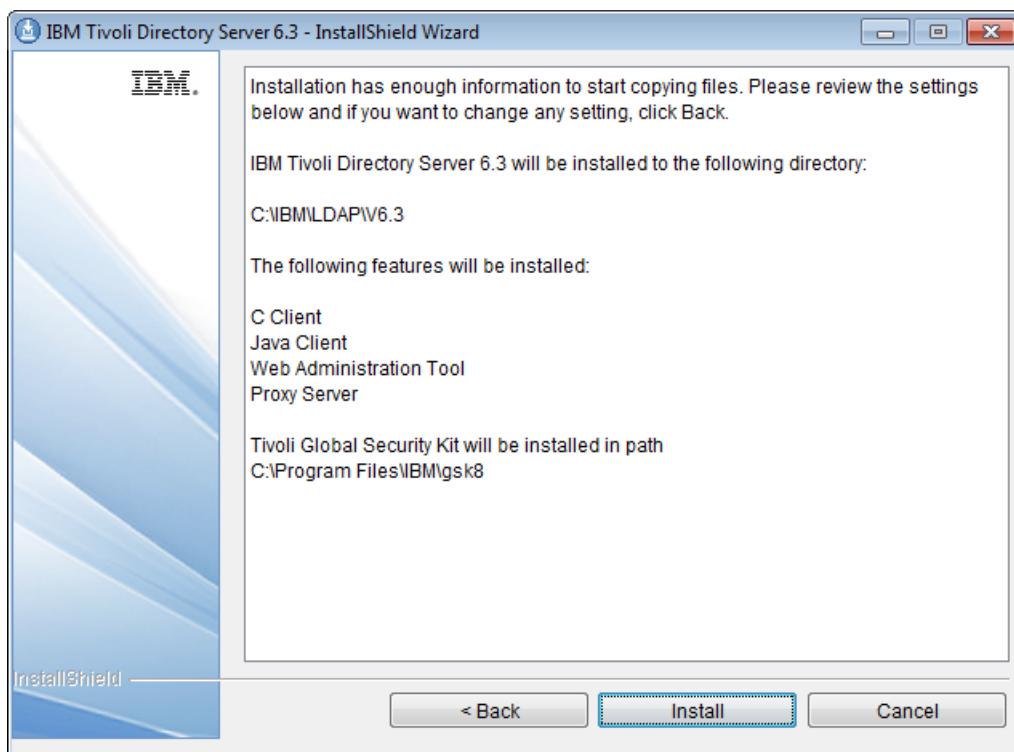
As of July 2012, the latest version of Tivoli Directoy Server is 6.3.

The installer for TDS looks as follows:









See also:

- [Home page for Tivoli Directory Server](#)
- [InfoCenter for TDS 6.3](#)

Apache Directory Server

The Apache Directory Server is an implementation of LDAP completely written in Java and made available as Open Source. The home page for this free product is:

<http://directory.apache.org/>

It is a relatively small download (11 Mbytes) and is self contained for execution. Once installed, here are some basic properties that are useful:

Userid	uid=admin,ou=system
Password	secret
Default port	10389

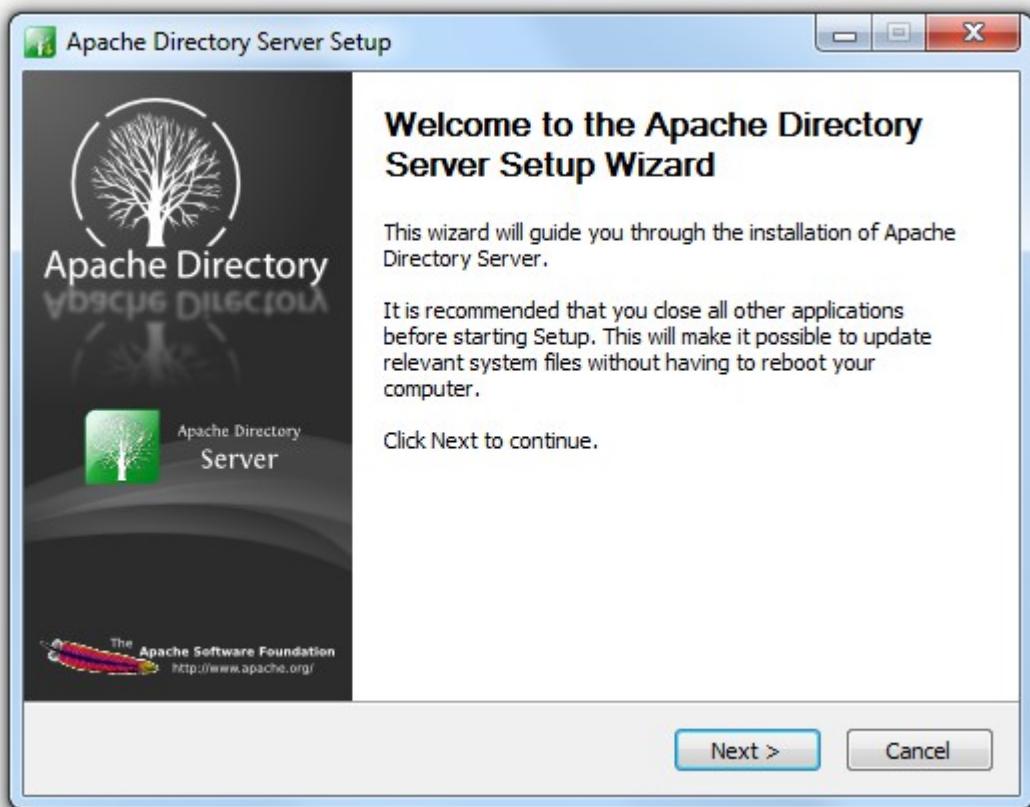
Installation of Apache Directory Server

What follows is a walk through of the installation of the Apache Directory Server.

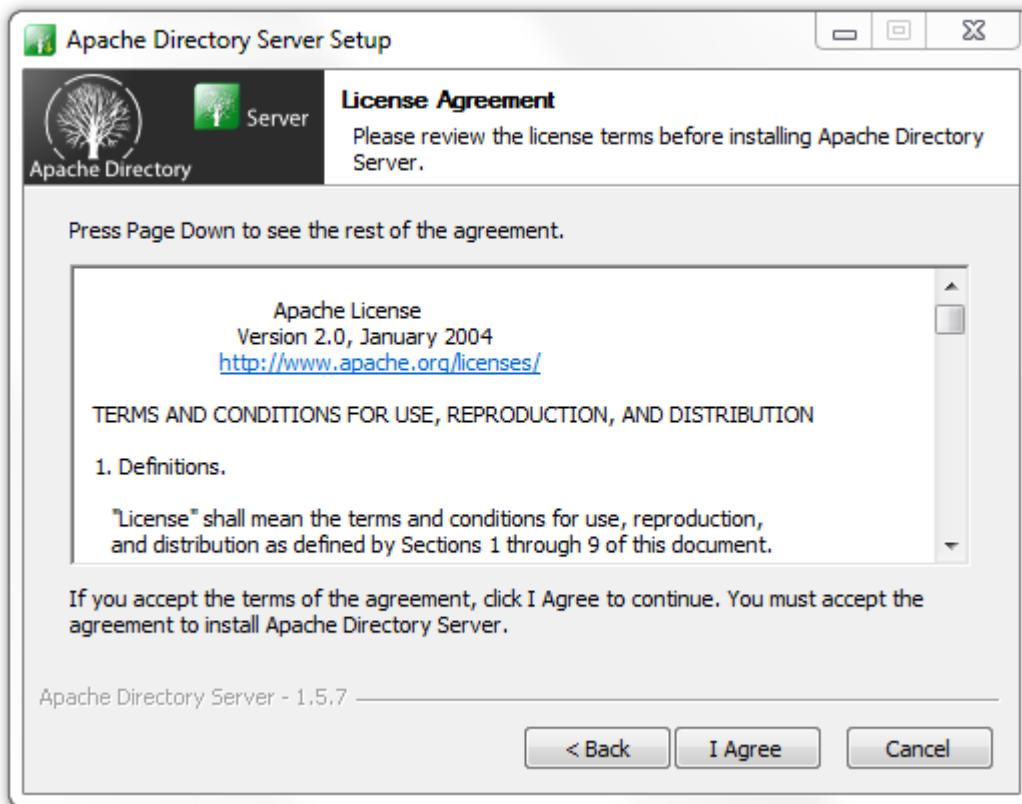
When initially downloaded, the server comes as a single setup file.



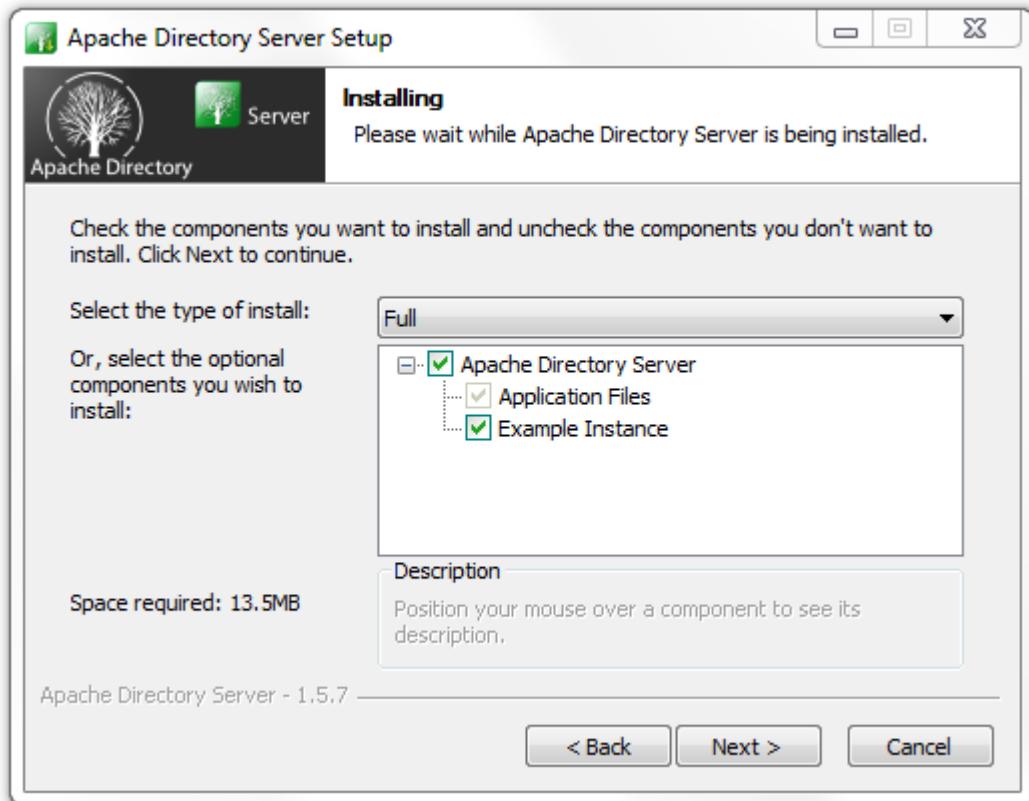
Once launched, the common boiler plate is shown.



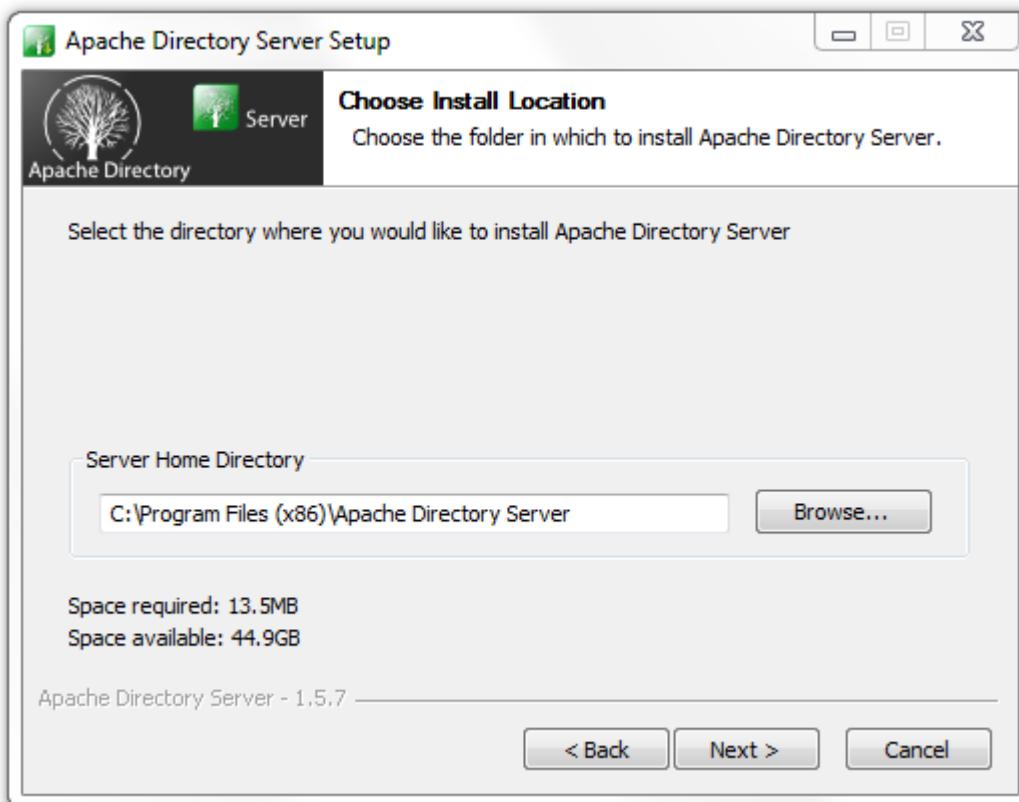
Next comes the license agreement. Read and accept if you agree.



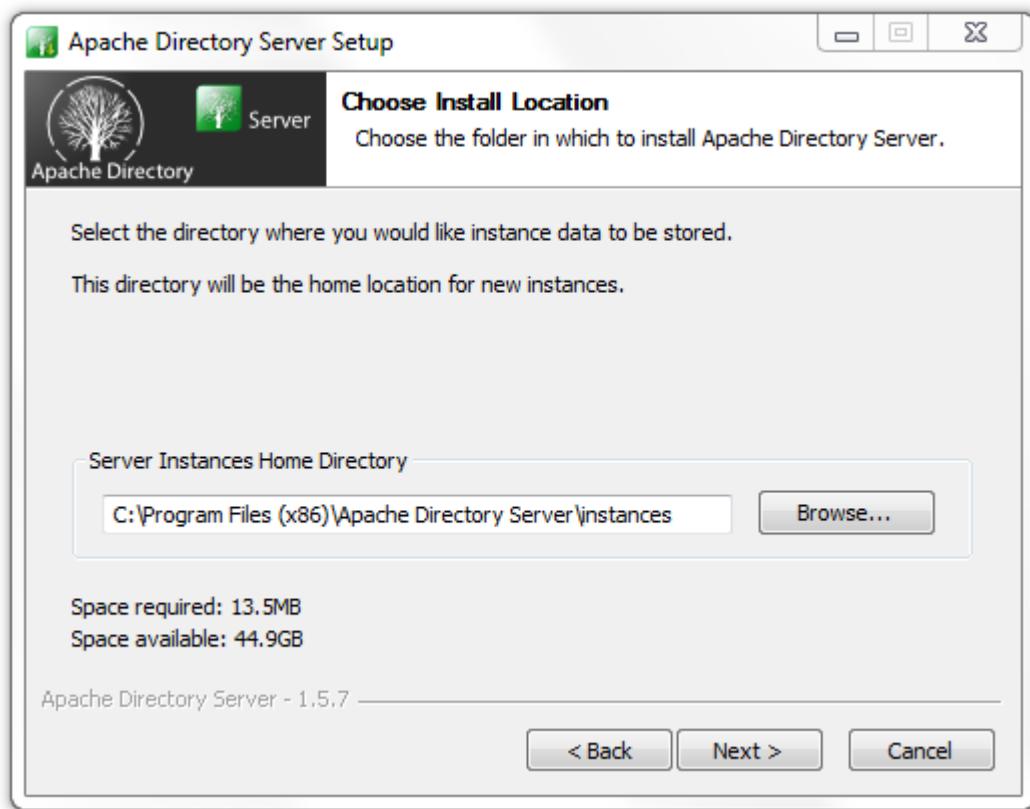
There are installation choices, a Full setup is what is shown in this walkthrough.



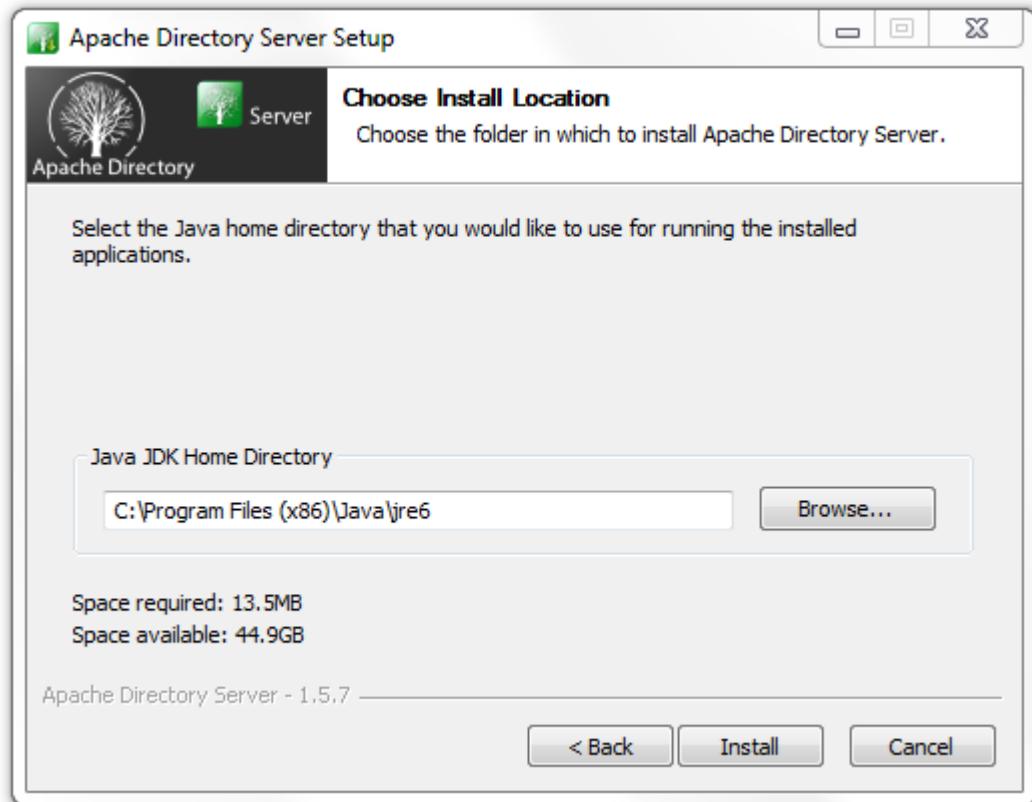
The file system location where the Apache Directory Server is to be installed is configurable.



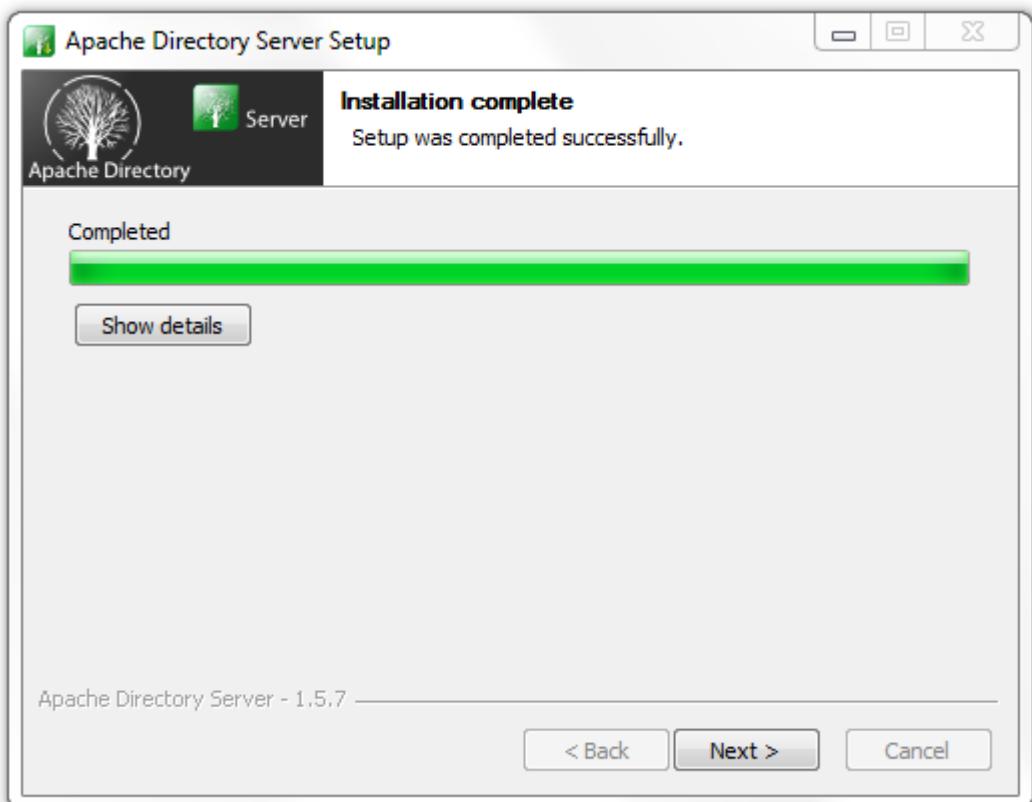
An LDAP directory contains data describing the structure of an organization. The location on the file system of where this data is stored can be configured.



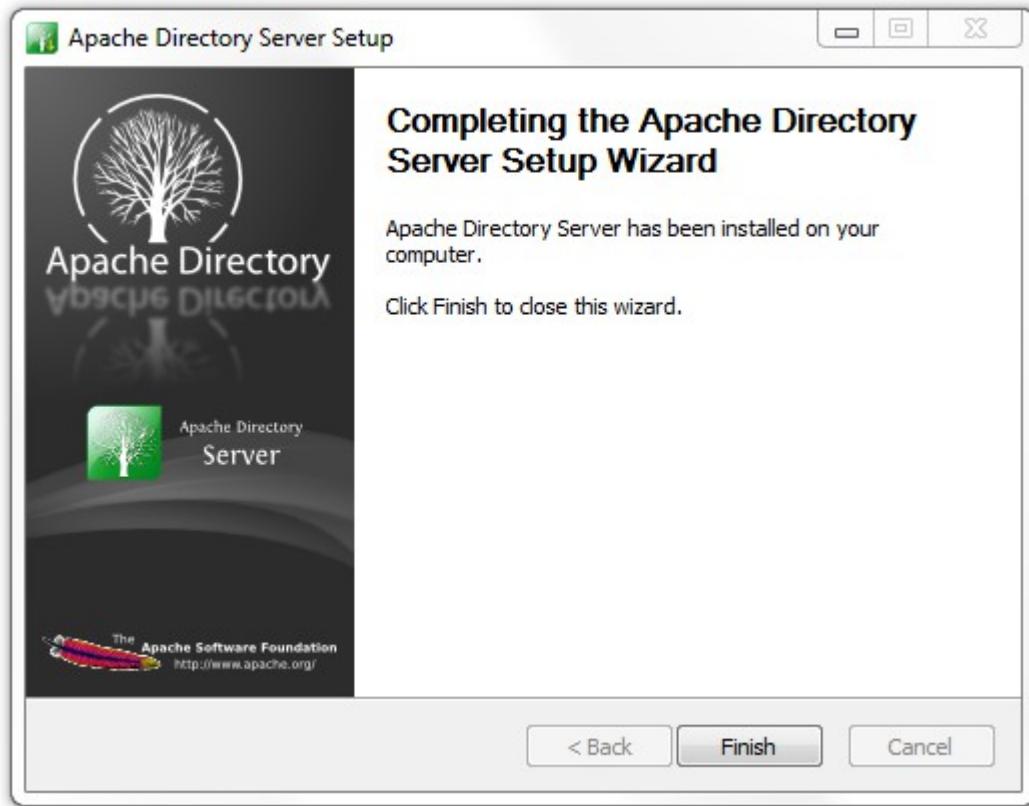
The Apache Directory Server is written in Java and needs a Java runtime. The location of the Java runtime to be used must be supplied.



Finally, the installation progresses.



At the conclusion, a completion page is shown.

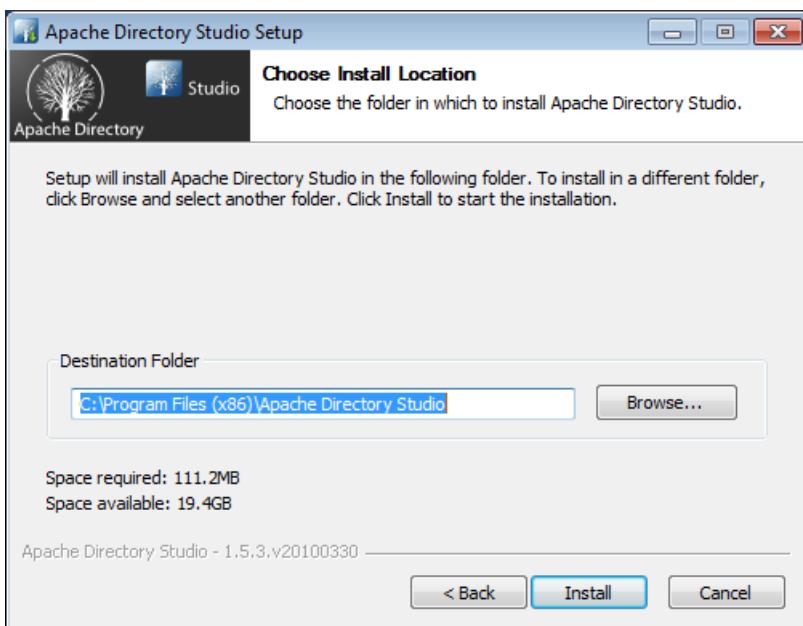
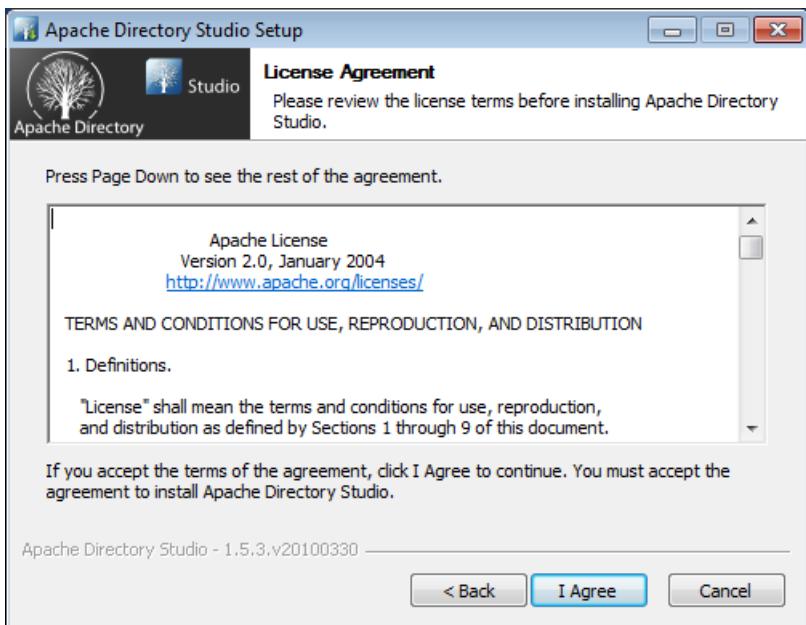


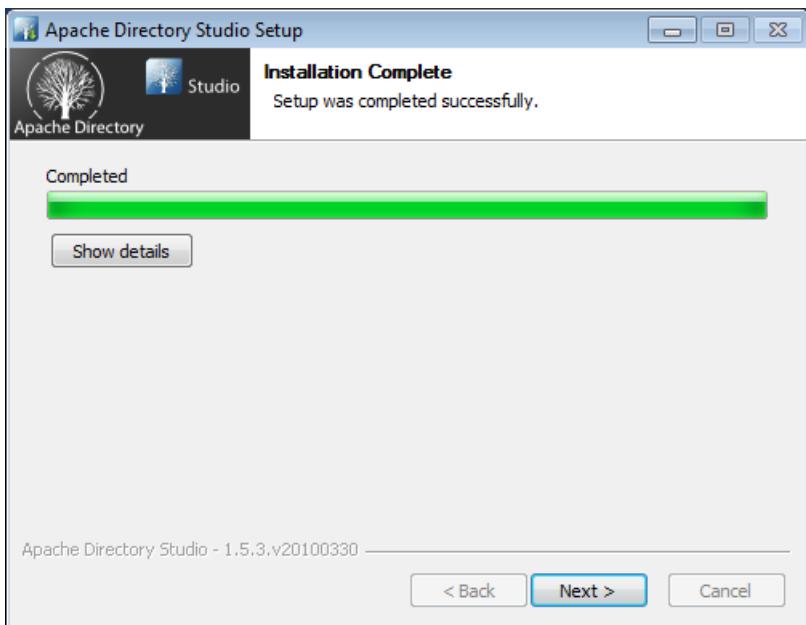
Apache Directory Server Studio

Another free Apache product is the Apache Directory Server Studio. This is a fantastic configuration tool for the Apache Directory Server and other LDAP servers.

Installation of Apache Directory Studio







Installation of Apache Directory Server Studio Eclipse Plugins

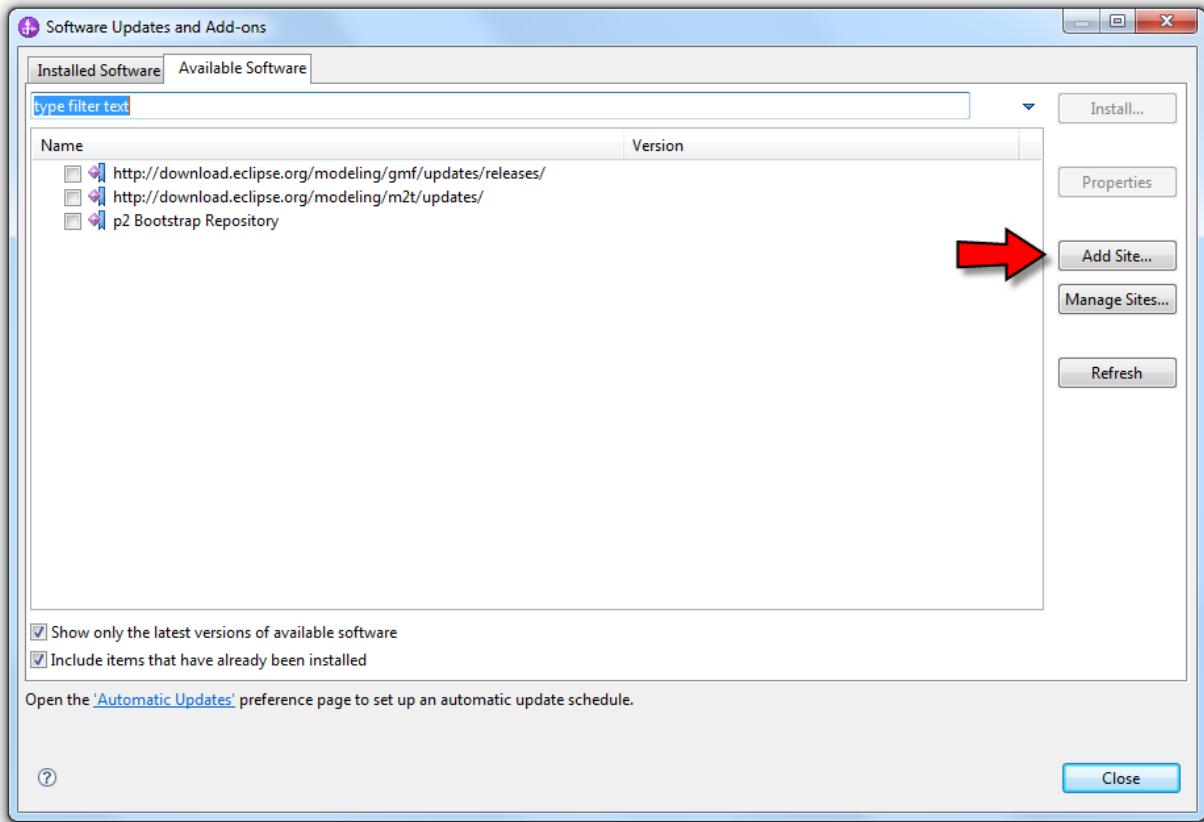
Apache Directory Server Studio can be installed as a set of Eclipse plugins. This set of Eclipse plugins that can be installed into ID or RAD. This provides a set of LDAP maintenance tools as well as an in-built LDAP server that is excellent for development and testing.

It can be installed directly into ID through dynamic retrieval.

Web site for Eclipse plugins:

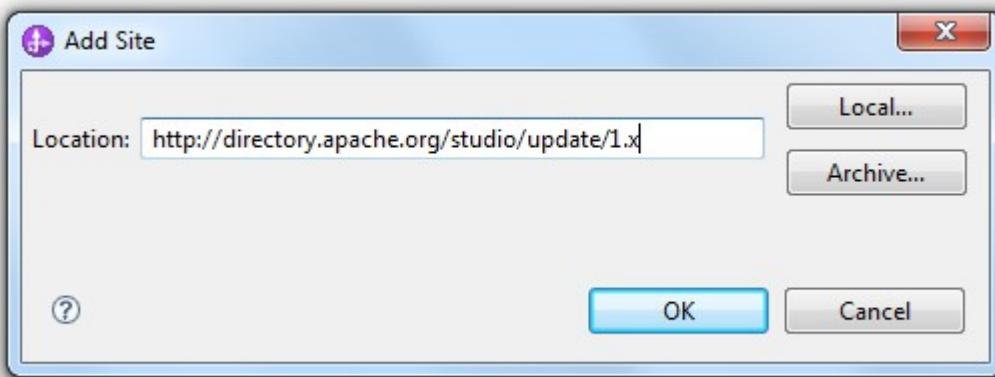
<http://directory.apache.org/studio/>

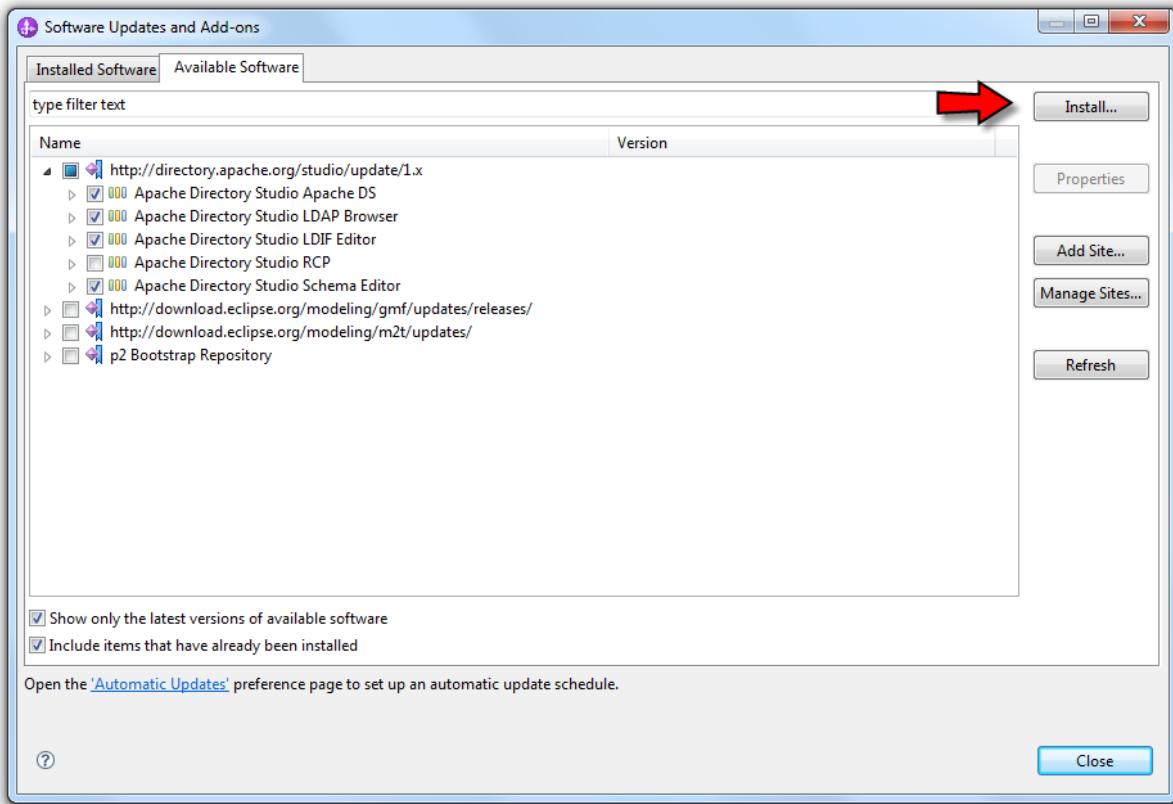
Go to Help >



For the download URL for the eclipse plugins, enter:

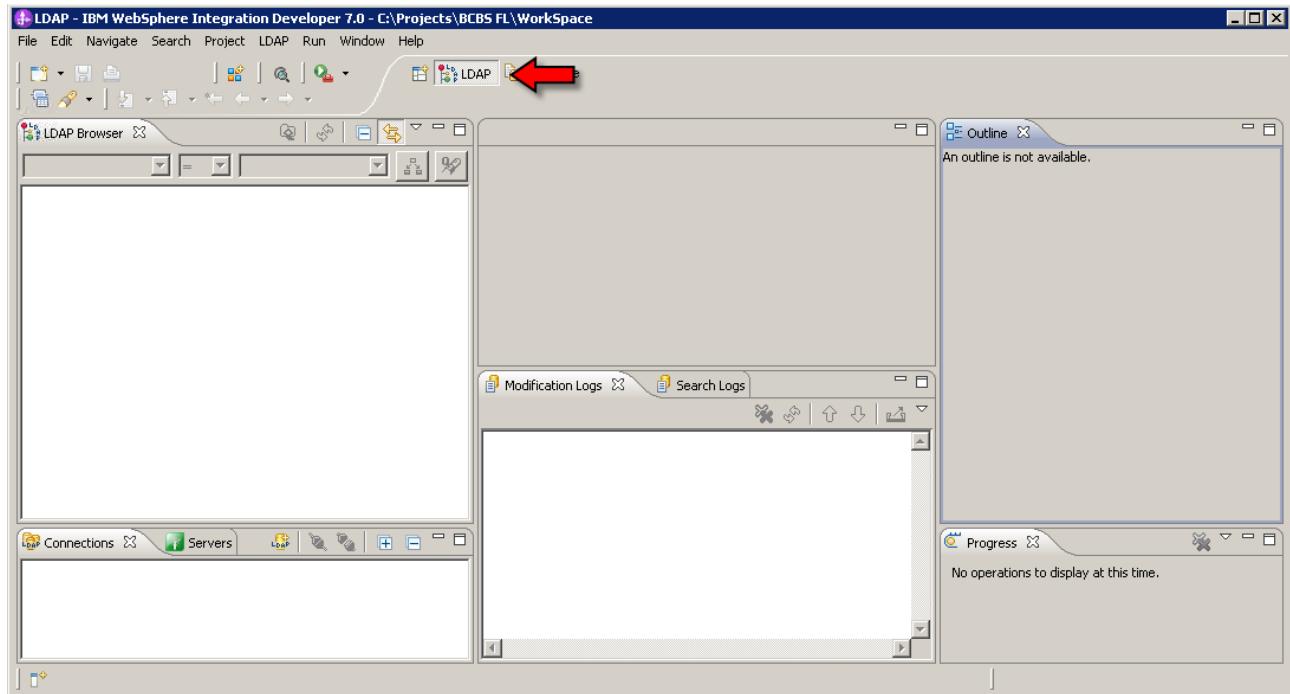
<http://directory.apache.org/studio/update/1.x>





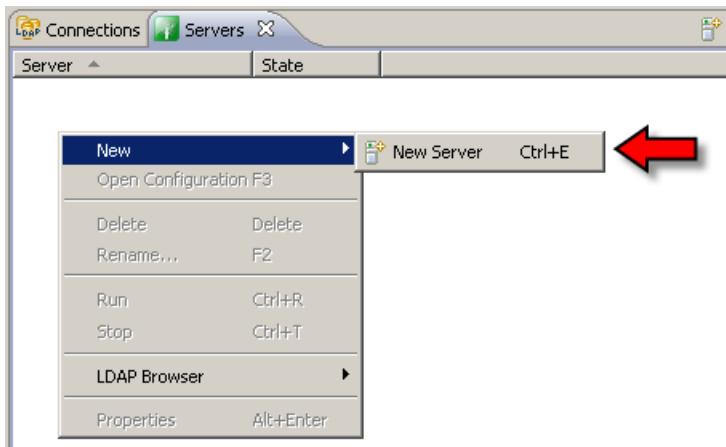
The packages will now be dynamically downloaded from the web and installed into WID.

Once installed, a new perspective called LDAP is available:

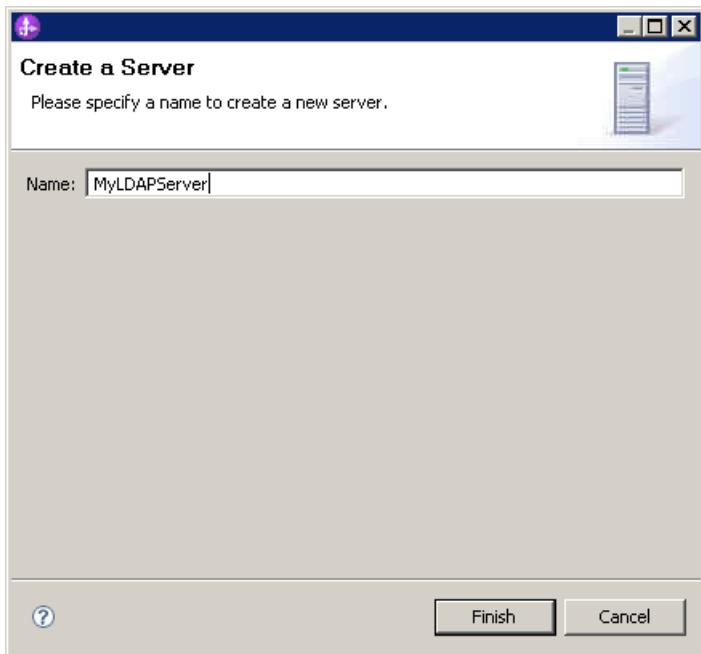


In addition to being able to work with external LDAP Servers, the Studio provides its own LDAP server instance. This provides the ability to quickly create an LDAP instance for testing but realize that once the studio has been shut down, the LDAP server shuts down too. It is probably much safer to run an LDAP server outside of the Studio.

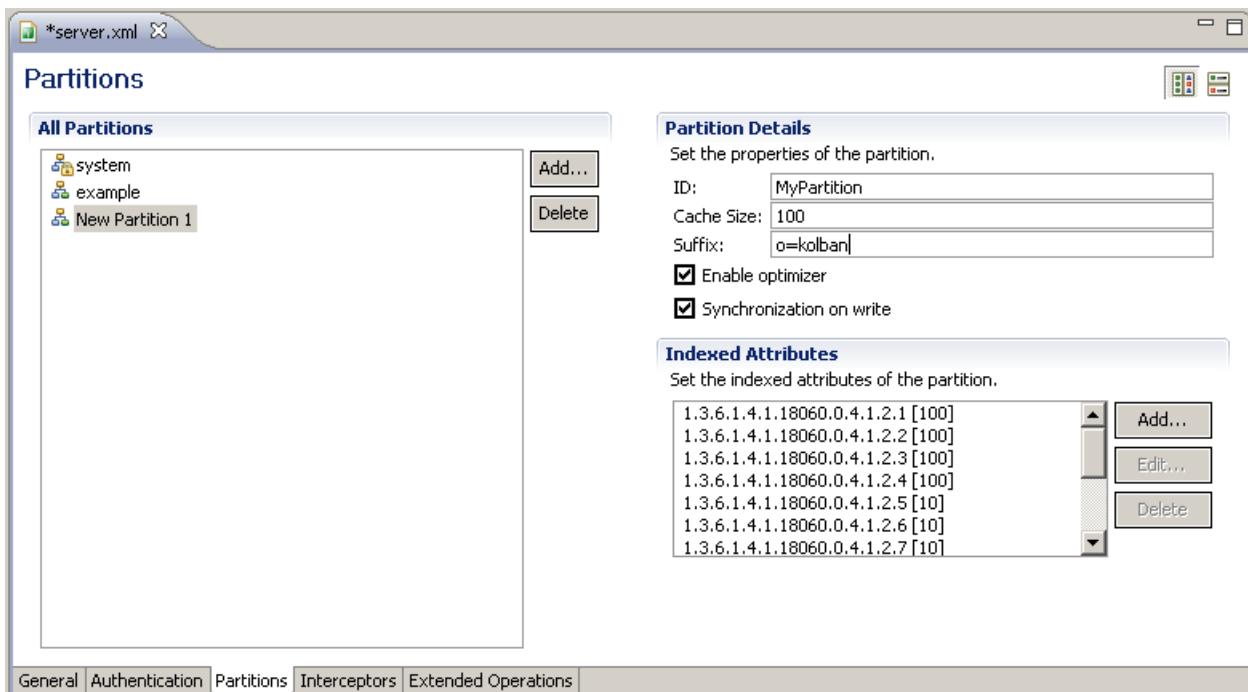
To create a Studio hosted LDAP server, create a New Server definition.



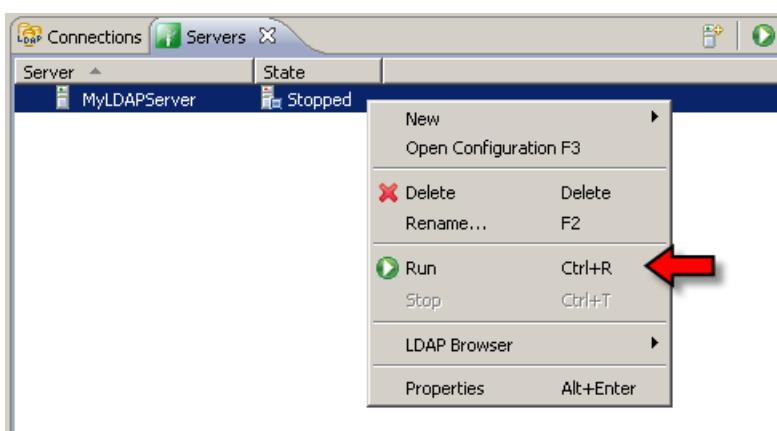
Give the new server a name:



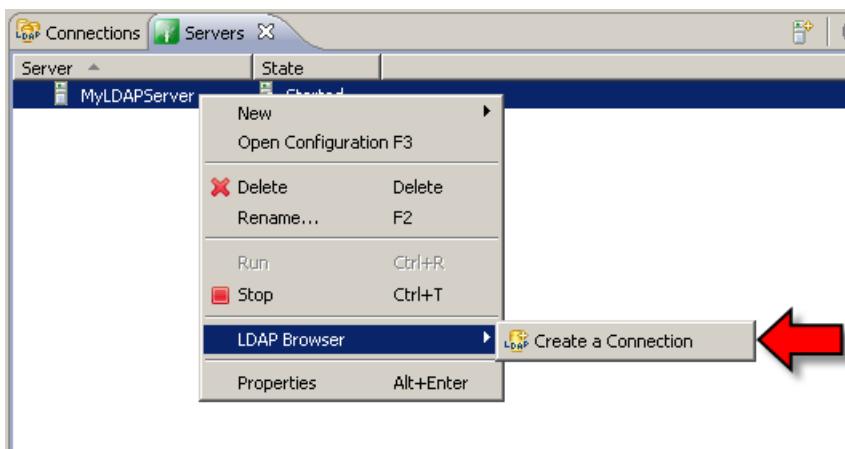
Provide any properties that are specific to the new server.



Start (run) the Server from the Servers panel.



Create a connection to the new server to access and populate it with content.



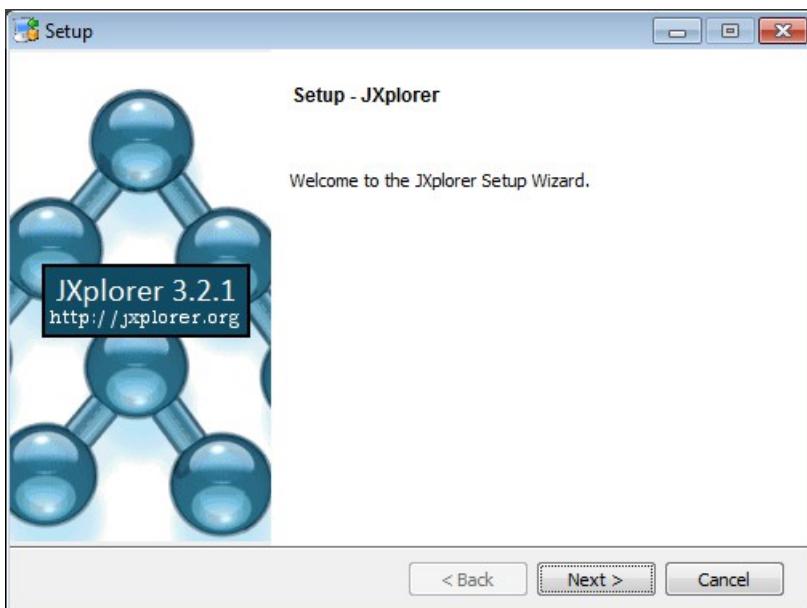
Installation of JXplorer

JXplorer is an Open Source GUI LDAP browser tool. If you don't install the Apache LDAP Directory Studio, this can provide a useful alternative. It can be found at:

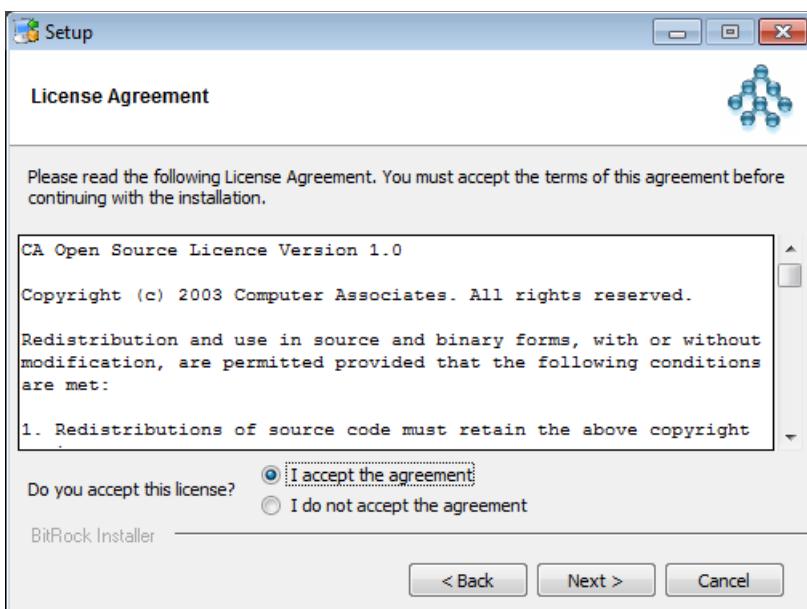
<http://www.jxplorer.org/>

A walk through of the installer is shown next.

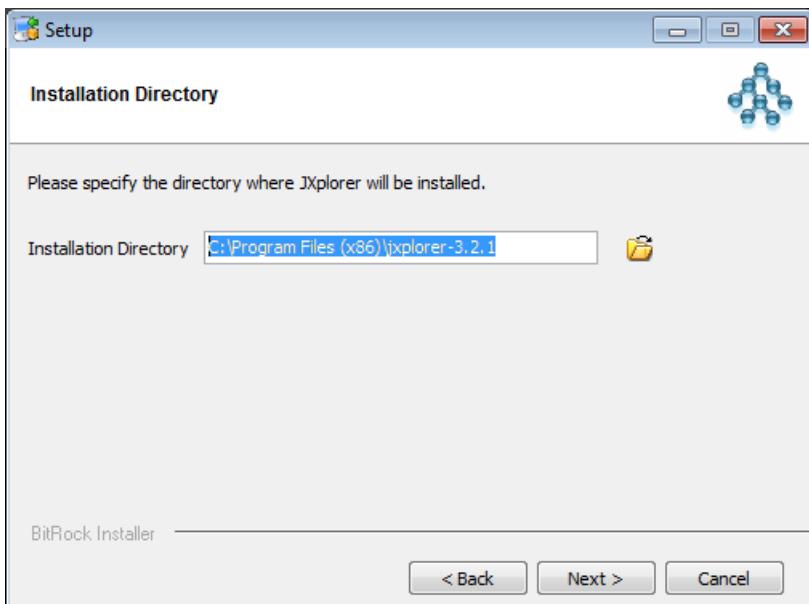
First comes the basic welcome/setup screen.



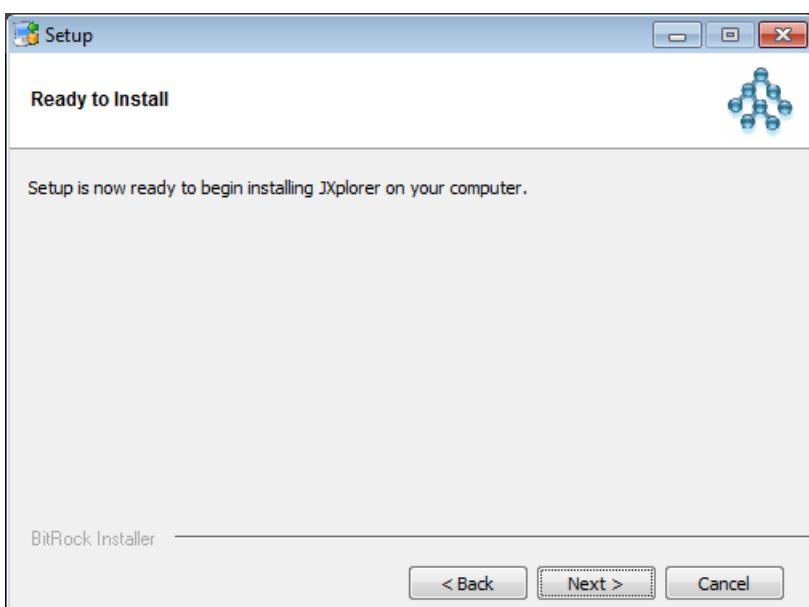
The license agreement is shown next and should be read. If agreed, accept the license and continue.



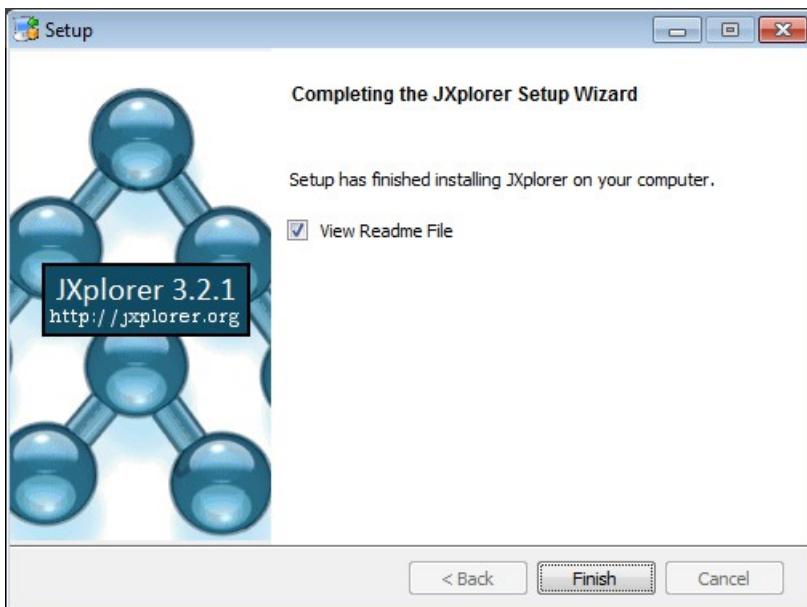
The directory in which the tool will be installed is shown next.



Following this, installation is ready to progress.



After installation is complete, a summary is shown.



The tool can now be found in the Start Menu under the JXplorer folder.

Getting locked out of WAS

While configuring WAS for LDAP, if an error is made during this time, you might find yourself getting "locked out". This means that you can't login to WAS to perform repairs. The resolution to this is to stop the WAS server and temporarily switch off security. Unfortunately, if you have lost admin access to WAS, you won't be able to stop it cleanly. You will have to locate the process id at the operating system level of the Java JVM running WAS and kill it. [Process Explorer](#) is a good tool for finding the correct process ID and killing it.

Once WAS has been stopped, we can run a wsadmin command to disable security.

```
wsadmin -conntype NONE  
wsadmin> securityoff  
wsadmin> exit
```

When the WAS server is restarted, security will be off. Access to the WAS admin console can then be made through a browser at:

<https://localhost:9043.ibm/console/unsecureLogon.jsp>

Configuring WAS for LDAP

In order to include an LDAP server in the mix of security repositories, work must be performed through the WAS Admin Console. Let us assume we have an Apache Directory server with base dn of ou=system.

We login to the WAS Admin console using admin/admin. In the Security settings, we select global security:

Integrated Solutions Console Welcome tw_user

View: All tasks

- Welcome
- ⊕ Guided Activities
- ⊕ Servers
- ⊕ Applications
- ⊕ Services
- ⊕ Resources
- ⊕ Security 
- Global security
- Security domains
- Administrative Authorization Groups
- SSL certificate and key management
- Security auditing

Next we click on the Configure... button associated with Federated repositories:

Global security

Global security

Use this panel to configure administration and the default application security policy. This administrative functions and is used as a default security policy for user applications. It policies for user applications.

[Security Configuration Wizard](#) [Security Configuration Report](#)

Administrative security

Enable administrative security [Administrative user roles](#) [Administrative group roles](#) [Administrative authentication](#)

Application security

Enable application security

Java 2 security

Use Java 2 security to restrict application access to local resources Warn if applications are granted custom permissions Restrict access to resource authentication data

User account repository

Current realm definition: [Federated repositories](#)

Available realm definitions: [Federated repositories](#) [Configure...](#)  current

[Apply](#) [Reset](#)

Click the Manage repositories link:

Global security

Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be managed in a single, virtual realm. The realm can consist of identity-based repository that is built into the system, in one or more external repositories, or in both the built-in repository and one or more external repositories.

General Properties

* **Realm name**
defaultWIMFileBasedRealm

* **Primary administrative user name**
tw_user

Server user identity

- Automatically generated server identity
- Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node
tw_user

Password

Ignore case for authorization

Repositories in the realm:

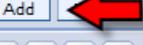
Add Base entry to Realm...	Use built-in repository	Remove	
Select	Base Entry	Repository Identifier	Repository Type
You can administer the following resources:			
<input type="checkbox"/>	o=twinternal	urbtwinternal	Custom

Additional Properties

- [Property extension repository](#)
- [Entry mapping repository](#)
- [Supported entity types](#)

Related Items

- [Manage repositories](#)
- [Trusted authentication realms - inbound](#)



Click the Add button

Global security

Global security > Federated repositories > Manage repositories

Repositories that are configured in the system are listed in the following table. You can add or delete external repositories.

Preferences

Add		
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>		
Select	Repository Identifier ▲	Repository Type ▲
You can administer the following resources:		
<input type="checkbox"/>	InternalFileRepository	File
<input type="checkbox"/>	urbtwinternal	Custom:null
Total 2		

Enter values for the Repository identifier, host name and port.

Global security

[Global security](#) > [Federated repositories](#) > [Manage repositories](#) > New

Specifies the configuration for secure access to a Lightweight Directory Access Protocol (LDAP) repository with a General Properties

* Repository identifier: Apache1

LDAP server

* Directory type: Custom

* Primary host name: localhost

Port: 10389

Failover server used when primary is not available:

Support referrals to other LDAP servers: ignore

The additional properties will not be available until the general properties for this item are applied or saved.

Additional Properties

- Performance
- LDAP entity types
- Group attribute definition

Security

Bind distinguished name:

Bind password:

Login properties: uid

LDAP attribute filter:

Certificate mapping: EXACT_DN

Certificate filter:

Require SSL

Centrally managed

- Manage entries
- Use specific NodeDefinitions

Buttons

Apply OK Reset Cancel

Back in the Global security page, click the Configure button again.

Click the Add Base entry to Realm... button

Global security

Global security > Federated repositories

By federating repositories, identities stored in multiple repositories can be managed in a file-based repository that is built into the system, in one or more external repositories, or both.

General Properties

* **Realm name**
defaultWIMFileBasedRealm

* **Primary administrative user name**
tw_user

Server user identity

(Automatically generated server identity
 Server identity that is stored in the repository

Server user ID or administrative user on a Version 6.0.x node
tw_user

Password

Ignore case for authorization

Repositories in the realm:

Add Base entry to Realm...	In repository	Remove
Select	Base Entry	Repository Identifier
You can administer the following resources:		
<input type="checkbox"/>	o=twinternal	urbtwinternal



Ensure the correct repository is selected and now supply the root DN for the searches for this repository.

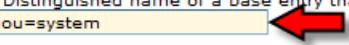
Global security

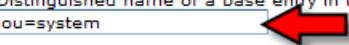
Global security > Federated repositories > Repository reference

Specifies a set of identity entries in a repository that are referenced by a base entry into this realm. If the base entry is located in the same realm, it might be necessary to define an additional distinguished name that is used to identify the base entry.

General Properties

* **Repository**
Apache1

* **Distinguished name of a base entry that uniquely identifies this set of entries in the realm**
ou=system 

Distinguished name of a base entry in this repository
ou=system 

Restart all the servers.

Configuring LDAP for IBPM

IBPM expects the following resources to be available:

- user – tw_portal_admin
- user – tw_user
- group – teamworks_admin

- group – teamworks_authors
- group – tw_admins
- group – TWSecurityProviderUsers

Debugging LDAP

If the authentication system being used by WebSphere is LDAP, then it can occasionally be useful to examine the requests and responses flowing to and from the LDAP server. TDS has a an audit log where one can see requests being sent to it but there is no obvious way to see the responses being returned. In addition, other LDAP providers may be used that you are not familiar with. A solution to tracing the LDAP flows is to use the package called [SLAMD](#). Although this package is all about performance and load testing, as part of this package, in the tools folder, a tool called `ldap-decoder` is provided. This tool acts as a proxy/interceptor between an LDAP client and an LDAP server and displays LDAP queries and responses.

Here is a quick cheat sheet on using `ldap-decoder`. First, understand that it is a DOS command window tool and hence must be run from the DOS prompt. It sits between a real LDAP provider and a client (such as IBPM) that is making LDAP requests. When IBPM makes a query against LDAP, it is really making the request to `ldap-provider`. `Ldap-decoder` then forwards the request to the real LDAP provider which in turn sends a response back to the `ldap-decoder` tool. This in turn sends the response back to IBPM. This means that `ldap-decoder` sees both the requests to LDAP and the responses returned from LDAP. `Ldap-decoder` knows how to parse the LDAP data stream (which is binary) and displays the requests and corresponding responses in the DOS console window.

The `ldap-decoder` tool can be found in the tools sub folder of the `slamd` installation directory. Before starting the tool, edit the file called `set-java-home.bat` and change the `JAVA_HOME` environment variable to point to a Java runtime.

To run `ldap-decoder`, execute the following from the `<slamd>/tools` folder:

```
ldap-decoder -h LDAPProviderHostname -p LDAPProviderPort -L LocalListenPort
```

For example:

```
ldap-decoder -h localhost -p 10389 -L 10390
```

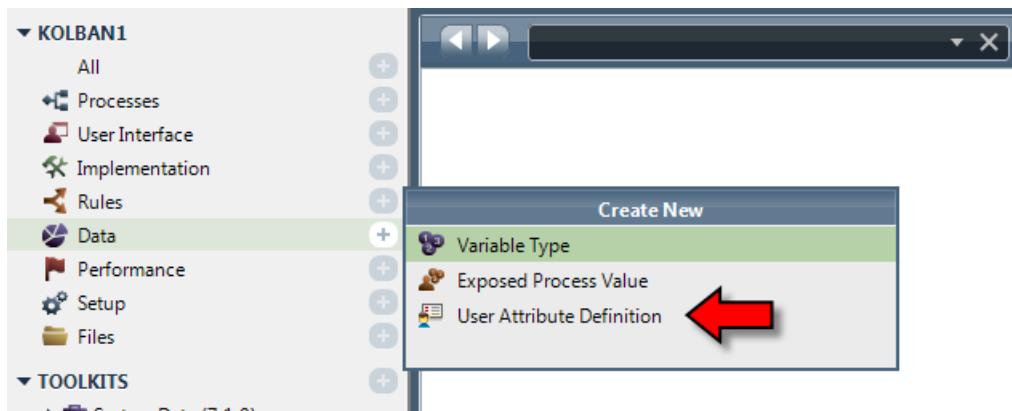
`Ldap-decoder` will then start listening on the `LocalListenPort` and when it receives a request, it will be forwarded to the `LDAPProviderHostname` at port `LDAPProviderPort`.

User Attribute Definitions

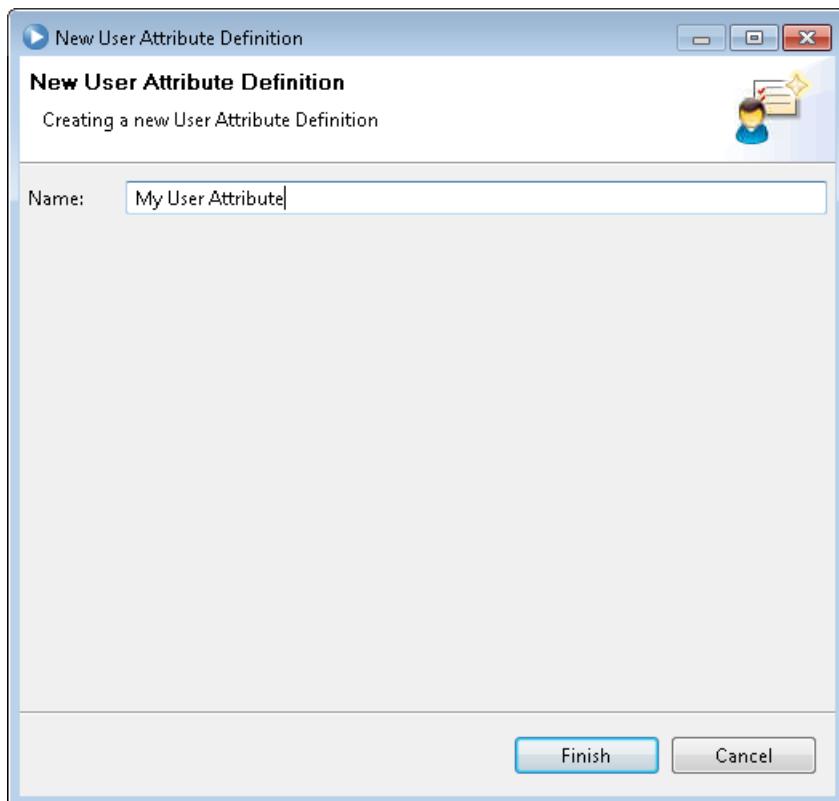
A User Attribute definition is an attribute (property) that can be configured on user definitions in IBM BPM. For example, if we needed to route a process task to users who speak Spanish, we need to know the languages spoken by our users. This example is not an attribute present by default on our user definitions. Through the creation of a User Attribute, we can augment the existing attributes with new ones.

A User Attribute definition made within a Process Application has the unusual effect of changing the environment as a whole. Normally we think of Process Application definitions as only being applicable to the Process Application in which they reside. User Attributes are different. Defining a new User Attribute causes that attribute to be created against all users for every process.

A User Attribute definition can be created from within the Data category of the Library.



When created, a dialog appears asking for a name for this attribute:



The definition parameters for a User Attribute looks as follows:

The fields contained in this definition are:

- Name – The name of the User Attribute.
- Modified – The person, date and time who last changed the definition.
- Documentation – Notes and other documentation about this definition
- Variable Type – The data type associated with this attribute
- (Obtain current value from...) Source – Only "IBM BPM" is available here
- (Obtain possible values from...) Source – This has two possible settings. The default is Any Allowed. This means that any text value can be set as the value of the attribute. The other setting is List meaning that the values are constrained to be taken from a list of possible values. If this selection is made, a list entry field is shown into which the possible values may be entered:

The value of the attribute for a particular user can be set within the Process Admin Console. See Bulk User Attribute Assignment.

The default attributes for a user are:

- Alert on Assign and Run
- Base Text Direction
- Calendar Type
- Image

- Job Skill
- Locale
- Phone Number
- Portal Dashboard Display Order
- Portal Default Page
- Portal Mention Timestamp
- Portal Notification New Task To Me
- Portal Notification New Task To My Group
- Portal Notification Post Mention
- Primary Role
- Send Connections Notification On New Task
- Show Unfollow Messages
- SkillLevel
- Task Email Address
- Task Notification
- TaskThreshold
- Title
-

Attributes can be accessed from the JavaScript data type called `TWUser`. This has a property/object on it called attributes. The attributes can be accessed from that object by name. If an attribute name contains a space then JavaScript associative array access should be used. For example:

```
var myValue = tw.system.user.attributes['My Attribute'];
```

See also:

- Security
- Bulk User Attribute Assignment

Process Admin Console

The Process Admin Console is a web based application that can be accessed from the following URL:

`http://<hostname>:<port>/ProcessAdmin`

The login screen looks as follows:

After a successful login, the screen looks as follows:

The Process Admin Console is used to administer a Process Server instance.

At a high level, the following tasks can be performed:

IBM BPM Admin	
Manage Caches	
Task Cleanup	
User Management	
User Management	Working with users. See: Process Admin - User Management
Group Management	Working with global groups and group membership. See: Group Management
Bulk User Attribute Assignment	Setting attributes on user definitions. See: Bulk User Attribute Assignment
User Synchronization	
Monitoring	
Instrumentation	
Process Monitor	
Event Manager	
Monitor	
Blackout Periods	
Synchronous Queues	
EM JMS Error Queue	
Admin Tools	
Manage EPVs	
My Password Management	
Change Password	
Saved Search Admin	
Saved Search Admin	

Process Admin - IBM BPM Admin

The IBM BPM Admin section of the Process Admin console provides for cache and task cleanup.

Name	Description	CA	UCA	UCP	Last A.	Status	Actions
GroupCache	Caches group information and list of groups with information.	1	0	0%	8:35 PM	ON	(Show) (Reset)
E@GroupMembers	Stores group members (User IDs and Group IDs)	19	10	0%	11:32 PM	ON	(Show) (Reset)
Runtime TWClass Cache	Caches all business objects for use by the runtime engines	-	-	-	-	ON	(Reset)
PO Cache	Caches all library items	-	-	-	-	ON	(Reset)

The columns on the cache table mean the following:

Column Name	Description
-------------	-------------

CA (Cache Access)	The number of times the cache was refreshed and accessed.
UCA (Un-refreshed Cache Access)	The number of times the cache was accessed but not refreshed.
UCP (Un-refreshed Cache Percentage)	The percentage of un-cached access versus cached access.
Last A. (Last Access)	The time of the last access.
Status	Is caching on or off.
Actions	Links to show and reset cache contents.

Task Cleanup

Current Counts		After Cleanup Counts	
Tasks	2	Tasks	0
Attachments	0	Attachments	0

Process Admin - User Management

The set of users known to IBPM can be updated from the Process Admin Console. A number of pre-defined resources are already present:

tw_admins	group
tw_admin	user
tw_author	user
tw_portal_admin	user
tw_runtime_server	user
tw_webservice	user

To retrieve a list of all existing users, use the wild card character '%' as the profile name to retrieve:

User Management > Maintain User Settings

Retrieve Profile

Internal Lombardi Users

kolban
tw_admin
tw_author
tw_portal_admin
tw_runtime_server
tw_user
tw_webservice

Internal Lombardi User Details

User Name	<input type="text"/>
Full Name	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
Last Login	<input type="text"/>
Disable User	<input type="checkbox"/>

New users can be added by entering the details of the user and pressing the Add button.

User Management > Maintain User Settings

Retrieve Profile

Internal Lombardi Users

kolban
tw_admin
tw_author
tw_portal_admin
tw_runtime_server
tw_user
tw_webservice

Internal Lombardi User Details

User Name	kolban
Full Name	Neil Kolban
Password	*****
Confirm Password	*****
Last Login	<input type="text"/>
Disable User	<input type="checkbox"/>

IBPM sets some requirements on the password values including:

- Must have a minimum length of six characters
- Must not be the same value as the user name
- Must not be the same as an existing password or the previous three passwords

Group Management

In the User Management > Group Management section of the Process Admin Console we can work with the global groups. We can create a new group or work with existing groups.

The screenshot shows the 'User Management > Group Management' page. On the left, a sidebar lists 'envName1' and 'IBM BPM Admin' sections like 'Manage Caches' and 'Task Cleanup'. Under 'User Management', 'Group Management' is selected. The main area has a search bar 'Select Group to Modify: tw_'. A list of groups is shown, with 'tw_admins' highlighted. To the right, a panel for 'tw_admins' shows it is a 'Team Manager Group' with no members listed under 'Add Members'.

To see all groups, enter three spaces. For more details, see Security Groups.

Bulk User Attribute Assignment

Attributes associated with users can be setup in this area. We can select an attribute and see the values that each user has for that attribute or else we can select a user and see all the attributes for that user.

Here we see the value of an attribute for a set of users.

The screenshot shows the 'Bulk User Attribute Assignment' page. The sidebar includes 'envName1' and 'IBM BPM Admin' sections. Under 'User Management', 'Bulk User Attribute Assignment' is selected. The main area has two tabs: 'View by Attribute' (selected) and 'View by User'. It shows a list of attributes: 'Alert On Assign And Run', 'Image', 'Locale', 'Primary Role', 'Task Email Address' (selected), and 'Task Notification'. To the right, a 'Select Users' section allows selecting users and assigning values. A table lists users with their current attribute values. At the bottom, there's a section for specifying a value and assigning it to selected users.

and here we see the attributes for a given user:

The screenshot shows the 'envName1' environment in the IBM BPM Admin console. The 'Bulk User Attribute Assignment' page is open. On the left, there's a sidebar with various administrative tabs. The main area has a title 'Bulk User Attribute Assignment' and two radio buttons for 'View by Attribute' and 'View by User'. Below that is a section 'Select a User' with a dropdown menu containing several user names. To the right, there's a section 'Assign Attribute Values' with fields for 'User ID' (set to 1), 'User Name' (tw_admin), and 'Full Name' (Internal TW Admin user). There's also a section for 'Custom Attributes' with fields for Alert On Assign And Run (true), Image, Locale (en), Primary Role, and Task Email Address.

The default attributes for a user are:

- Alert on Assign and Run
- Image
- Locale
- Primary Role
- Task Email Address
- Task Notification

If a user attribute is defined as having values from a list of possible values then a drop-down entry is displayed from which the possible values can be selected:

This screenshot shows the same 'Bulk User Attribute Assignment' page as above, but with a specific focus on a dropdown menu. The dropdown is located in the 'Assign Attribute Values' section under the 'Task Notification' row. It starts with '-Select Attribute-' and then lists 'a', 'b', and 'c'. A large red arrow points to this dropdown menu, highlighting it as the point of interest.

See also:

- Security

Process Admin - Monitoring

See

Process Admin - Event Manager

Within the Event Manager section of the Process Admin Console we are presented with a number of options relating to event processing within IBPM.

Event Manager > Monitor

Within the Monitor section, we are shown a list of the scheduled items that are to be executed when the event time is reached:

The screenshot shows the Process Admin Console interface. The top navigation bar includes 'Server Admin' (highlighted in green), 'Deployed Apps', and user information 'Logged in as admin | Preferences | Log out'. The left sidebar under 'envName1' has sections for IBM BPM Admin, User Management, Monitoring, Event Manager (with 'Monitor' selected), Blackout Periods, Synchronous Queues, EM JMS Error Queue, Admin Tools, and My Password Management. The main content area is titled 'Event Manager > Monitor'. It displays a table with one row:

Scheduler ID	Status	Connect expiration	# Jobs Executing
win7-x64	● ●	Jul 10, 2011 10:25:31 AM	0

Below the table are buttons for Refresh, Pause, Resume, Pause All, and Resume All. A summary at the bottom indicates 'Total Jobs Executing: 0' and 'Total Jobs: 1'. The table below lists scheduled jobs:

Scheduler	Process App / Toolkit	Snapshot	Job Name	Job Queue	Scheduled Time	Last Scheduled Time	Last Execution Time	Next Scheduled Time	Job Status
Process Portal	7.5.0	Execute UCA Periodic SLA Update, on set schedule		SYNC_QUEUE_1	7/10/11 10:30:00 AM	7/10/11 10:15:00 AM	7/10/11 10:15:00 AM	7/10/11 10:45:00 AM	Scheduled

Event Manager > Blackout Periods

We can define periods of time when the scheduling of events should be suspended. This can be set in the Event Manager > Blackout Periods

The screenshot shows the Process Admin Console interface. The top navigation bar includes 'Server Admin' (highlighted in green), 'Deployed Apps', and user information 'Logged in as admin | Preferences | Log out'. The left sidebar under 'envName1' has sections for IBM BPM Admin, User Management, Monitoring, Event Manager (with 'Blackout Periods' selected), Blackout Periods, Synchronous Queues, EM JMS Error Queue, Admin Tools, and My Password Management. The main content area is titled 'Event Manager > Blackout Periods'. It displays two panels: 'Blackout Periods' (empty) and 'Blackout Period Details'. The 'Blackout Period Details' panel contains two sections: 'Date/Time Range' and 'Weekday/Time Range'. Both sections have 'From' and 'To' fields with a date/time format of '(M/d/yy) (HH:mm)'. At the bottom are 'Delete', 'Add', 'Update', and 'Clear' buttons, and a note: 'The blackout period is displayed in your local timezone.'

Event Manager > Synchronous Queues

When the Event Manager is ready to dispatch a job for execution, it can place the request in a queue. In IBPM there are two "types" of queues. Asynchronous queues and synchronous queues. If a job is dispatched through an Asynchronous queue then the work associated with that job does **not** need to complete before the next job can be submitted for processing.

For synchronous queues, when work is dispatched to a synchronous queue, that work must be completed before the next piece of work can be processed from the same queue.

By default, IBPM provides three synchronous queues but others can be added. A synchronous queue is utilized in an UCA definition.

The screenshot shows the 'Event Manager > Synchronous Queues' page. The left sidebar is titled 'envName1' and includes links for IBM BPM Admin, User Management, Monitoring, Event Manager (with Monitor and Blackout Periods), Synchronous Queues (which is selected and highlighted in blue), Admin Tools, and My Password Management. The main content area has a title 'Event Manager > Synchronous Queues'. On the left, a panel titled 'Synchronous Queues' lists three entries: SYNC_QUEUE_1, SYNC_QUEUE_2, and SYNC_QUEUE_3. On the right, a panel titled 'Synchronous Queue Details' has a 'Description' field containing an empty text box. At the bottom are four buttons: Delete, Add, Update, and Clear.

Event Manager > Event Managed JMS Error Queue

If the Event Manager is unable to dispatch a job for processing, the "data" associated with that job is saved in an error queue managed by the underlying JMS subsystem. The content of this queue can be examined:

The screenshot shows the 'Event Manager > EM JMS Error Queue' page. The left sidebar is identical to the previous one. The main content area has a title 'Event Manager > EM JMS Error Queue'. Below it is a message: 'This page displays records in the JMS error queue used by the event manager that have not been processed because of errors or that it was in a Blackout Period.' There are four buttons: Delete All, Delete Selected, Reprocess All, and Reprocess Selected. Below these are navigation buttons: First, Previous, Page: 1, Next, Last. At the bottom is a 'Select' button followed by a 'Message Data' link.

Admin Tools > Manage EPVs

As discussed here: Exposed Process Values (EPVs), EPVs allow us to dynamically change values available to a process at run-time. These values can be changed through the Process Admin Console in the Admin Tools > Manage EPVs section:

IBM | Process Admin Console Server Admin Process Inspector Installed Apps Logged in as admin | Preferences | Log out ?

Admin Tools > Manage EPVs

Exposed Process Values Feedback: [Email](#)

Snapshot: Select Snapshot Name: Select EPV

IBM BPM Admin
Manage Caches
Task Cleanup
User Management
Monitoring
Event Manager
Admin Tools
Manage EPVs ←
My Password Management
Saved Search Admin

When selected, the Snapshot (which includes acronym and snapshot name) is selected from the snapshot pull-down. The lists of EPVs associated with that entity are then selectable and one should be selected for the Name field.

The list of variables in the named EPV container is then displayed along with their current values:

Admin Tools > Manage EPVs

Exposed Process Values Feedback: [Email](#)

Snapshot	Test1 (Main ,Tip)
Name	MyEPV
Description	This is the description of MyEPV

Variable List

Variable Name	Description	Current Value
Untitled1	Untitled 1 desc	Hello World
Untitled2	Untitled 2 Desc	Def Val 2
Untitled3	Untitled Desc 3	Def Value 3

A particular variable name may then be selected at which point a new panel appears allow us to set a new value:

Variable List

Variable Name	Description	Current Value
Untitled1	Untitled 1 desc	Hello World
Untitled2	Untitled 2 Desc	Def Val 2
Untitled3	Untitled Desc 3	Def Value 3

Schedule Change / View Details

Effective Date	Value	Modified By	Modified Date
12/11/2010 19:14:19	Hello World	tw_admin	12/11/2010 19:12:41
12/11/2010 19:11:40		tw_admin	12/11/2010 19:10:45

→ [New](#) [Edit](#) [Delete](#)

Clicking on the **New** button opens a window in which a new value for the variable may be defined including a date and time at which that new value should become effective:

The dialog box has a title bar 'View site information'. It contains the following fields:

- Variable name: GL
- Reason: (empty text area)
- Value: 5/13/2013
- Schedule to interval: 19:01:21

At the bottom are 'OK' and 'Cancel' buttons.

The ability to schedule EPV variable changes for the future is incredibly useful. If we know in advance that some law or rule is to take effect from a given date/time then we can define that those changes can take effect in the future automatically. If the changes were not automatic but were instead manual

Note that the effective date/time must be in the future. Unfortunately, the date/time set by default is "now" which means that if it isn't changed to a future date, pressing the OK button will result in an error. Once a value has been created for a scheduled future change, that value and date/time may be edited until the change happens.

Variable List		
Variable Name	Description	Current Value
Untitled1	Untitled 1 desc	Hello World
Untitled2	Untitled 2 Desc	Def Val 2
Untitled3	Untitled Desc 3	Def Value 3

Schedule Change / View Details			
Effective Date	Value	Modified By	Modified Date
12/11/2010 21:22:53	My New Value	tw_admin	12/11/2010 19:26:04
12/11/2010 19:14:19	Hello World	tw_admin	12/11/2010 19:12:41
12/11/2010 19:11:40		tw_admin	12/11/2010 19:10:45

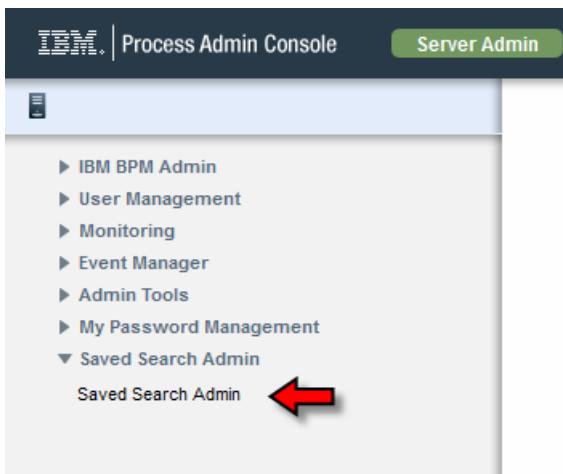
After a change has happened, the change becomes fixed until another change request entry is created. The history of changes including the old values, who changed them and when the changes were made are recorded for subsequent viewing.

See also:

- Exposed Process Values (EPVs)

Process Admin – Saved Search Admin

Under the Saved Search Admin area we can find the entry to create Saved Searches. A Saved Search is a way to define searches that can be given a name and then a user can execute such a search merely by providing that name.



Once selected, the Saved Search editor is presented:

The screenshot shows the 'IBM BPM Search Criteria Editor' interface. At the top, there is a 'Select Search:' dropdown with 'Define New Search...' and a 'Select' button. Below it is a 'Search Name:' field with a required asterisk (*). There are three tabs: 'Columns', 'Conditions', and 'Ordering'. Under 'Columns', there is an 'Add' button. Under 'Conditions' and 'Ordering', there are also 'Add' buttons. Below these tabs is a 'Search Organized By:' dropdown set to 'Task'. At the bottom, there is a 'Search Results' panel with a search bar, a 'Save New Search' button, and a 'Cancel' button. The text 'Search Id: 0' is displayed at the bottom left of the results panel.

From here we can enter a description of the search we wish to have saved.

If we wish to build and create saved searches programmatically, we can use the REST API for searching which has a "Save As" option.

See also:

- Process Portal – Saved Searches
- Working with REST Search Queries

Process Admin – Process Inspector

Within the Process Admin console is a primary section called "Process Inspector". This is a search and filter mechanism to allow the BPM administrator to view the state of the system. It is not intended to be used by any kind of business user. At the highest level, we can execute a search for process instances. We can filter those processes by a variety of different states, owners and date ranges. The result is a set of processes that meet the desired criteria. The following is an example screen shot of such a search:

The screenshot shows the IBM Process Admin Console interface. The top navigation bar includes links for Server Admin, Process Inspector (which is highlighted in green), and Installed Apps. On the right, it shows the user is logged in as Neil with links for Preferences and Logout.

The main content area is titled "Process Inspector" and displays a search results table. The table has columns for "Select All", "Process Name", "Status", and "Last Action". There are three results listed:

- Wire Transfer - Priority629**: Status Active, Last action 34 minutes ago by WT_REQ due 16 hours ago.
- Hire New Employee 617**: Status Active, Last action 9 hours ago by dadmin due 3 days ago.
- Hire New Employee 615**: Status Active, Last action 10 hours ago by dadmin due 3 days ago.

To the right of the table, there is a detailed view for the first process, "Wire Transfer - Priority629". It shows the following details:

- Status:** Active
- Start time:** 2013 September 16 11:10:46 AM
- Last action:** 2013 September 17 11:08:04 AM
- Due date:** 2013 September 16 7:10:46 PM
- Actions:** Suspend (disabled) and Terminate
- Tasks:** Request Wire Transfer (WT_REQ closed task due 1 day ago), Approve Wire Transfer (The task is owned by Wire Transfer Approve, due 23 hours ago).
- Timers:** Untitled1

The filtering rules available include:

- Process states
- Process names
- Person – The person who started the process
- Date ranges

After executing a search, the results are shown in the middle of the page. Selecting a process then shows addition details on the right of the page as well as actions that can be performed against that process instance.

Each process instance has an icon beside it. These icons are:

Icon	Description
Flag icon	Active
Checkmark icon	Completed
Red circle icon	Terminated
Two vertical bars icon	Suspended
TBD icon	Failed

Process Admin - Installed Apps

The Installed Apps section of the Process Admin Console allows us to see which applications and their snapshots are currently deployed on the Process Server. From here we can perform some administration.

Application Name	Status
App01 (APP01) - SN1	Active, Default
App01 (APP01) - SN2	Active
Hiring Sample (HSS) - Hiring Sample 3	Active, Default
Mon1 (MON1) - sn3	Active, Default
Process Portal (TWP) - 7.5.0	Active, Default
UCATests (UCA1) - SN6	Active, Default

If we select an individual application, we have options including:

- Deactivating the application
- Migrating data
- more ...

Exposed Item	Value
Start Process UCA	8bfa1d71-9e38-4d63-be17-ff59b99c403a
ScheduledUCA	Scheduled
Default BPD Event	defaultBPDEvent

For applications which have UCAs associated with them, we can choose to enable/disable their execution.

Customizing the Process Admin Console

The Process Admin Console contains a list of actions that can be performed in the left hand side of the web page:



These actions are configurable. New ones can be added and the existing ones removed. This allows administrators to add new functions specific for the local environment. To add a new entry, create a new Human Service within a Process Application. In the Exposed As selection, choose Administration Service. This is the attribute that says it will be exposed in the Process Admin Console.

My Human Service

Overview Diagram Variables Coaches

Common

Name: My Human Service
Modified: admin (Mar 23, 2012 11:39:15 AM)
System ID: guid:56d35d2f221c8d0e-119f05cc:13640
Documentation:

Exposing

Exposed to: A...s Sys...ata Select... New ... X
Exposed As: Administration Service

(See also: Exposing the Human Service for starting)

When a user logs into the Process Admin Console, they will see a new entry corresponding to the Process Application in which the service is defined. Underneath the Process Application, will be the name of the new Human Service definition. Selecting that entry will cause the service definition to run.

To remove the original entries for the console, the file located at:

```
<Profile Root>/config/cells/cell name/node/node name/servers/server name/process-server/config/console.xml
```

can be edited. This XML file contains entries for the folders and choices that appear by default on the console. Rather than delete the entries, it is suggested that they be commented out by bracketing the XML element with the <!-- and --> pair.

Process Portal

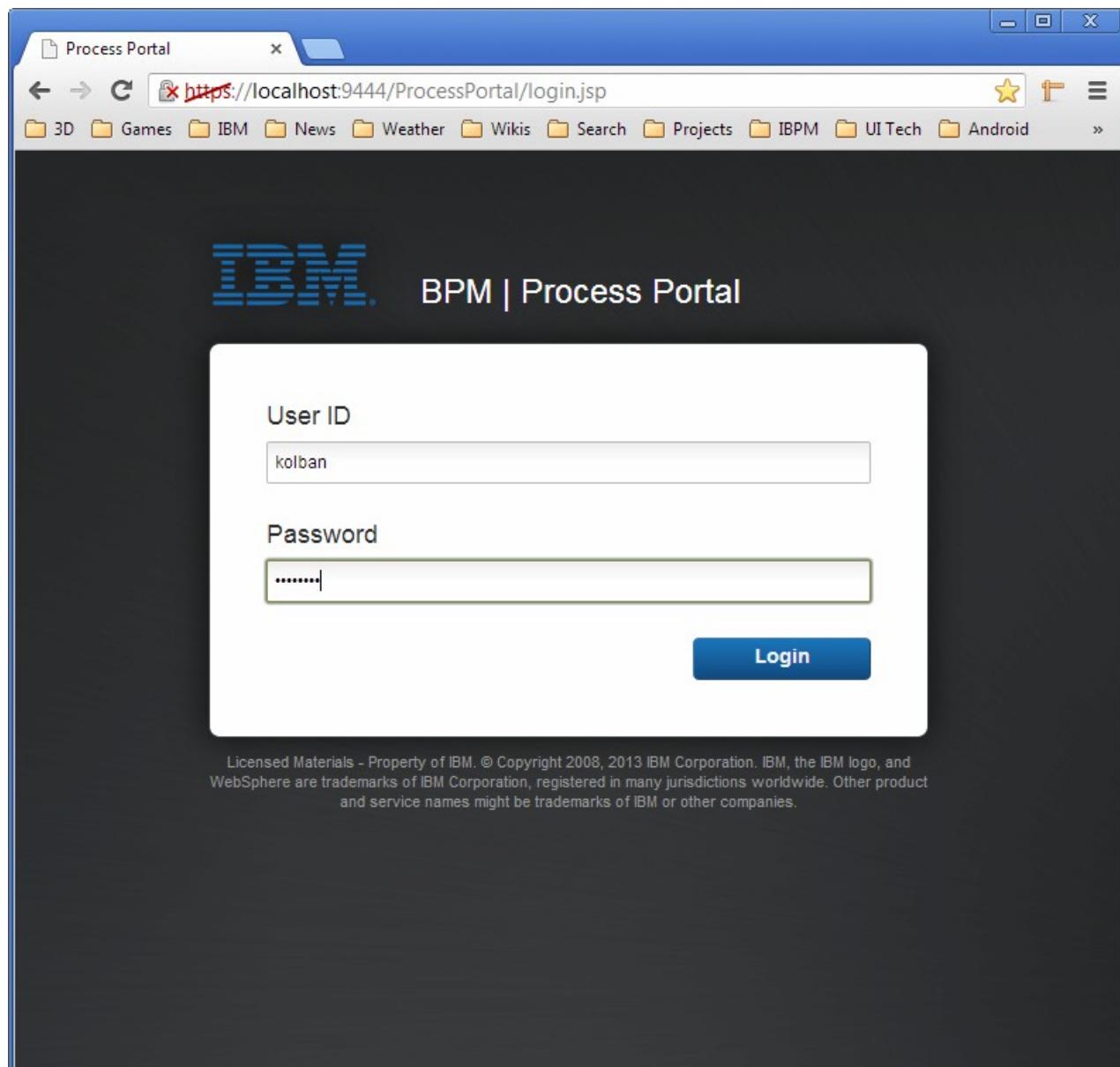
With the release of IBM BPM v8, the whole implementation of IBM Process Portal was reworked. The following focuses on IBM BPM v8 and above. If you are running a version of IBM BPM prior to that release, this information will not be applicable to you.

Process Portal is the IBM supplied environment from which users can interact with processes managed by IBM BPM. Process Portal is a browser based framework for interacting with process instances.

It can be reached at the following URL:

<https://localhost:9444/ProcessPortal/login.jsp>

From that location we are prompted to login with a userid/password pair.



After logging in we are shown the primary portal page.

The screenshot shows the 'Process Portal' interface. At the top, there's a navigation bar with links like '3D', 'Games', 'IBM', 'News', 'Weather', 'Wikis', 'Search', 'Projects', 'IBPM', 'UITech', 'Android', 'Google Docs', and 'Bookmarks'. Below the navigation bar, there are three main tabs: 'WORK' (selected), 'TEAM PERFORMANCE', and 'PROCESS PERFORMANCE'. The 'WORK' tab has a sub-section titled 'My Work'. Under 'My Work', there's a 'My Tasks' section with a search bar and a link to 'Open Tasks | Completed Tasks'. The 'My Tasks' section is divided into sections: 'Overdue' (empty) and 'Due Today (1)'. A button labeled 'Create and submit a new form' is shown, along with a note 'Emory Forms:382' and a due date 'Due: July 16, 2013 12:10 PM'. To the right of the task list, there are buttons for 'Launch', 'Following', and '@Mentions'. A message states 'Either you are not authorized to launch any processes, or no processes are installed.' At the bottom of the screen, there's a footer bar with the 'IBM' logo.

Let us start by looking at a user's settings. In the top right of the screen is the your name. This is actually click-able and can be used to change your own profile settings.



When selected, a page is shown which shows the current details.



Neil Kolban
Job Title

Phone Number
Email Address

[Change](#)

Portal Preferences

Language

Calendar Type

Show a confirmation window when I open an unclaimed task

Send me an email when I have a new task assignment

Send me an IBM Connections notification when I have a new task
IBM Connections is not configured for activity notifications.

Show a confirmation message when I stop following a process

Display a notification when

A new task is assigned to me

A new task is assigned to my team

I'm mentioned in a post

Update

Most of these details can be changed.



Administrator
Senior Dish Washer
817-555-5555
ceo@ibm.com

Change | Remove

Portal Preferences

Language English ▾

Calendar Type Gregorian ▾

Show confirmation window upon running a task

Send me an email when I have a new task assignment

Show a confirmation message when unfollowing a process

Display a notification when

A new task is assigned to me

A new task is assigned to my participant group or groups

I'm mentioned in a post

Update

The image used to represent the user is shown in a number of other dashboards. The biggest image seen in the web pages seems to be 105 pixels square.

Process Portal – My Tasks

The primary area of the portal is called "My Tasks" and shows lists of tasks awaiting your attention. It can be reached by clicking on the "Work" tab:

The screenshot shows the 'My Tasks' interface. At the top, there's a search bar and links for 'Open Tasks' and 'Completed Tasks'. Below that, a section for 'Overdue' tasks is shown. Under 'Due Today (1)', there's a single task entry:

- Create and submit a new form** (Emory Forms:382)
- Due: July 16, 2013 12:10 PM

At the bottom of the list, it says 'Showing 1 of approximately 11 results'.

The tasks are grouped into categories:

- Due Today – Tasks which should be completed today
- At Risk – Tasks which should be completed very soon
- Overdue – Tasks which should already have been completed

The tasks shown in the My Tasks section have two primary textual eye-catchers. One is taken from the task's Subject while the other is taken from the task's Narrative:

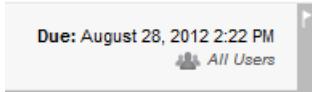
The screenshot shows the 'Properties' dialog for a task. The 'Implementation' tab is selected. In the 'Task Header' section, the 'Subject' field contains the text 'This is my subject line' and the 'Narrative' field contains the text 'This is my narrative'. Red arrows point to both of these fields.

Properties		Validation Errors	Where Used
General	Implementation		
Simulation	User Task		
Implementation	Review C		
Assignments	Task Header		
Data Mapping	Clean State:		
Pre & Post	Subject:	This is my subject line	
KPIs	Narrative:	This is my narrative	
Condition			

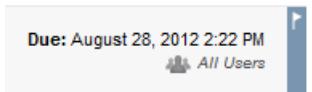
Immediately beneath the subject is the instance name of the process.

To the very right of the task entry is a flag which colors according to the priority of the task:

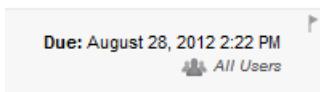
Very Low



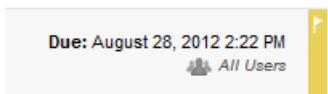
Low



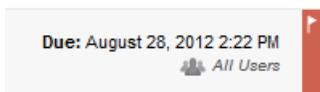
Medium



High



Very High



If a task is assigned to a group you are a member of (as opposed to assigned to you), it will have an indication of such:

▼ Due Today (1)

Step: Review Customer BPD1:54

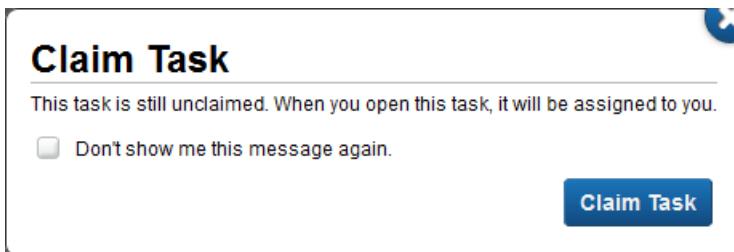
Due: June 1, 2012 3:00 PM All Users

This indicates that it is both associated with a group and names the group as well.

To work upon a particular task, we click the title of that task.

Step: Review Customer BPD1:54

If you do not already own the task (i.e. it is owned by a group that you are associated with) a dialog appears confirming your request:



The dialog can be disabled so that it is not shown each time a new claim is made. When a task is opened, the Coach associated with the Task is displayed.

Step: Review Customer

Due: June 1, 2012 3:00 PM 

Review Customer

name

address

phoneNumber

Button 1

To the right of the Coach area is a summary of information about the process instance which owns the Coach:

Details Stream Experts

☆ BPD1:54

Due: June 15, 2012 2:43 PM

 [View Process Diagram](#)

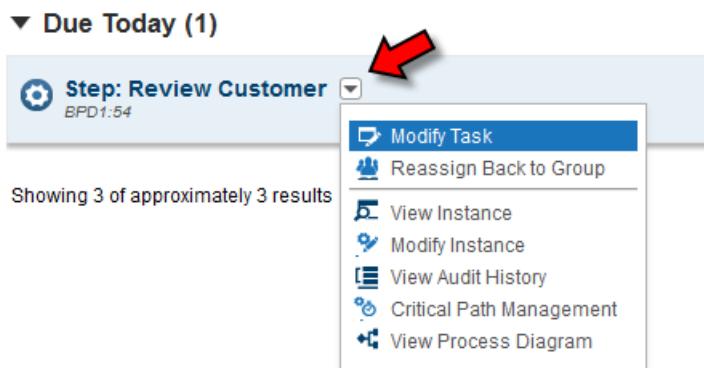
Tasks

 Step: Review Customer

 admin

Created: June 1, 2012 2:43 PM
Due: June 1, 2012 3:00 PM

Each task has a context menu associated with it:



The context menu allows a number of administrative items to be performed:

Modify Task	The ability to assign the task to another person or group, change the priority or change the due date/time:
-------------	---

	<p>Modify Task</p> <p>Escalate or modify the settings for task Step: Review Customer.</p> <p>Task Assignment</p> <p><input checked="" type="radio"/> <input type="radio"/> Reassign back to group</p> <p>Priority</p> <p>Medium</p> <p>Due Date</p> <p>6/1/2012 3:00 PM</p> <p>Save</p>
Reassign Back to Group	Reassign the task back to the group which originally owned it.
View Instance	View the process instance associated with the task.
Modify Instance	Modify the process instance. This currently means modifying the process instance due date/time:
	<p>Modify Instance</p> <p>Modify settings for this process instance</p> <p>Due Date</p> <p>6/15/2012 2:43 PM</p> <p>OK</p>
View Audit History	
Critical Path Management	
View Process Diagram	<p>View a diagram showing the overview of the process and where we are in that process.</p> <pre> graph LR Start((Start)) --> Review[Review Customer] Review --> End((End)) </pre>

When working on a particular task, there is a menu box that can be shown that allows you to perform related items to that task:

Step: Review Customer

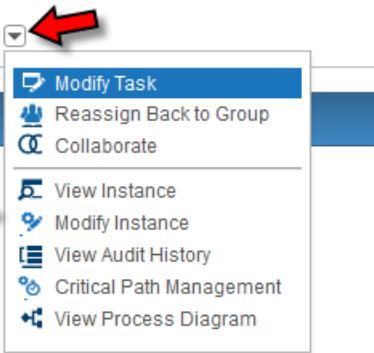
Review Customer

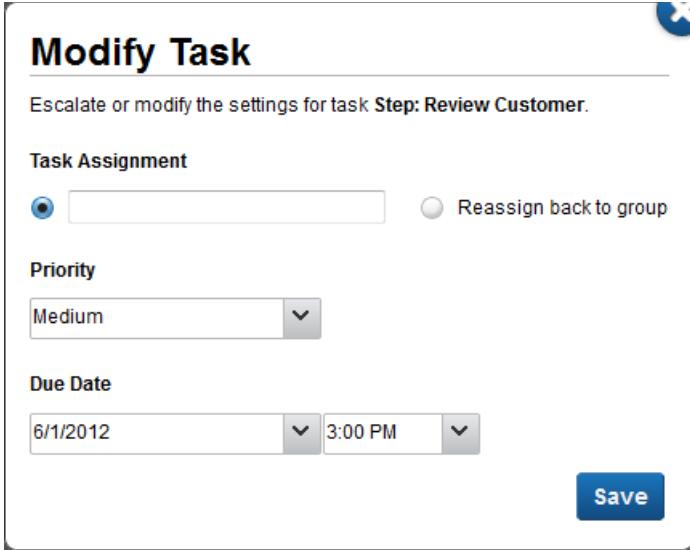
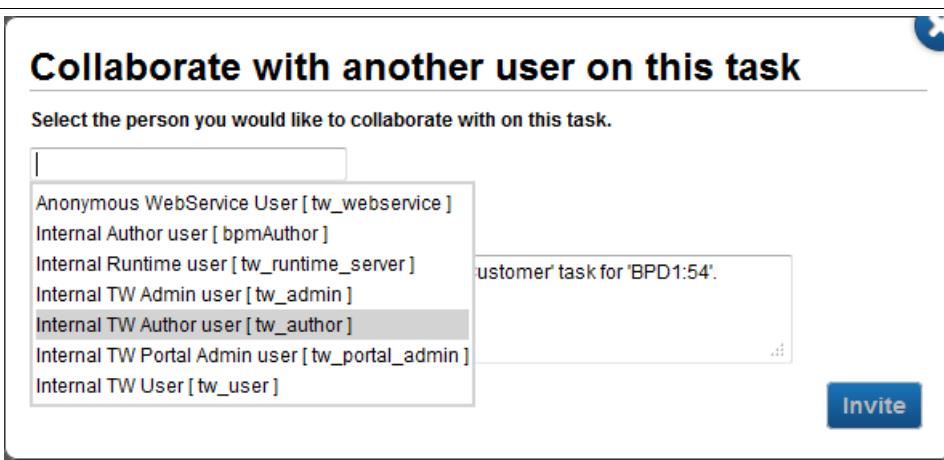
name

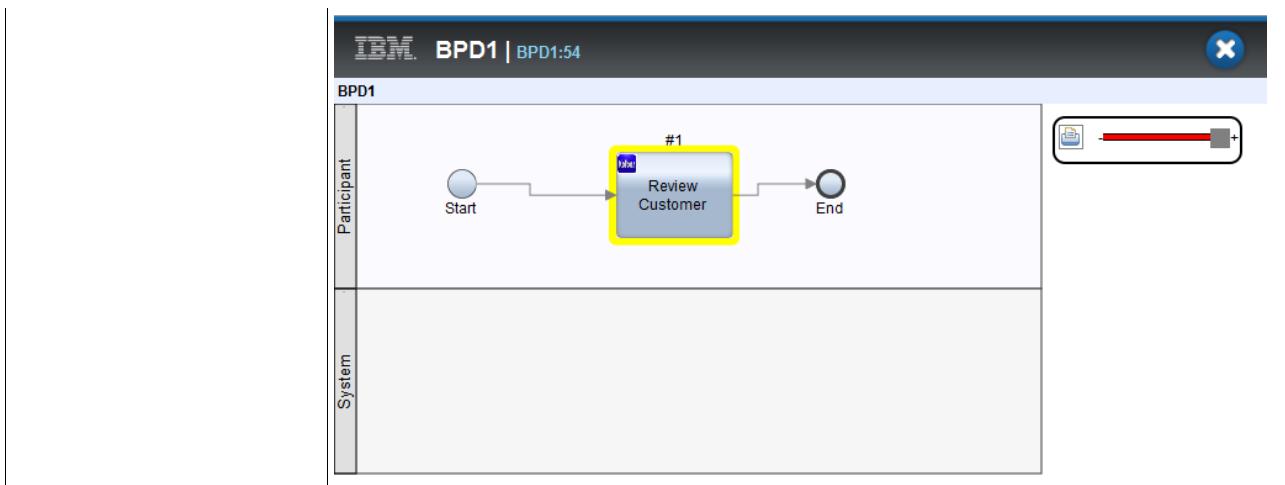
address

phoneNumber

Button 1



Modify Task	The ability to assign the task to another person or group, change the priority or change the due date/time: 
Reassign Back to Group	
Collaborate	
View Instance	
Modify Instance	
View Audit History	
Critical Path Management	
View Process Diagram	View a diagram showing the overview of the process and where we are in that process.



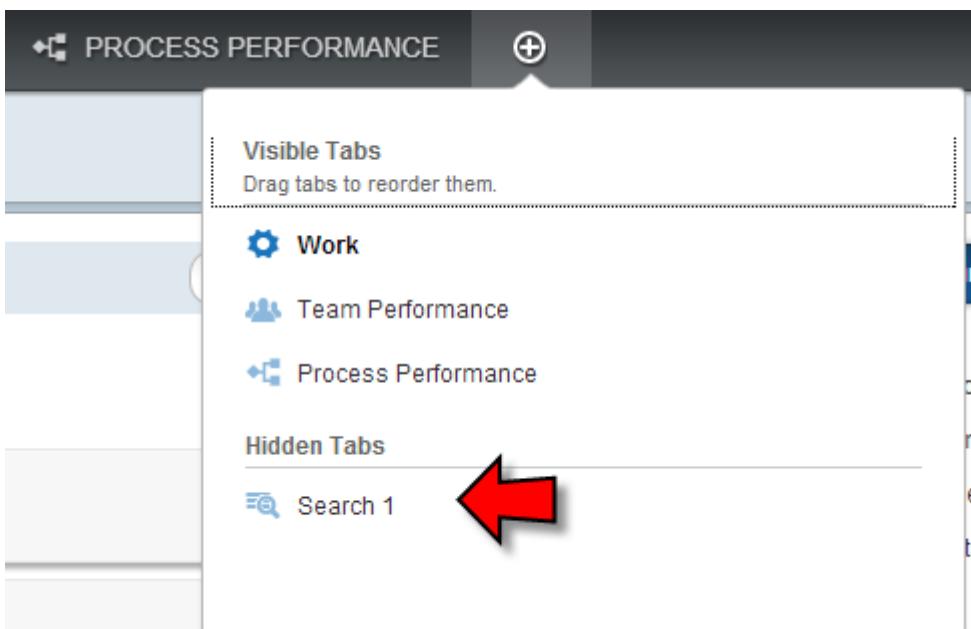
Process Portal – Saved Searches

When there are large numbers of tasks and process instances it can become difficult to find what one is looking for. To solve this problem, IBM BPM provides the capability to create custom searches. These searches are queries made against process instances and task instances which determine what information is to be shown and which tasks and instances qualify to be shown. These searches are generically called "Saved Searches" and can be defined in the Process Admin Console. Once defined, these searches can then be executed from within the Process Portal.

In the Process Portal, we can click the show/add entries button:



From there, a pull-down of options will be shown. Included in the options will be the option to see previously defined saved searches.



By selecting one of these, the search is performed and the results shown in a tabular form.

Instance status	Process definition name	Instance ID	Actions
Failed	Emory Forms	3	
Failed	Emory Forms	4	
Active	Emory Forms	53	

Launch Following @Mentions

- Advanced HR Open New Position
- Emory Forms
- ReplenishmentBPD
- Standard HR Open New Position

To the right of each row there are icons which allow the user to open the task, see details of the process instance associated with the task or change some of the properties of the task.

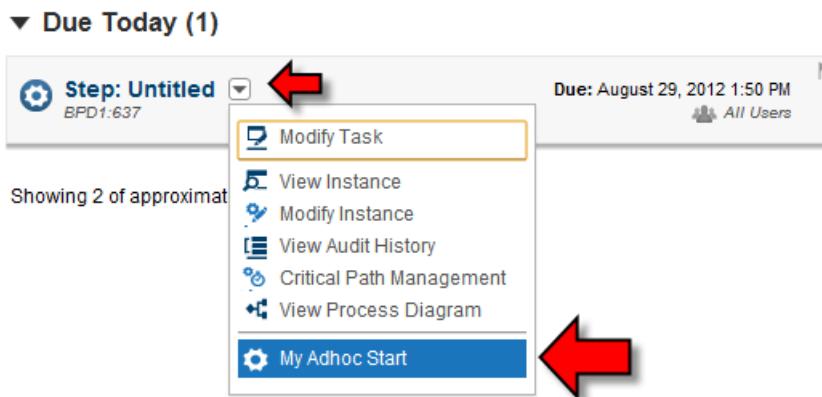
See also:

- Process Admin – Saved Search Admin

Starting Ad-Hoc entry points

A BPD process can have ad-hoc entry points defined with it. This means that for a given instance of a process, additional work can be performed by that process at any time by invoking its ad-hoc entry. Within Process Portal, ad-hoc starting points are shown in both a task instance and a process instance view. Imagine a BPD that has an ad-hoc entry point called "My Adhoc Start". By selecting a task within the BPD process, we see the ad-hoc entry point in the task's pull-down.

Selecting this will start that ad-hoc portion of the flow.



Additionally, looking at the similar area within the Process Instance section we see an additional location from which the ad-hoc start can be fired:

BPD1:637

Due: September 12, 2012 12:50 PM

[View Process Diagram](#)

Tasks

Step: Untitled
Created: August 29, 2012 12:50 PM

All Users
Due: August 29, 2012 1:50 PM

- [Change Process Due Date](#)
- [View Audit History](#)
- [Critical Path Management](#)
- [My Adhoc Start](#)

See also:

- Ad-hoc Start Event
- Securing Process Portal capabilities

In-line Tasks in process portal

When displaying tasks that a user may work upon, the default action is that when the user selects such a task, a new window will appear showing the coach for that task. However, there are certain tasks that may simply need a yes/no answer or some input equally simple. Rather than opening a whole new Coach window, we might simply wish the task to be completed in-line within the process portal.

IBM provides three per-implemented Human Services that can be defined as implementations for a BPD Human Service.

The first is the "Simple Approval". When displayed in the Process Portal, it looks as follows:

Step: Simple Approval BPD3:75

Due: June 4, 2012 1:17 PM

No details found.

Add a comment...

Approve **Reject**

It has data mappings as follows:

Inputs: None

Outputs:

approved	boolean	A true or false outcome. The outcome is determined by whether or not the "Approve" or "Reject" button is pressed. Reject does not become selectable until after comment text has been entered.
comment	string	A piece of comment text that can be used to describe the reason for the outcome. Comment text is required for a rejection.

The second type of inline entry is the "Simple Completion". When displayed in Process Portal it looks as follows:

Step: Simple Completion ▾ Due: June 4, 2012 1:23 PM
BPD3:76 All Users

No details found.

Add a comment...

Complete

It has data mappings as follows:

Inputs: None

Outputs:

comment	string	A piece of comment text that records the completion notes (if any)
---------	--------	--

The third type of inline entry is the "Simple Choice". When displayed in Process Portal, it may look similar to the following:

Step: Simple Choice ▾ Due: June 4, 2012 1:30 PM
BPD3:79 All Users

No details found.

Add a comment...

Yes No Maybe

Inputs:

choices	List of String	A list of possible choices that the user can select
---------	----------------	---

Outputs:

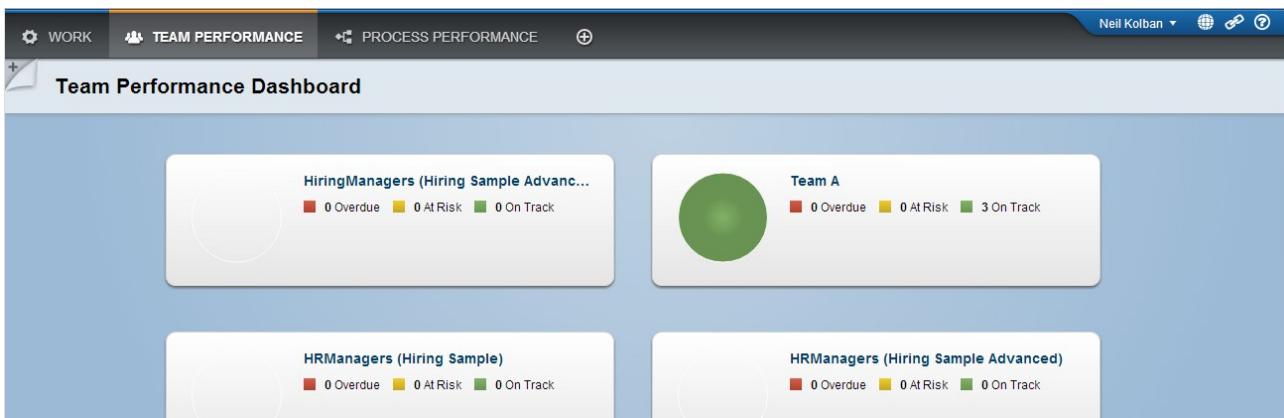
decision	String	The value of the string (button) selected by the user
comment	String	A textual comment optionally provided by the user

Team Performance

Built into Process Portal is the ability to examine work by teams. To view this dashboard, click the Team Performance tab:

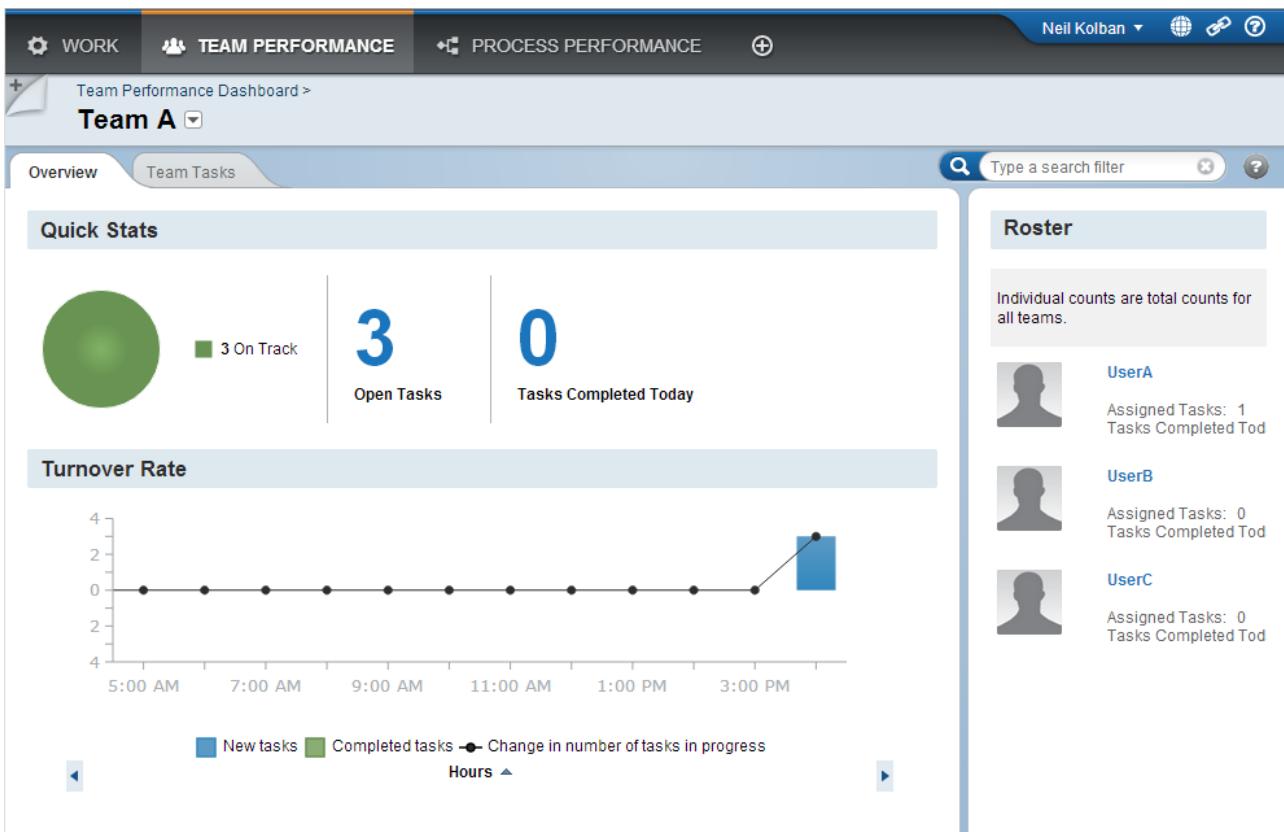


This will bring up a page showing the summary of all the teams:



Each box represents a specific team. A pie chart is shown within the box showing how many tasks are associated with each team and how many are overdue, at risk or on track.

If we drill into a specific team, we are given details of that team:



We see a pie chart for that team, the count of how many tasks are opened and how many completed today. In addition, we see a chart over time of how new tasks have been arriving and how existing tasks have been completed.

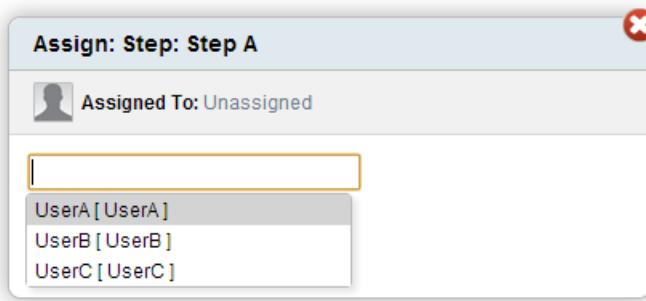
To the right of the chart, we see a roster showing the members of the team.

Team Tasks

In the Team Tasks view we can see a calendar of when tasks are due to be completed. From there we can Assign those tasks to different team members if we are logged in as a manager of the team.

The screenshot shows the Team Performance Dashboard for 'Team A'. The 'Team Tasks' tab is active. In the 'Open Tasks' section, there are three items under 'Due Today (3)'. Each item has a dropdown menu showing 'Step: Step A' and a status message like 'Tuesday: 3 On Track'. The 'Roster' section on the right lists three users: UserA, UserB, and UserC, each with their assigned task counts.

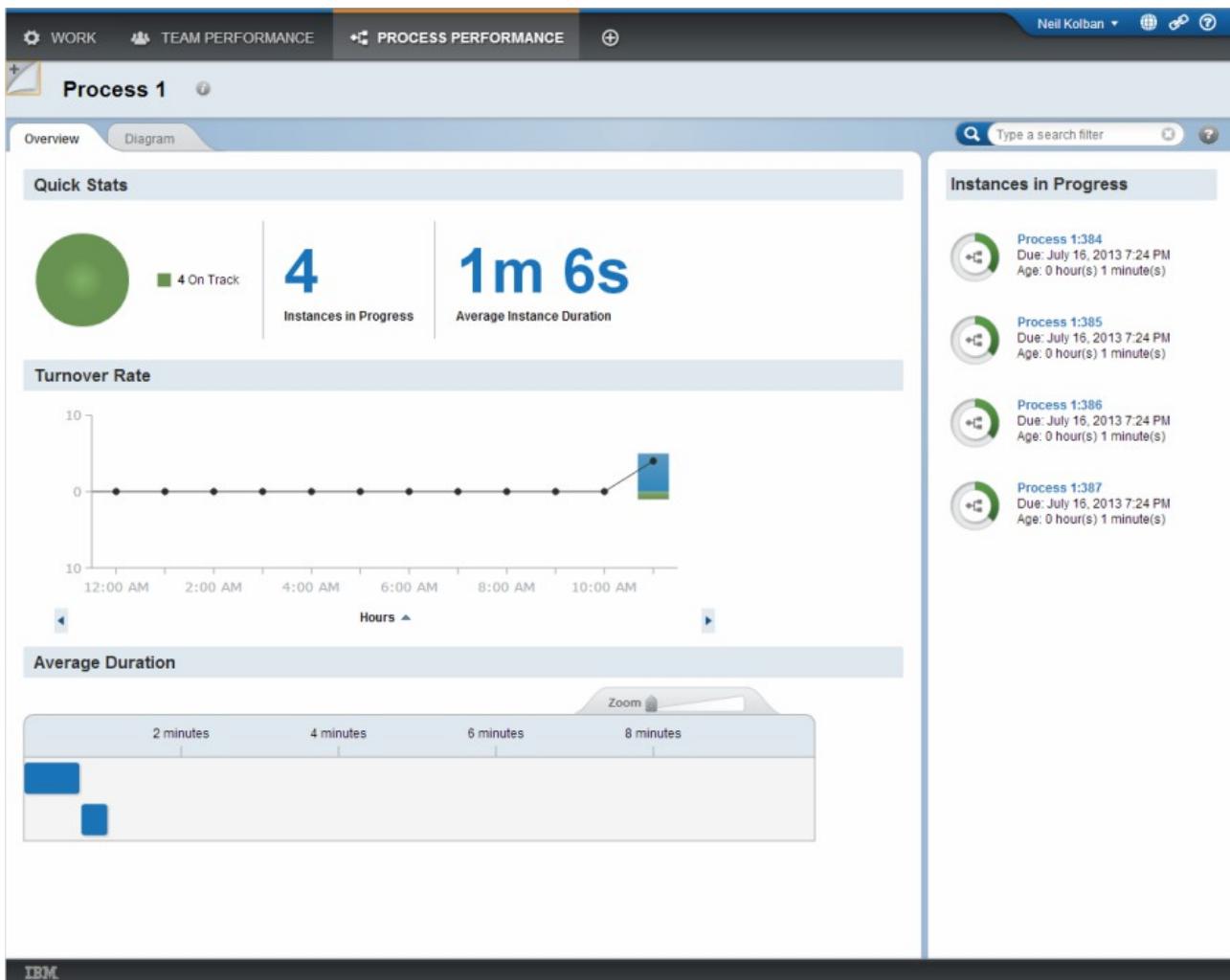
Clicking Assign allows us to change the owner of the task if we are logged in as a manager of the team. The allowable staff members are filtered by the team memberships although we can over-ride those as needed.



Process Performance

Built into the Process Portal is the ability to examine the performance of the overall solution. To view this dashboard, click the Process Performance tab:

This will then display the details of Process Performance that is show in the next screen shot:



As we can see, there is a lot of "stuff" in this screen so we need to spend some time looking at it.

We will break the screen down into four sections:

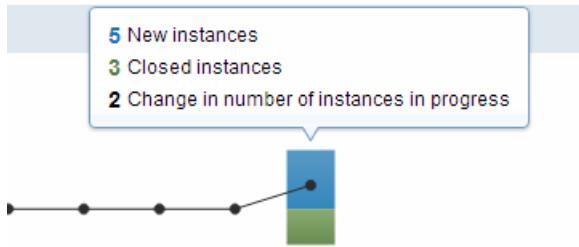
- Quick Stats – shown at the top
- Turnover Rate – shown in the middle
- Average Duration – shown in the bottom
- Instances In Progress – shown on the right

Quick Stats

The Quick Stats section shows the number of processes that have not yet completed. This means that there is work still to be performed against them. The pie chart is split into their states meaning on track, overdue and at risk. Next to the pie chart is a count of the total number of processes in progress. Finally, there is a statistic on the average duration that it takes a process to complete.

Turnover Rate

The Turnover Rate section shows the arrival and completion rate of process instances over time. If we hover our mouse over the chart, additional data is shown:



The chart needs a little explanation. First notice that the mid-line axis of the chart is zero. As new instances arrive, the blue portion (top) of the chart will grow upwards. As process instances are completed, the green portion (bottom) of the chart will grow down. The line chart shows the number of processes in progress.

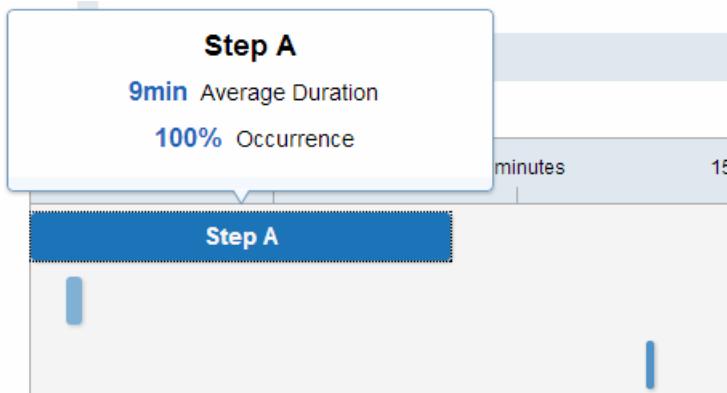
One way to interpret the chart is simply to focus on the line. If it is trending upwards then the processes are not being completed fast enough.

The chart can be scrolled left and right to see previous time intervals. The scale can also be changed between Days and Hours. The hover data shows details of the selected period.

Average Duration

The average duration chart shows an entry for each step in the process (top to bottom). The width of the entry is the average duration that step takes. The left edge of the element is the average point in time when that step in the process is reached.

Each entry is labeled with the name of the step. Hovering over a step shows a pop-up with details which includes the numeric average duration and the percentage of time that step is executed as part of a process instance (remember processes can take different paths and hence not all steps in a process may be executed for each instance of that process).



Instances In Progress

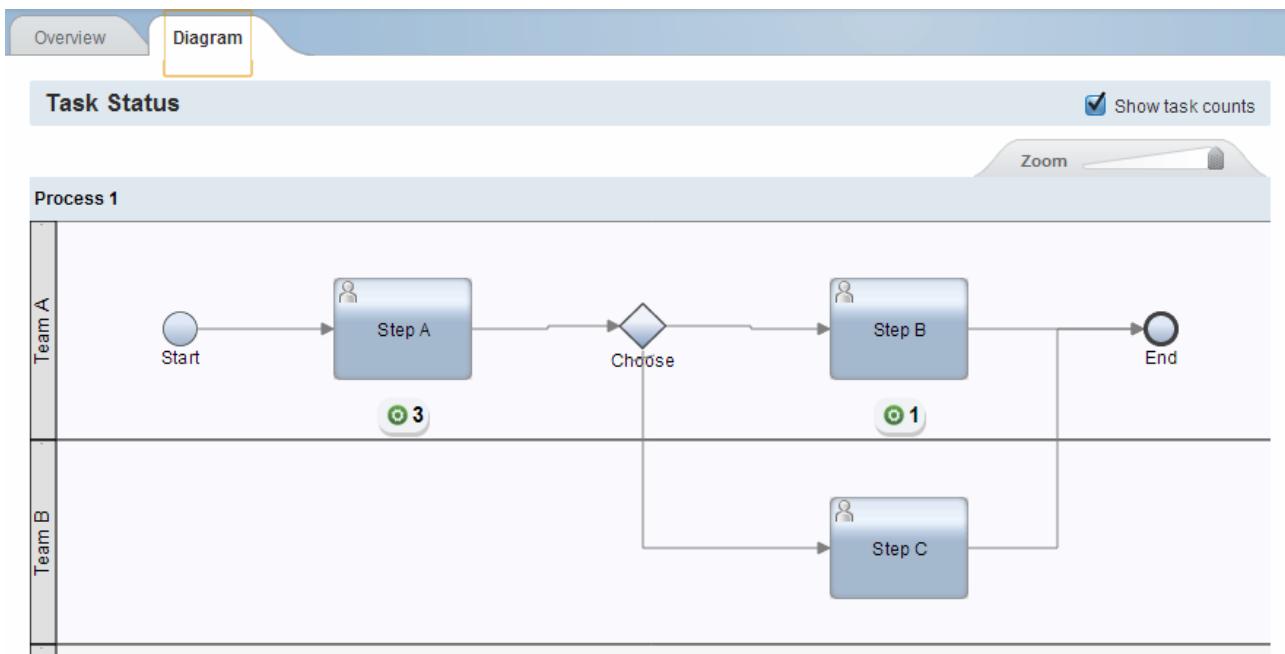
On the far right is a list of the instances currently in progress. Hovering the mouse over these will show their details including:

- Process name
- Due date and time (of the process)
- Age of the process

- Estimated completion date

Diagram View

Clicking on the tab at the top labeled "Diagram" will show the process diagram of the process. The diagram will be annotated with the number of instances at each task step.



Clicking on a task counter will filter the process instance to show only those ones which are blocked at the associated tasks.

Gantt View

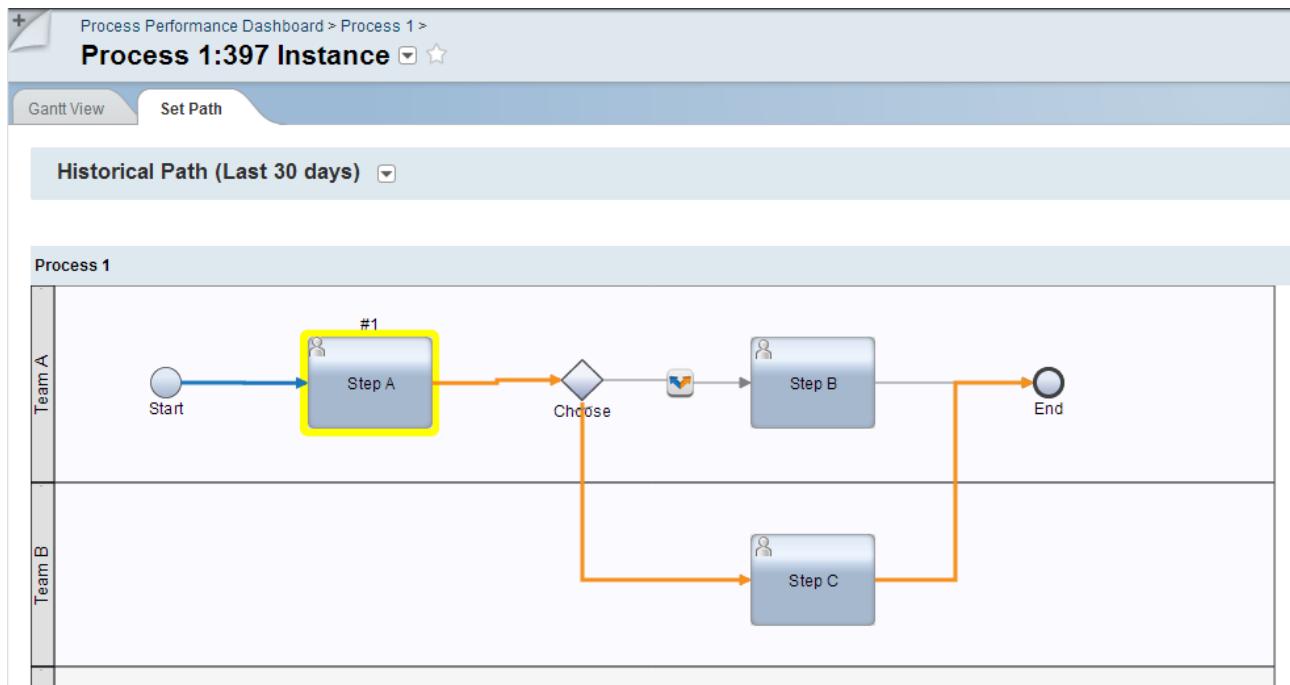
If we click on a process instance in the "Instances in Progress" list, we are shown a Gantt chart view of the expected times of that process instance:



The chart shows pictorially which step we are on and when subsequent steps are potentially going to start and complete.

Set Path View

When determining whether or not a process is on target for completion within its due dates, IBM BPM makes assumptions about the process based on averages of similar processes seen before. In many cases, the path a process takes is a function of the decisions made within that instance of the process. As a user of the Process Performance dashboard, we can see the expected path the process will take and decide to change the anticipated path for reporting and modeling purposes. Changing the path in the Process Performance dashboard has no effect on the actual path that the process will finally take but does allow us to use our judgement when examining such an instance.



From the "Set Path" view we can see the current step, see the anticipated path and change the anticipated path route.

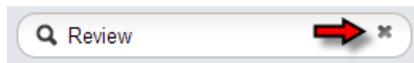
Searching in Process Portal

At the top of many of the portal pages there is a search box that looks as follows:



Entering search criteria within this box and then clicking the magnifying glass will filter what you see to only include matched entries.

To clear the search and show all entries, click the "x" also shown in the search:



The search has a variety of options and meta data filters. Here are some notes on this:

- Searching is not case sensitive. Searching on "John" or "JOHN" will produce the same result.
- The default operator between terms is the logical "and". So if we enter "John Smith" the search will return results which have **both** "John" and "Smith".
- We can search for results which have one value or another using the "OR" operator. If we enter "John OR Smith", we will find entries which have "John" or "Smith".

- We can combine logical operators using bracket notation. For example "(John OR Susan) AND Smith" will find "John Smith" and "Susan Smith".
- We can include and exclude items. For example "+Smith -Susan" will find all the "Smiths" but will exclude any that contain "Susan".
- We can use wild cards in terms. The "?" character is a single character. The "*" character represents a set of one or more characters.
- We can match on exact terms by enclosing the search in quotes. Wild card characters will not be honored.

We can also prefix search queries with the context in which they should be sought. For example, searching on "subject:IBM" will find all tasks which contain "IBM" in their subject line but will not return entries that contain "IBM" within other properties.

The list of prefixes are:

Prefix	Description
subject:	The subject line of a task.
instancename:	The instance name of a process.
application:	The name of the process app.
atriskdate:	A range of dates for the At Risk value of a task.
bpd:	The name of a BPD (Process).
createdate:	A range of dates for the creation of an entity.
duedate	A range of dates for the due date of a task or process.
username:	The user id of a user.
userfullname:	The real name of a user.
teamname:	The name of a team.
isassignedtouser:	Whether or not a task has been claimed or assigned to a specific user.
activityname:	The name of the Task.
<Business Data>:	The value of a specific business data item.

The data used to populate the Process Performance dashboard

As a BPM process runs, the history and state of that process is written to the database used by IBM for internal and operational purposes. It is that data that is used to populate the Process Performance dashboard.

The data that is used can be managed through the Process Admin console through the Process Inspector area.

Customizing Process Portal look and feel

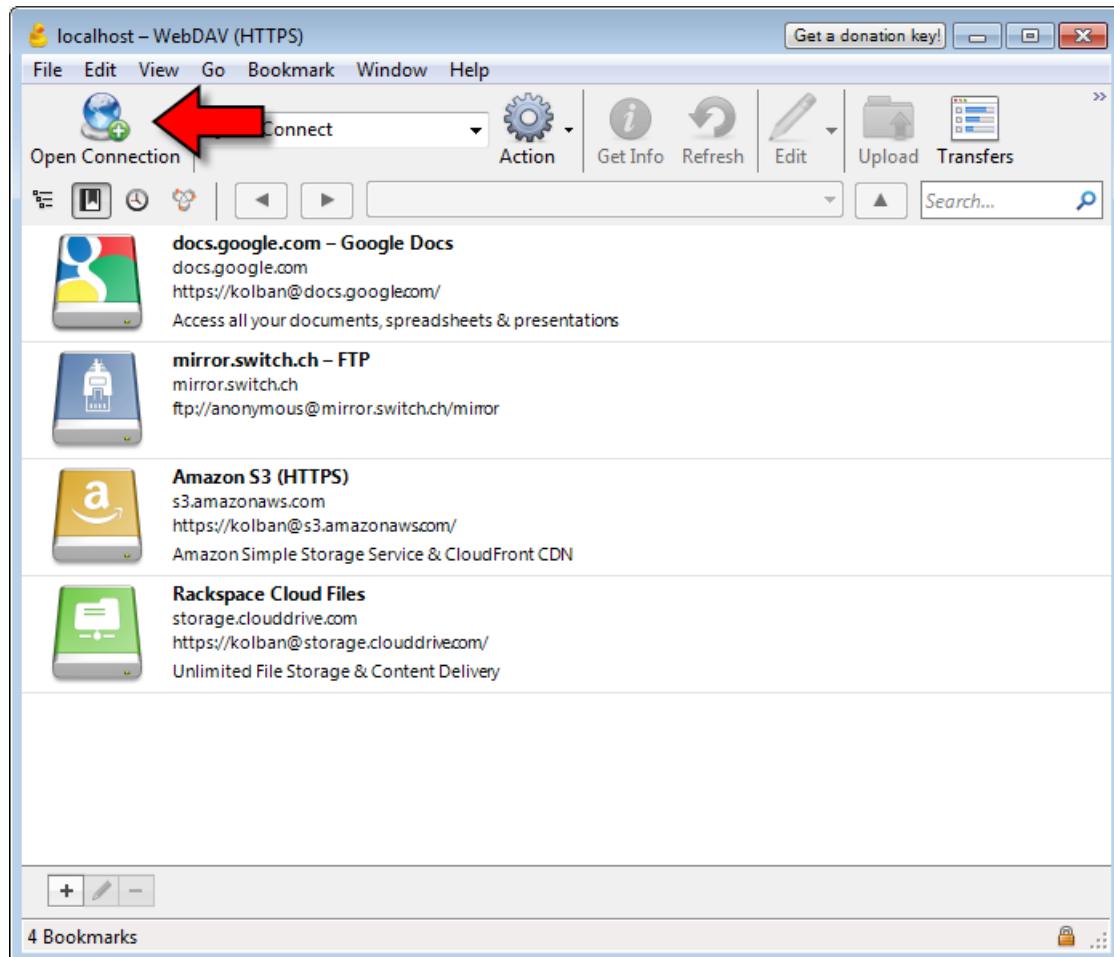
The Process Portal supplied by IBM has a distinct look and feel. Given that the Process Portal is the suggested mechanism for interacting with end users, we wish to be able to customize the Process Portal so that it may look more like a particular user community would like it to look.

IBM has exposed a way to achieve that task.

Process portal's design is achieved through a series of files that include images, CSS, JavaScript and HTML. These files can be updated or replaced with alternatives to change the style.

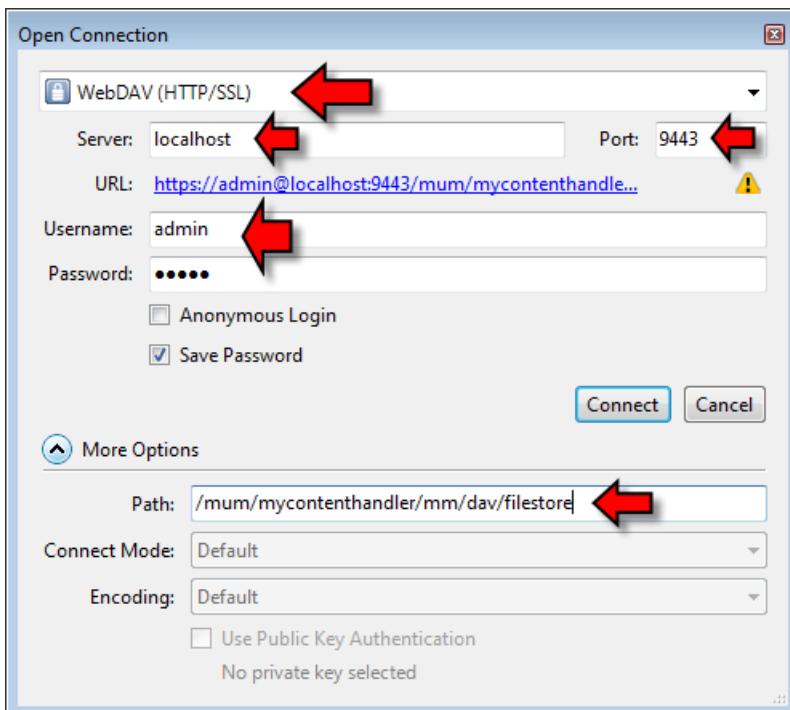
Access to these files is achieved through WebDAV. It is believed that Windows 7 claims to have some WebDAV support but the reality is that no-one seems to have been able to get it working (at least that is what I find when I search Google). The alternative is to use a client tool to access the WebDAV data. I performed an Internet search of WebDAV client tools and found "BitKinex". However, I was not able to get that tool working with IBM BPM's WebDAV data. I then found a second tool called "Cyberduck" which worked just fine.

After installing Cyberduck, I configured a new connection within it to access the BPM WebDAV data.

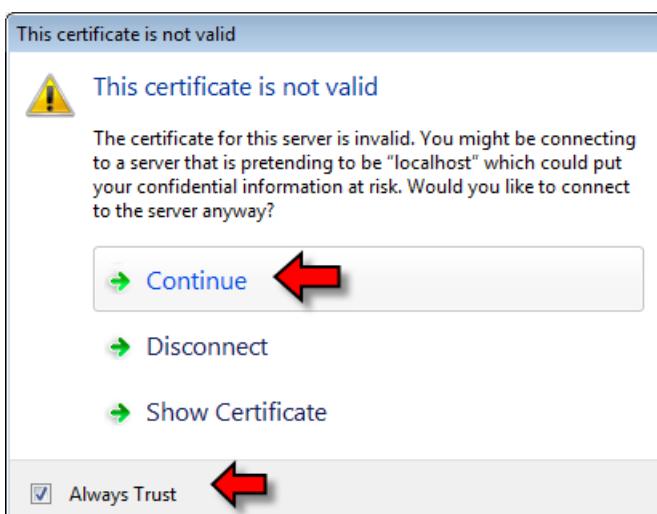


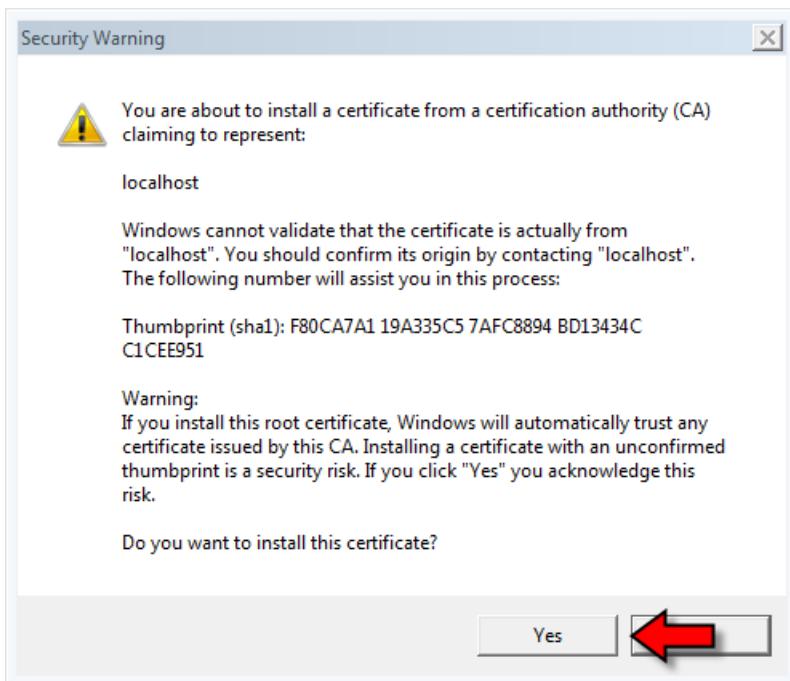
The BPM product exposes its WebDAV information via HTTP/SSL. The name of the server and SSL port number must be supplied as well as a user with sufficient authorities. In addition, a path to the WebDAV data must be supplied. This is:

```
/mum/mycontenthandler/mm/dav/filestore
```



Prompts will be issued about SSL self signed certificates. These should be accepted.





Once connected, we will be presented with a list of the folders and files managed by IBM BPM through WebDAV.

The interface has a toolbar with icons for Open Connection, Quick Connect, Action, Get Info, Refresh, Edit, Upload, and Transfers. The address bar shows the URL: /mum/mycontenthandler/mm/dav/filestore. The main area displays a file list:

Filename	Size	Modified
_users	0 B	5/29/2012 1:01:52 PM
layout-templates	0 B	5/29/2012 1:01:09 PM
public	0 B	5/29/2012 1:01:20 PM
common-resources	0 B	5/29/2012 1:00:54 PM
login	0 B	5/29/2012 1:01:16 PM

At the bottom left, it says "5 Files".

Customizing the login screen

The default Process Portal login screen looks as follows:



BPM | Process Portal

User ID

Password

Login

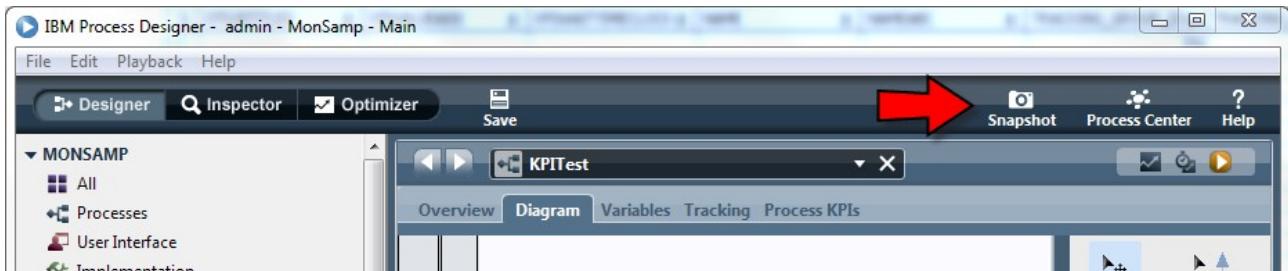
Licensed Materials - Property of IBM. © Copyright 2008, 2012 IBM Corporation. IBM, the IBM logo, and WebSphere are trademarks of IBM Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies.

Versioning Solutions

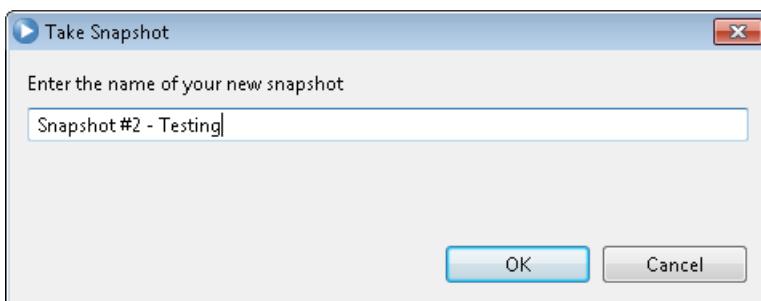
Snapshots

A snapshot is a "copy" of the state of all the artifacts in a Process Application or Toolkit at the point in time when the snapshot was made. The purpose of taking a snapshot is to allow you to revert back in time to the state of the snapshot if that is desired.

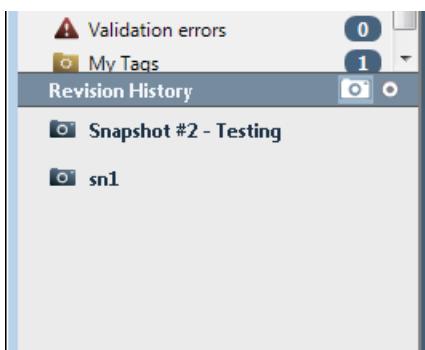
A snapshot can be captured by clicking on the Snapshot icon in IBPM PD.



When selected, a dialog appears prompting for the name of the new Snapshot. This must be unique from other snapshots.



The revision section of IBPM PD will show the list of current and past snapshots:

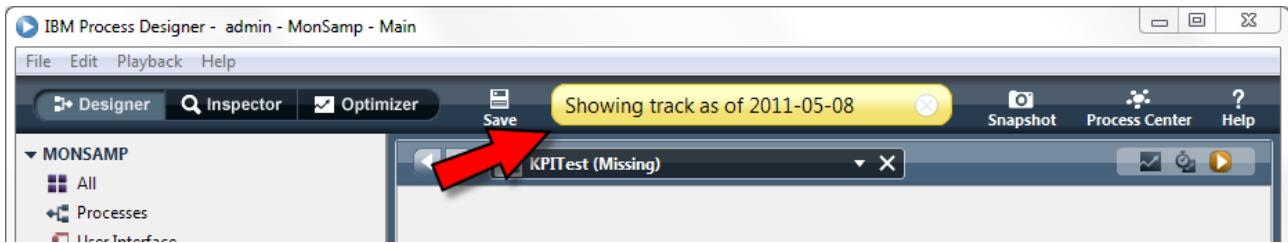


A snapshot is required in some circumstances such as:

- A snapshot of a toolkit is required before it can be added as a dependency on other toolkits or process applications.
- A snapshot of a process application is required before that application can be installed on a Process Server
- A snapshot is required before a new "workspace" can be created.

Clicking a Snapshot in the Revision History area gives that Snapshot focus allowing you to see the state of the solution at that point in time.

The fact that it is a historic snapshot we are viewing is shown in the top of the IBPM PD:



Clicking the close button on this marker will return us to the current environment.

While viewing a snapshot, you can select an older version of an asset and copy it into the latest version. Alternatively, you can revert back to an earlier snapshot, effectively taking you back in time to a previous version.

Managing Snapshots

There are a number of relatively low-level commands available that will allow you to manage snapshots. These are:

- BPMDeleteSnapshot
- BPMSnapshotCleanup
- BPMShowProcessApplication
- BPMSnapshot
- BPMDeactivate
- BPMStop
- BPMUndeploy

BPMDeleteSnapshot

This command can be used to delete a named snapshot from a Process Server. This will not work against a Process Center.

BPMSnapshotCleanup

This command can be used to delete snapshots from a Process Center.

The command has a number of options:

- containerAcronym – The acronym of the process app that contains the snapshots to be deleted.
- containerTrackAcronym
- containerSnapshotAcronyms – The acronym of a specific snapshot to be deleted.
- keptNumber
- createdBeforeLocal
- createdAfterLocal
- createdBeforeSnapshotAcronym

- deletedArchivedSnapshot
- ignoreDependency – Whether or not the snapshot can be deleted if it is currently a dependency of some other project. The default is false which means that it won't be deleted if another project depends upon it.
- outputFile

First we must connect to the server using wsadmin. For example:

```
C:\IBM\WebSphere\AppServer\profiles\Node1Profile\bin>wsadmin -conntype SOAP -port 8881 -lang jython
-user cadmin -password password
```

One particular useful application of this command is to reload a Toolkit that has become "out of sync" with the rest of the world.

Imagine we have a SN1 and then SN2 ... and now we wish to reload SN1 so that it becomes the tip. We can't do that because Process Center will tell us that SN1 still exists. To achieve our goal, we first have to tell Process Center to forget SN1. The recipe for this is as follows:

1. From Process Center, archive SN1.
2. Run the BPMShowProcessApplication to list the details of the Process App containing the snapshot to be deleted

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym XYZ]')
```

3. Examine the list to find the "Acronym" for the archived snapshot. It will have an entry that looks like:

```
Name: SN1
Acronym: XXXX
Created On: 2013-12-19 12:31:29.041
Created By: User.1002
Is Default: false
State: State[Archived]
Capability: Capability[Standard]
No of running instances: 0
```

4. Run the command to delete the snapshot

```
print AdminTask.BPMSnapshotCleanup ('[-containerAcronym XYZ -containerSnapshotAcronyms XXXX
-ignoreDependency true]')
```

The results of the command's operation will be seen in the Console Log for Process Center:

```
PALAdminComma I PALAdminCommands snapshotCleanup Entering
SnapshotDelete I [BPMSnapshotCleanup] -- There is/are 1 snapshot(s) in total that has/have been
deleted within this transaction.
[BPMSnapshotCleanup] -- The following snapshot(s) for the process application KolbanTK was/were
deleted:
[ID: Snapshot.bd2d9b64-e081-4b80-a6a6-1e9eca361379 | Name: 2013-12-19-1 | Created on: 2013-12-19
12:31:29.041]
PALAdminComma I PALAdminCommands snapshotCleanup Total amount of snapshot to be deleted: 1
PALAdminComma I Total amount of snapshot to be deleted: 1
PALAdminComma I PALAdminCommands snapshotCleanup Exiting
Restore the Toolkit from its archived state
```

BPMShowProcessApplication

This command lists the status of a Process Application. It has one mandatory parameter.

- containerAcronym – The name of the process app acronym to list

Here is an example of usage:

```
wsadmin>print AdminTask.BPMShowProcessApplication('[-containerAcronym XYZ]')
Name: XYZ
```

```

Acronym: XYZ
Description:
Toolkit: false
Tracks:

    Track Name: Main
    Track Acronym: Main
    Default: true

Tip:
    Created On: 2013-11-19 10:00:23.123
    Created By: User.1002
    State: State[Inactive]
    Capability: Capability[Standard]
    No of running instances: 0

List of Snapshots:
    Name: backup
    Acronym: B
    Created On: 2013-11-19 09:56:44.363
    Created By: User.1002
    Is Default: false
    State: State[Inactive]
    Capability: Capability[Standard]
    No of running instances: 0

    Name: 2013-11-19-1
    Acronym: 2013111
    Created On: 2013-11-19 10:00:23.123
    Created By: User.1002
    Is Default: false
    State: State[Inactive]
    Capability: Capability[Standard]
    No of running instances: 0

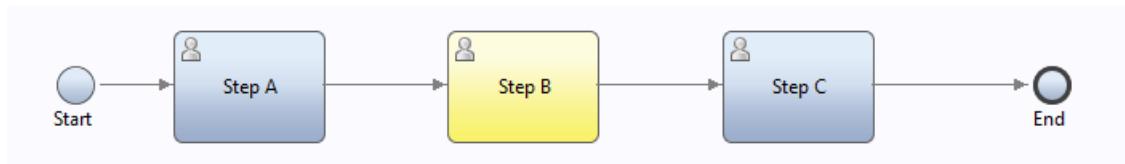
```

Migration of in-flight BPMN process instances

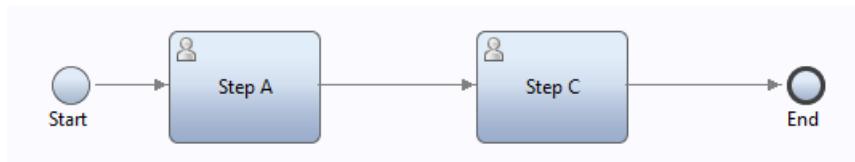
Consider the notion of versioning a process where we already have instances of that process running. What should happen to any instances that are already running? We term these as "in-flight" instances. I believe this name comes from the notion of an airline flight from one city to another. We can do maintenance on the plane at the "*from*" city or the "*to*" city but not while the plane is "*in-flight*".

First let us consider the base of the problem.

Imagine we have a process that looks as follows:



Now imagine we have a new version that now looks like:



As we can see the step that was known as "Step B" is simply no longer part of the process model. If we have instances of this process in-flight then instances which are at "Step A" will be migrated to "Step A" in the new version. The same will be true for instances at "Step C". But what about instances currently at "Step B"? What should happen to them in the new story?

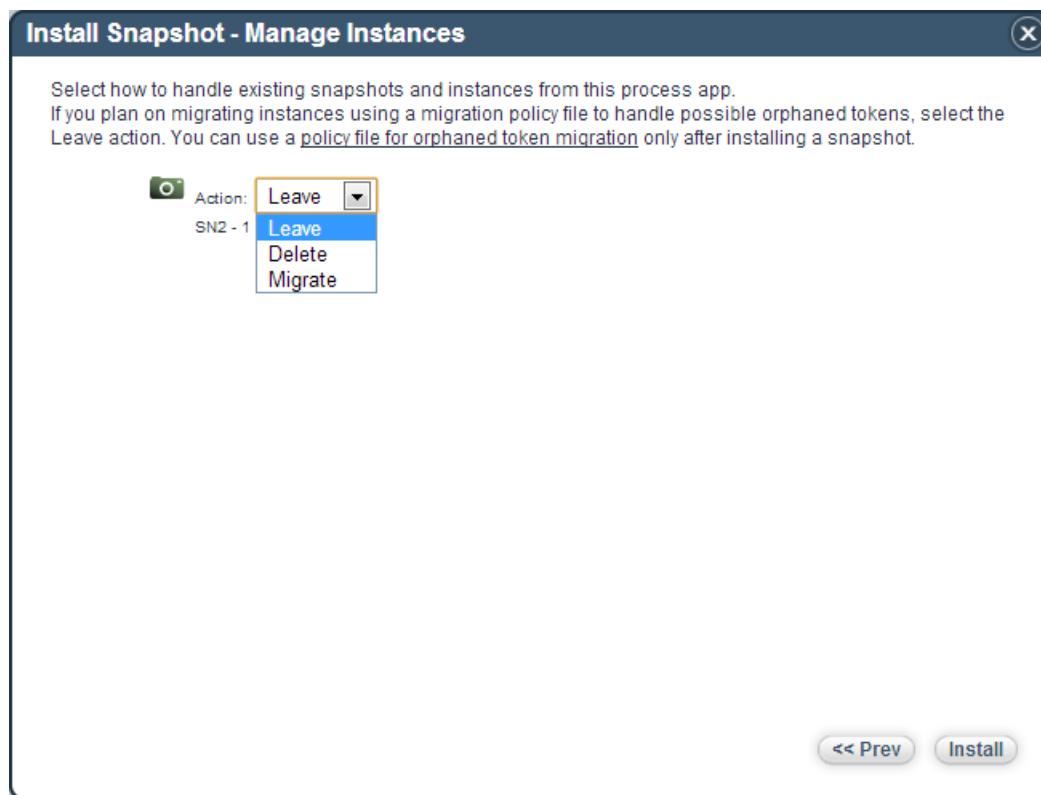
It feels like we have a number of possibilities:

- Move the token to Step A (previous step)
- Move the token to Step B (next step)
- Ask the administrator what to do and let him choose the new token step
- Terminate the process
- Force the existing processes to continue on the original process model

Let us back up a little and think about when migration of in-flight instances may be required. When a process instance is live, it is based upon a snapshot definition of a process application. If we deploy a new snapshot definition of the same process, this is where migration can take effect.

When we deploy a new snapshot version we are presented with a dialog that contains the following options.

- Leave – Instances currently running will run to completion on the original snapshot. New instances will be carved from the new snapshot.
- Migrate – Instances currently running will be migrated to the new snapshot. At the conclusion of this, there should be no remaining instances on the original snapshot.
- Delete – All instances currently running on the original snapshot will be terminated. Nothing will be migrated to the new snapshot.



Let us now look at the migration option. If we had selected "Leave" then the process instances in the old snapshot will still run against the old snapshot. If we now wish to migrate, we can select that option. Here we see a page from the Process Admin Console where we have two snapshots of the same application. SN2 has an instance of the process called BPD1 still running.

IBM | Process Admin Console Server Admin Process Inspector **Installed Apps** | Log out ?

Sort Snapshots By: Application Name ▾ All | Active | Default

<input checked="" type="radio"/> Migrate Tests (MIG1) - SN3	Active
+ BPD1' - 0 instances	
<input checked="" type="radio"/> Migrate Tests (MIG1) - SN2	Active, Default
+ BPD1' - 1 instances	
<input checked="" type="radio"/> Migrate Tests (MIG1) - SN1	
<input checked="" type="radio"/> Saved Search Admin (SSA) - 8.0	Active, Default
<input checked="" type="radio"/> Process Portal (TWP) - 8.0.0.0	Active, Default

?

Installed Apps

This page shows the process application snapshots installed on the current Process Server. Within each snapshot in the list, only the processes and services that have been exposed are shown. For each process, you can see the number of instances currently running.

[Managing installed snapshots](#)

If we click on SN3, we are presented with an option to migrate inflight data:

IBM | Process Admin Console Server Admin Process Inspector **Installed Apps** | Log out ?

Migrate Tests (MIG1) - SN3 (Active)

Deactivate Application

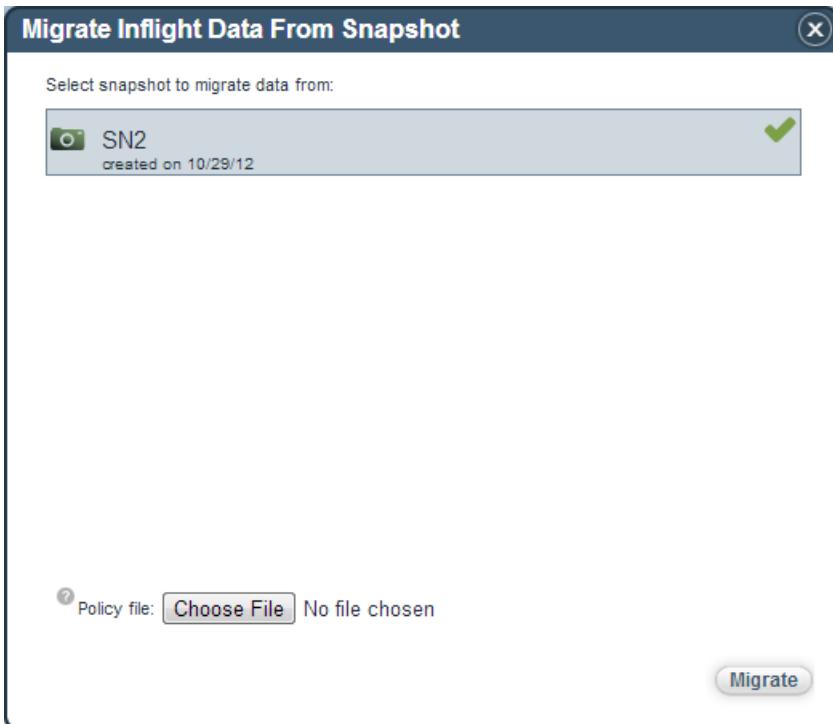
+ Migrate Inflight Data **→**

Sync Settings

Make Default Version

Update Tracking Definitions

Upon selecting that option, we are now asked which snapshot instances we wish to migrate the data from:



We also see that we are prompted to choose a "policy file". A policy file can be generated using the `wsadmin` command:

```
print AdminTask.BPMCheckOrphanTokens
```

Name	Description
<code>-processAppAcronym</code>	
<code>-sourceSnapshotName</code>	
<code>-targetSnapshotName</code>	
<code>-outputFile</code>	
<code>-overwrite</code>	

```
AdminTask.BPMCheckOrphanTokens('-processAppAcronym MIG1 -sourceSnapshotName SN2 -targetSnapshotName SN3 -outputFile c:/temp/dat.xml -overwrite')
```

The format of the policy file is:

```
<orphanTokenPolicy>
  <processApplication acronym="..." id="..." name="...">
    <sourceSnapshot acronym="..." id="..." name="..." />
    <targetSnapshot acronym="..." id="..." name="..." />
    <process bpdId="..." name="...">
      <step id="..." name="...">
        Command
      </step>
    </process>
  </processApplication>
</orphanTokenPolicy>
```

The commands can be one of "move" or "delete".

If "delete", then the token is deleted.

If "move", then the token is moved. If moved, we need to specify the id of the new target with a "targetStepId" option.

For both the "delete" and the "move" commands, an attribute called "suspendProcess" can

be supplied with a value of true or false.

In addition to the recipe described above, the REST API provides API for both moving and deleting tokens. The Methods are:

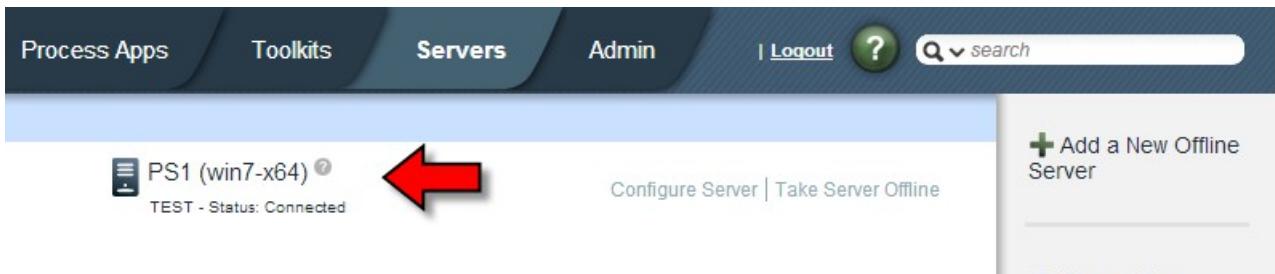
- /rest/bpm/wle/v1/process/<instance>?action=moveToken
- /rest/bpm/wle/v1/process/<instance>?action=deleteToken

See also:

- Deployment of applications to servers

Notes on testing

We need a real Process Server Process Center will not do. In my tests, the name of the Process Server was PS1:



BPD Integrations

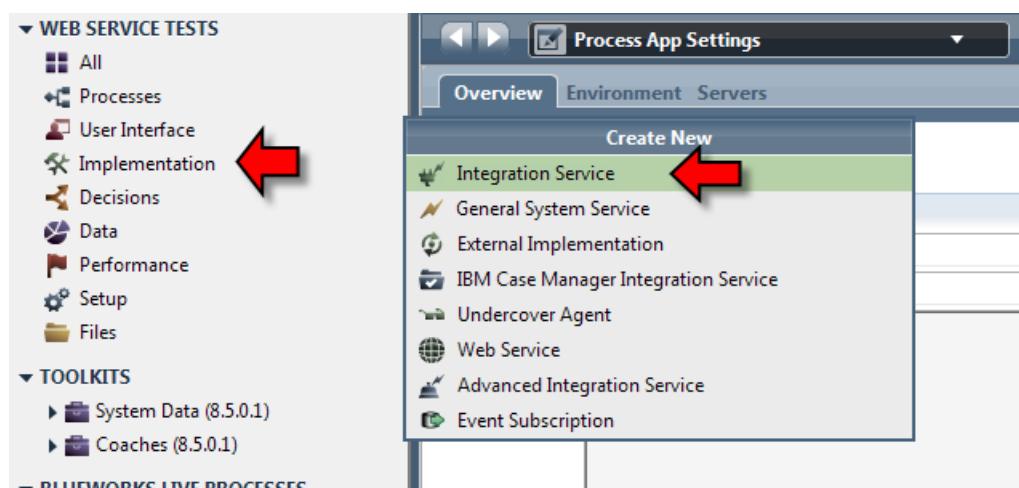
IBPM BPM has the ability to call out from a process to invoke the services of an external service provider running outside of BPM. Unless the processes being built (and even then) are solely composed of Human Services, it is extremely likely that interaction with external applications and systems will be needed. There are a number of technologies that can be used to interact with back-end systems. The following describe some of the more commonly used.

Outbound Web Services

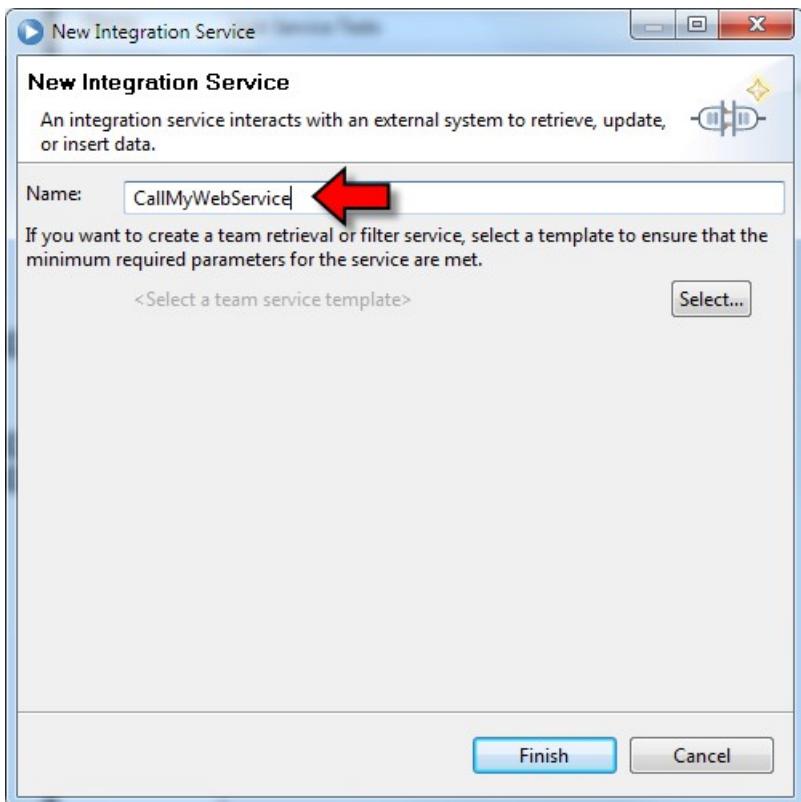
One of the more common ways that a service is exposed is through the technology known as Web Services. Web Services is an encoding of an XML document sent over an HTTP transport. Web Services is an open industry standard composed of a number of key specifications including SOAP and WSDL.

In the IBPM PD, we can create an IBPM service that encapsulates the invocation of a Web Service. When this IBPM Service is called, the result is a call to the Web Service provider that the IBPM service is configured to call.

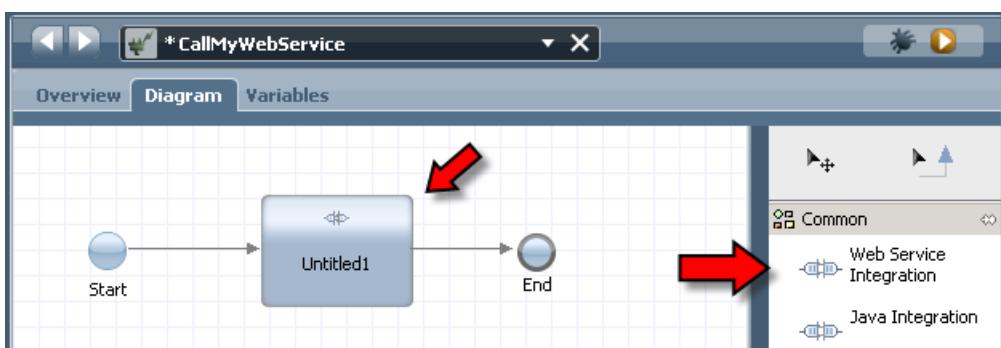
To achieve this, select the Implementation category in the Library and in the Create New menu, select Integration Service.



Once selected, a new dialog will appear asking for a name for the new integration service component.



Once completed, an empty service diagram is shown. From the palette, a Web Service Integration primitive can be dragged and dropped onto the canvas and wired into the flow. It is this component that will perform the call to the external Web Service. Notice the icon representing an external integration on the component. Learn this icon so that when you see it again you will know that this service component represents an external call.



Simply placing a Web Service component on the canvas isn't the end of the story. The component doesn't yet know which external service to call. The next step on our journey is to select the Web Service calling component on the canvas and examine its Implementation properties.

Properties Validation Errors Where Used

Step

Implementation **Discovery**
Discovery Scheme: From process application settings

Header

Data Mapping

Pre & Post

Process Application Selection
Web Service: <none>
Operations: <none>

[Use the Process Application Settings editor to add a server](#)

Generate Types...

Now we need to pause for a moment. Prior to v8.5 of IBM BPM, the mechanism for defining how we call a Web Service was performed one way but from v8.5 onwards, it is performed a very different way. Both styles are fully supported. The two styles are called "From process application settings" and "Provide in-line configuration". The selection is made from the "Discovery Scheme" selection at the top of the implementation section.

The primary difference between the two styles is that in one, the definition of the Web Service is made globally and then any integration service can call it by an alias name while in the other style, the target web service is defined there and then. The new style was introduced to allow a single place to define a Web Service that can then be leveraged throughout the solution. If the web service definition needs to be changed, it can be changed in one place rather than having to hunt through all the project artifacts.

Let us first look at the process application settings style.

From the Process App Settings of our Process Application, we can define a new server definition of type Web Service:

Process App Settings

Servers

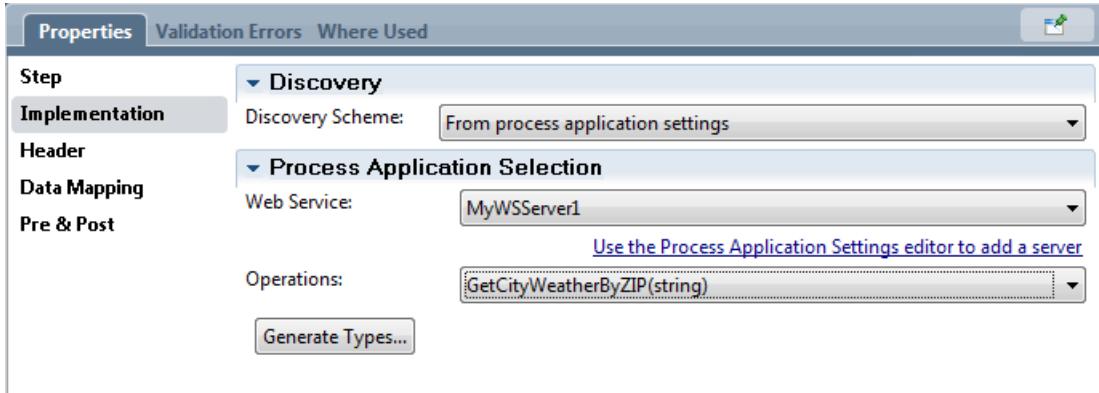
Server Details
Name: MyWSServer1
Type: Web Service
Description: Click [Edit](#) to add or edit text.

Server Locations
Environment Type: Default
WSDL URL: [Browse...](#) [View](#) [Discover](#)
Protected WSDL:
Username: _____
Password: _____
Discovery Status:
Override Endpoint:
Endpoint Address: _____
Port: _____
Security and Policy: Use Basic Security
Authentication: None
Username: _____
Password: _____
Client certificate alias: <none>
Sign request:
Expect encrypted response:
Server certificate alias: <none>
Encrypt request:
Expect signed response:

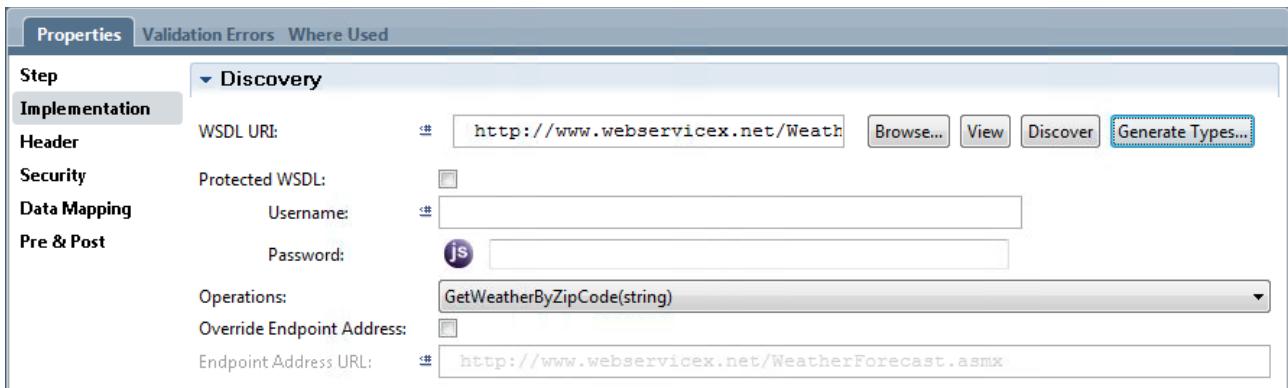
In this section, we can define an external Web Service by supplying the URL to its WSDL. This

web service then becomes a "named" entity.

In the Integration Service, when we wish to invoke the Web Service, we can now select it from a pull-down list:



Here is the inline configuration style.



The description of an external Web Service is commonly described in a Web Services Definition Language (WSDL) file. In order for IBPM to know how and where to call a Web Service, we need to make the WSDL document that describes that service known. A common convention is that when a Web Service is published a URL can be supplied which will return the WSDL for that Web Service. This is usually the URL of the web service with the suffix ?WSDL added.

For example, if a service is available at

`http://localhost:9080/MyProject/MyWebService`

then sending a request to

`http://localhost:9080/MyProject/MyWebService?WSDL`

will return the WSDL description for that service. In the WSDL URI: entry field in the editor, the URL pointer to the WSDL can be entered. Once entered, the Discover button can be pressed to dynamically retrieve the WSDL from the remote server and have IBPM parse the content to determine what operations are available and what data types are expected for those operations.

Note: If the service is provided by IBPM Advanced SCA Exports, you can not use this mechanism to properly discover the artifacts. Fortunately, there is a solution. Open a web browser and enter the URL of the service description as the page to load. You will find that the WSDL returned has

a new URL in the address bar of the browser.

For example:

Entering:

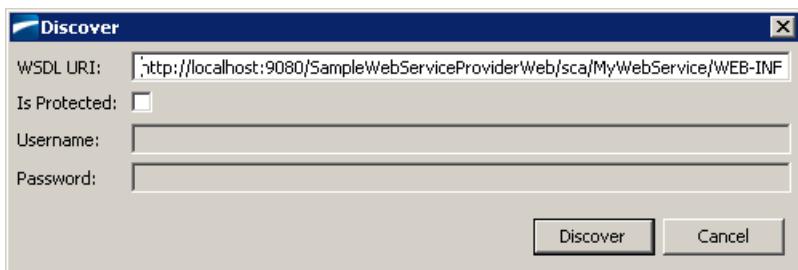
```
http://localhost:9080/SampleWebServiceProviderWeb/sca/MyWebService
```

resulted in a redirect to:

```
http://localhost:9080/SampleWebServiceProviderWeb/sca/MyWebService/WEB-INF/wsdl/SampleWebServiceProvider_MyWebService.wsdl
```

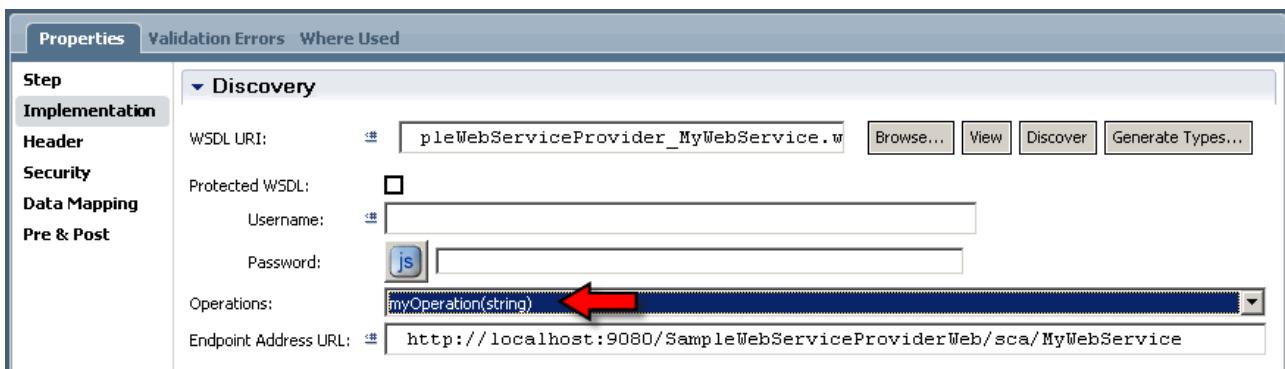
Using **this** URL will work with IBPM

When the Discover button is pressed, a new dialog is presented. This allows you to enter a username/password combination if the access to the target URL is restricted.

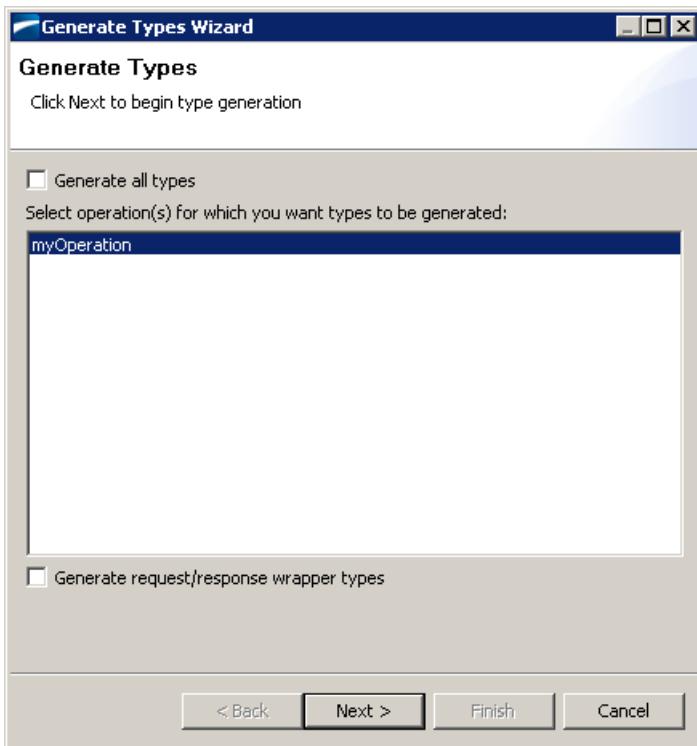


A button called View can be used to open a browser at the defined URL so that the WSDL file can be viewed in its source form.

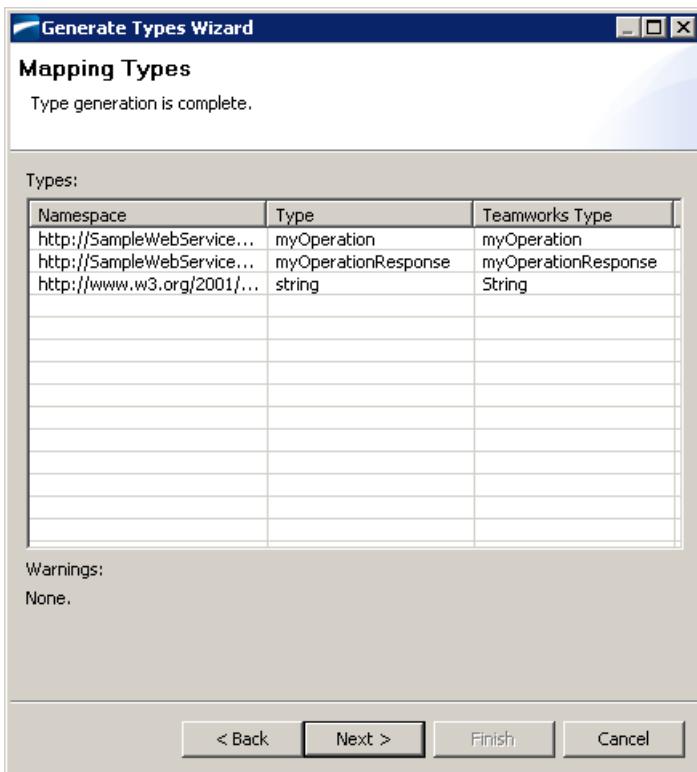
Once this step has been completed, the development tooling is now aware of all the operations that are available at the target service. From the Operations : pull-down, select the operation you wish to call. Also notice that the endpoint information for the service has now been determined.



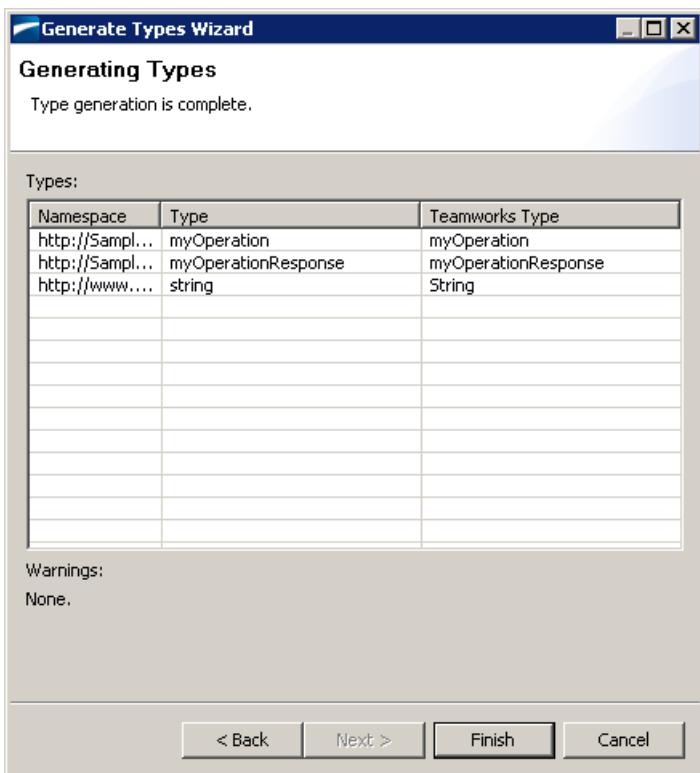
The next step is to generate the data types (if any) that are expected for the operations you wish to call. Click on the Generate Types button to create data type definitions for the data types used as inputs and outputs from the service. A dialog is presented to allow you to create some or all of the data types. Typically you would only need to select the operations that you plan to use and only data types associated with those operations are created.



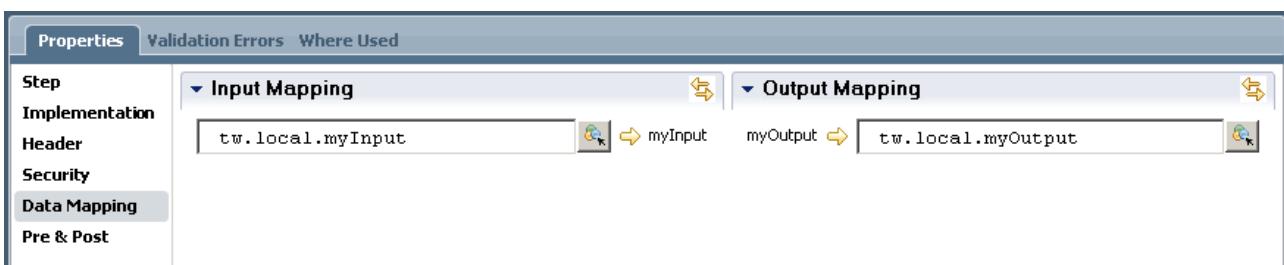
A list of the data types to be created or mapped is shown.



A second dialog is produced before completion.



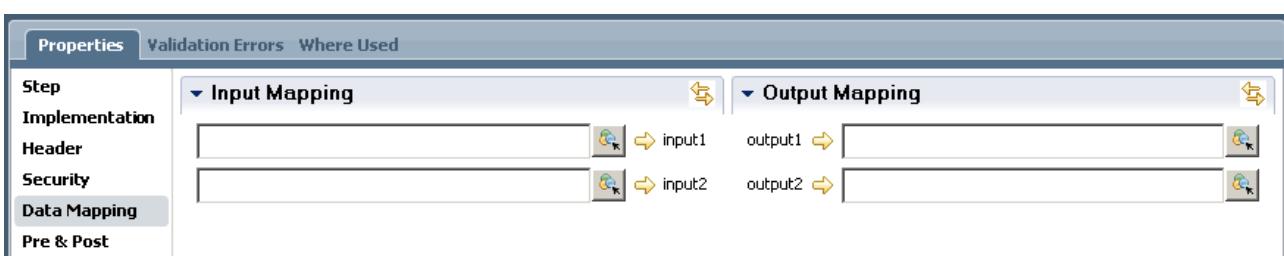
Once done, all that is left is to map variables to the data to be passed in and returned from the service call.



The names of the formal parameters are shown. If one hovers a mouse over these names, the associated data types can also be seen:

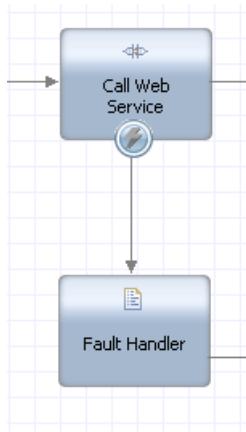


If the operation has multiple input and output parameters, each of the input and output parameters can be individually mapped:



If the service being called throws an exception, this can be caught by adding a Catch handler to the

service:



See also:

- Catch Exception

The target of the Web Service request is defined in the Endpoint Address URL box. By default, it has the value defined in the WSDL file. By checking the Override Endpoint Address check box, a new URL value can be entered and that will be the target of the request. The URL string can be retrieved from a variable's value with the <#= variable name #> technique.

A good service for testing Web Service interactions can be found [here](#). This public web service returns weather forecast data for US cities.

It is important to note that 7.5 and prior releases of the product only support SOAP 1.1 as the transport protocol. Putting this another way, SOAP 1.2 is not yet supported. If SOAP 1.2 is required, consider using the IBM BPM Advanced edition.

There are times when the Web Service target is defined over an HTTPS protocol as opposed to an HTTP protocol. This means that SSL certificate exchanges need to also occur. Fortunately, if a problem is encountered, IBM BPM generates this extremely useful guidance message:

[10/23/12 15:04:07:411 CDT] 00000007 SystemOut O CWPKI0428I:

The signer might need to be added to the local trust store. You can use the Retrieve from port option in the administrative console to retrieve the certificate and resolve the problem. If you determine that the request is trusted, complete the following steps:

1. Log into the administrative console.
2. Expand Security and click SSL certificate and key management. Under Configuration settings, click Manage endpoint security configurations.
3. Select the appropriate outbound configuration to get to the (cell):win7-x64Node01Cell:(node):win7-x64Node01 management scope.
4. Under Related Items, click Key stores and certificates and click the NodeDefaultTrustStore key store.
5. Under Additional Properties, click Signer certificates and Retrieve From Port.
6. In the Host field, enter <hostname> in the host name field, enter 443 in the Port field, and <hostname>.<domain>_cert in the Alias field.
7. Click Retrieve Signer Information.
8. Verify that the certificate information is for a certificate that you can

trust.

9. Click Apply and Save

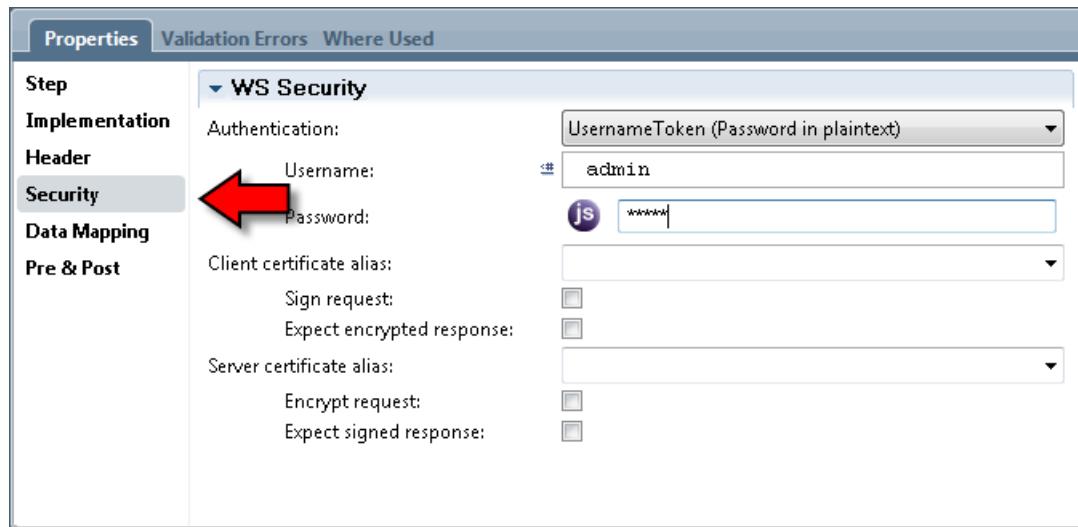
If the instructions contained within this message are followed, the problems with SSL will be resolved.

See also:

- Web Service Integration

Setting up security for outbound Web Services

It is often the case that a Web Service that is being called needs to know the identity of the caller in order to authorize the request to be performed. Settings in the Web Service integration component can be used to provide such information:



There may be a potential "bug" in this area. Setting security parameters here appeared to be ignored unless security was also set in the Implementation screen for accessing the raw wsdl.

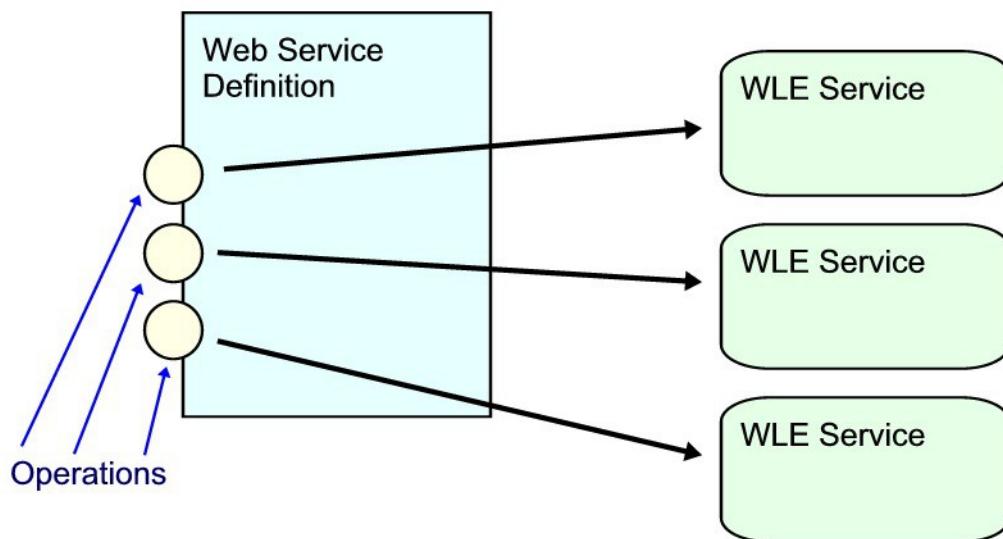
See also:

- [Enable WS-Security and Transport Layer Security in IBM Business Process Manager Standard](#) - 2013-08-28

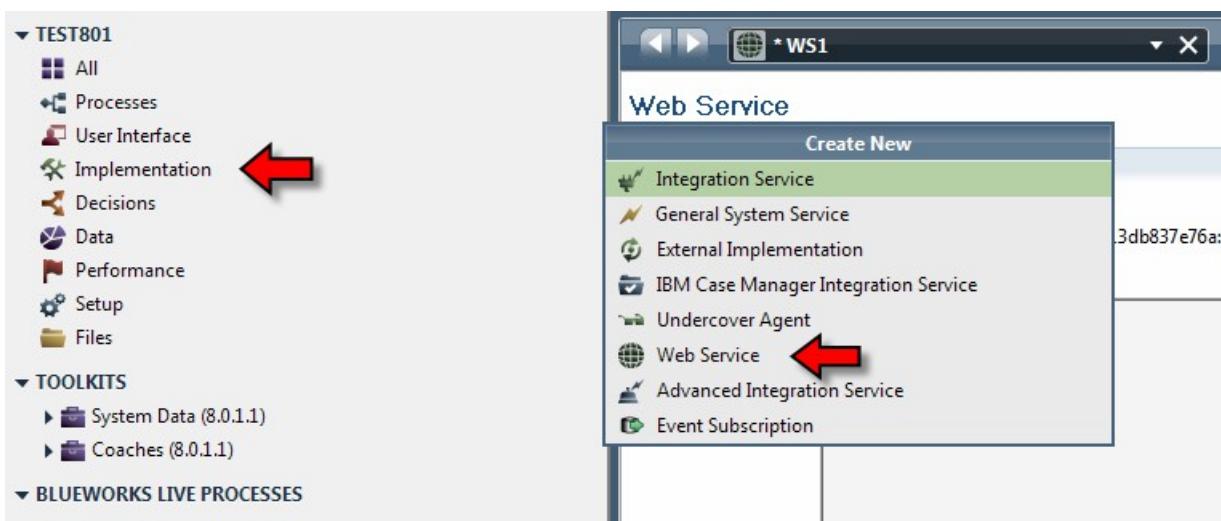
Inbound Web Services

An inbound Web Service is the notion that a IBPM solution can "advertise" or "expose" itself as a Web Service allowing it to be called from an external Web Service caller or client. In a typical Web Service, there can be multiple "operations" defined on a single service. IBPM's mapping to the concept of Web Services is that when a web service definition is made, one or more operations may be defined. For each operation defined on that Web Service, an IBPM Service definition is mapped to it.

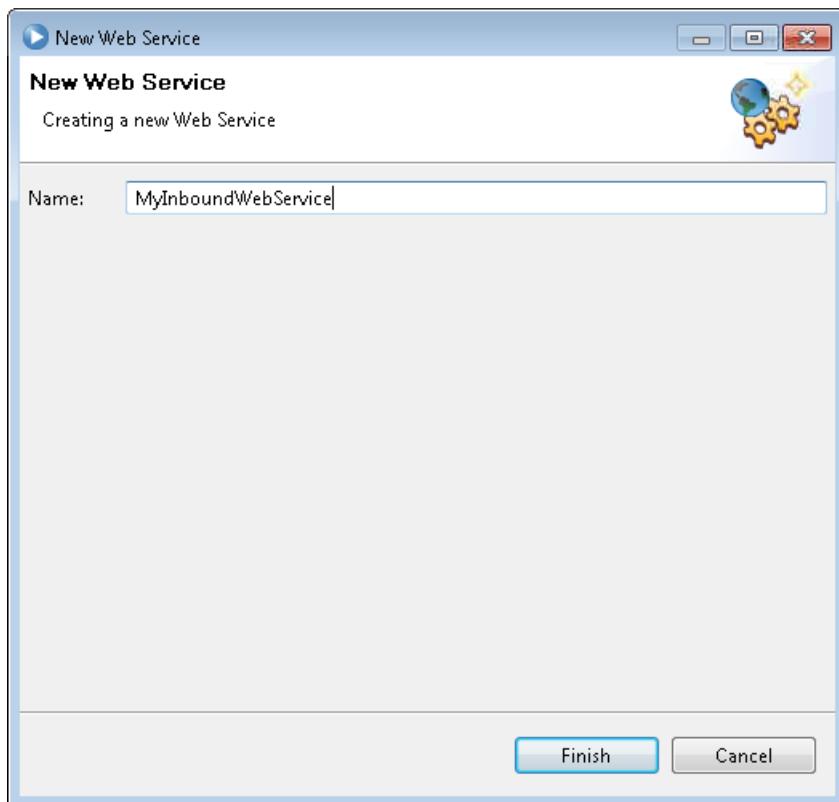
This is illustrated in the following diagram. There we see a IBPM Web Service definition with three operations defined upon it. For each operation, there is a reference to a corresponding IBPM Service definition.



To construct an inbound Web Service definition, the selection can be made from the Implementation category and selecting Web Service.



A dialog will appear allowing us to name the Web Service definition:



Once completed, the properties of the Web Service definition can be completed.

Web Service

* MyInboundWebService

Common

Name: MyInboundWebService
System ID: guid:cd2e6288625ccebfa43615481:13db837e76a:-7cd9
Modified: admin (Apr 1, 2013 7:23:35 PM)

Documentation: Click [Edit](#) to add or edit text.

Behavior

Protected:
WSDL URI: <http://localhost:90...WebService.tws?WSDL>
Target namespace scheme: Per Process App/Toolkit settings
Target namespace: <http://T801/MyInboundWebService.tws>

Operations

Add Remove

Operation Detail

Operation Name:
Attached Service: <none> Select... New...
Documentation: [View More](#)

In the Operations section, press Add for each operation that is desired to be exposed on the Web Service. For each operation added, an IBPM Service needs to be associated with it. This implies that the service must be created before adding the operation. When deployed, a call to a named operation will result in the invocation of the corresponding IBPM Service.

Any parameters defined as input on the IBPM Service will be exposed as parameters on the Web Service call. Any parameters defined as output on the IBPM Service will be returned as parameters on the Web Service call.

Take special note of the WSDL URI in the behavior section. This is the Web URL that can be used to retrieve the WSDL file that describes the exposed Web Service.

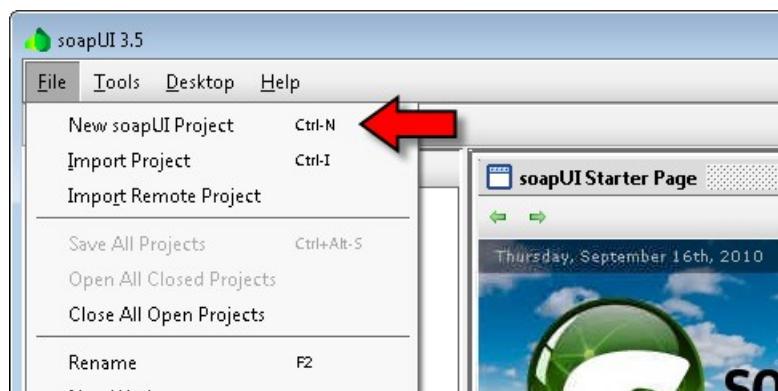
Testing an Inbound Web Service with soapUI

One of the most popular Web Service testing tools on the Internet is called soapUI. It can be found at the following web address:

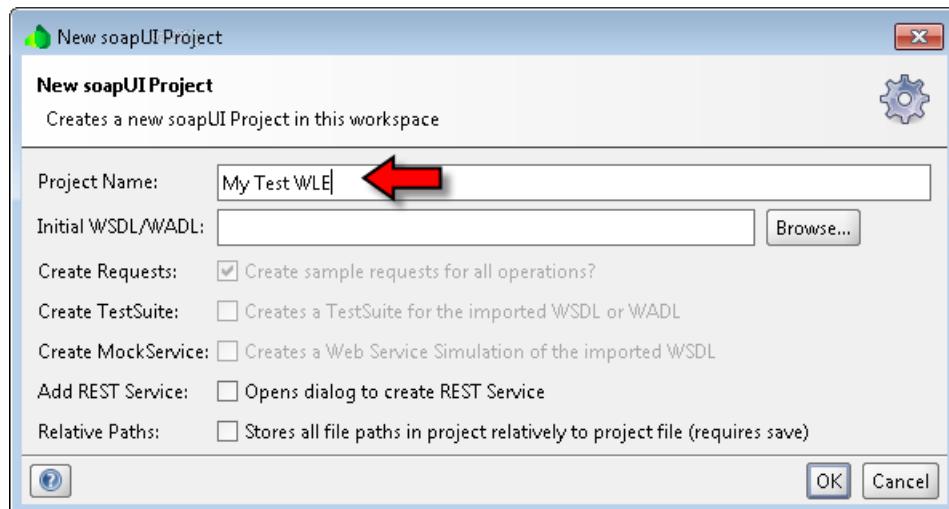
<http://www.soapui.org/>

This tool/package can be used to exercise calls to IBPM that are exposed as Web Services. The following is the recipe used to achieve this task.

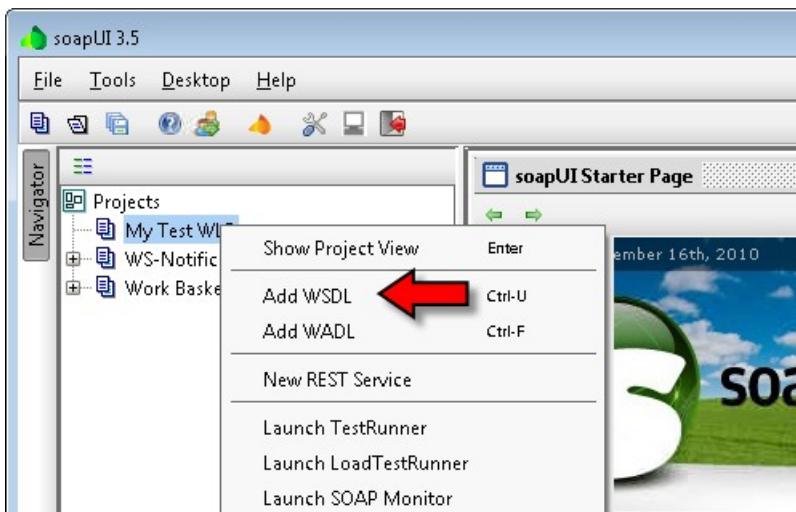
Launch the soapUI tool and select File > New soapUI Project



Give the new project a name and click OK to complete



Right click the new project and select Add WSDL



In the WSDL location text box, enter the URL for the WSDL describing the Web Service. This can be obtained from the IBPM PD Web Service definition in the "Behavior" section.

Common

- Name: MyInboundWebService
- System ID: guid:cd2e6288625ccebf:43615481:13db837e76a:-7cd9
- Modified: admin (Apr 1, 2013 7:26:08 PM)
- Documentation: [\(Edit\)](#)

Web Service Definition Usage

A Web Service Definition exposes a IBM BPM service (or a set of IBM BPM services) as a SOAP-callable Web service. This allows enterprise applications to access business process management applications developed in IBM BPM.

Behavior

- Protected:
- WSDL URI: <http://localhost:90...WebService.tws?WSDL>
- Target namespace scheme: Per Process App/Toolkit settings
- Target namespace: <http://T801/MyInboundWebService.tws>

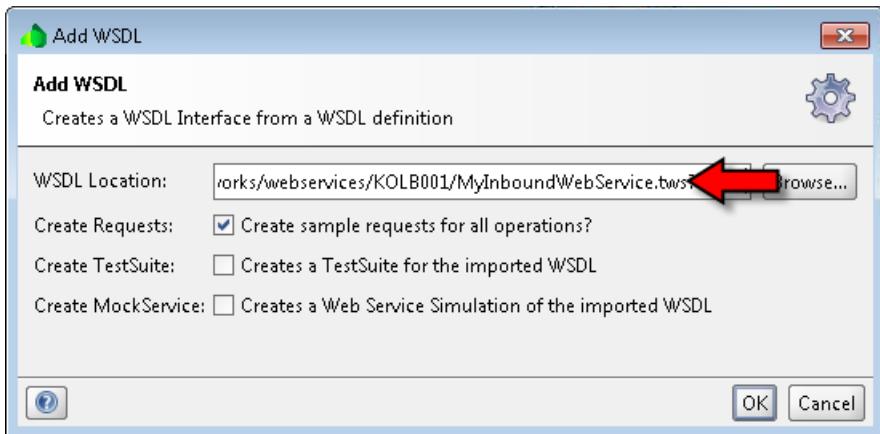
Operations

op1
Add
Remove

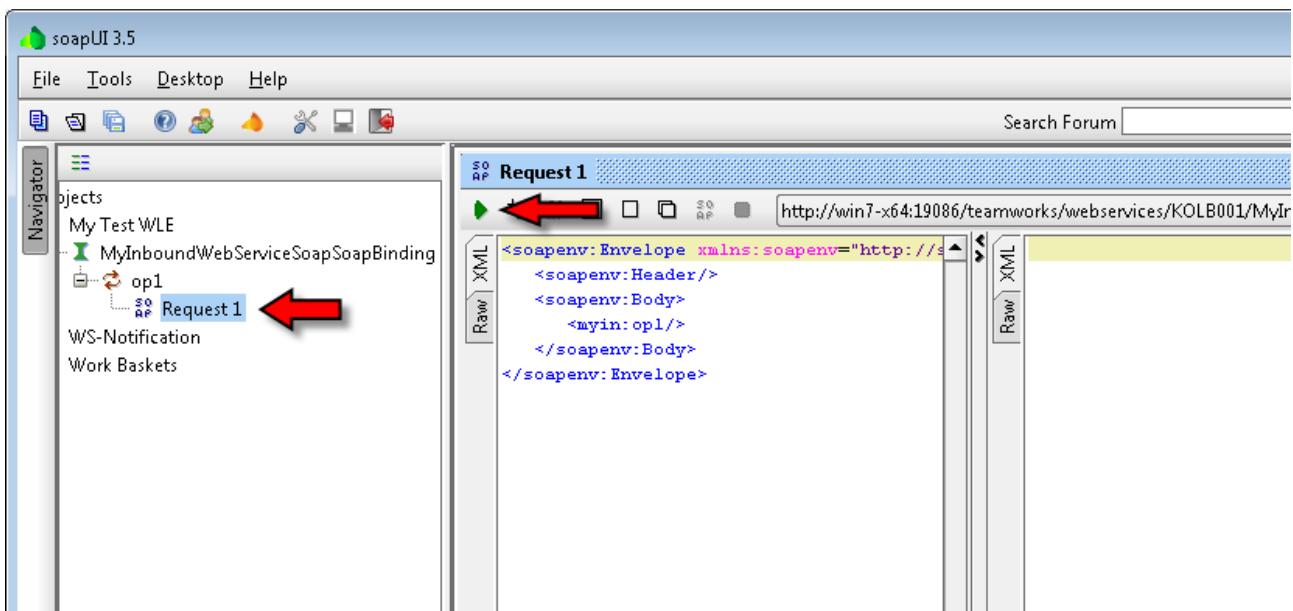
Operation Detail

- Operation Name: op1
- Attached Service: [op1Service](#) [Select...](#) [New...](#)
- Documentation: [\(Edit\)](#)

Clicking on this URL opens a Web Page to the WSDL content. The URL can then be copied from the address bar of the newly opened browser.



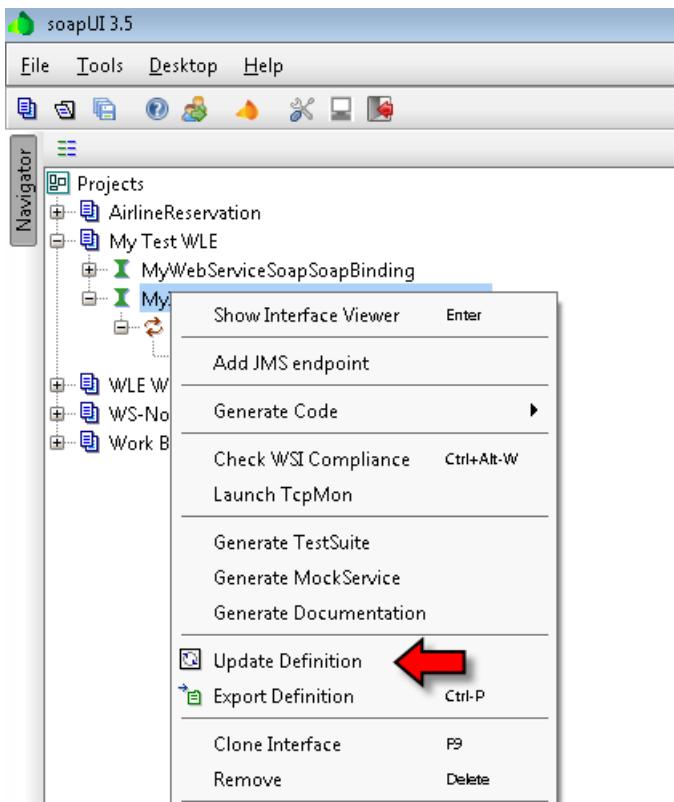
Once entered, soapUI will retrieve the WSDL description and parse it. From the WSDL, a sample request message will be built. This request message is a SOAP message that, if sent to IBPM, will cause the service to execute. Drilling down into the request will show the details of the SOAP message which can be modified.



Pressing the submit button (green arrow) will send the SOAP message to IBPM for processing. Multiple soap Request messages can be created representing different tests. The project can be saved for later replay of these requests.

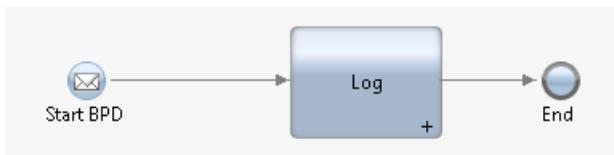
If the service associated with the Web Service definition is changed, the soapUI WSDL will have to be reloaded before testing continues otherwise an out-of-date WSDL will be used with unpredictable results.

The Update Definition menu option will re-load the WSDL:



Invoking a BPD as a Web Service

It is not uncommon to want to expose a BPD to be invoked as a Web Service. To achieve this, a few steps are involved. Working backwards from the BPD, it must first be defined as being started by Start Message Event node. Here is an example BPD:

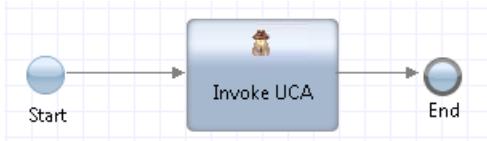


Associated with a Message Start Event node is a UCA. A UCA acts as the "trigger" to the Start Message Event. Before we can define a UCA, we need a General Service that is to be associated with the UCA. The result (output) of the General Service is the "value" passed by the UCA to its associated partner which in this case will be the Start Message Event Node in the BPD.

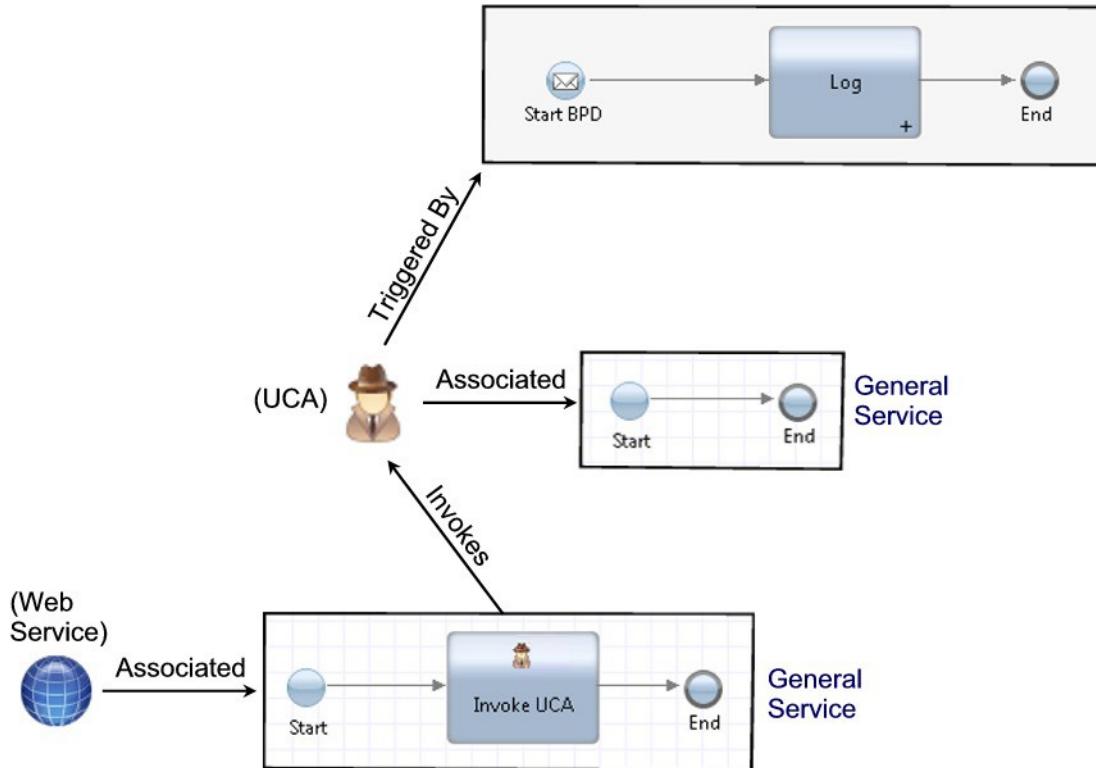
A suitable General Service may just pass the input to the output. For example:



Finally, a Web Service definition is also associated with a second General Service. It is this General Service that will be called when the Web Service is invoked. This General Service should cause the invocation of the UCA. A suitable General Service may be:



So, putting it all together we have:



This looks pretty scary so it won't do us any harm to speak about it some more. From the bottom left, we have a Web Service definition which refers to a General Service. This General Service includes an invocation of a UCA. The UCA has to have an associated General Service (because that is the rules) that knows how to build the data to be passed onwards by the UCA. The UCA is used as the trigger to the Start Message Event in the BPD.

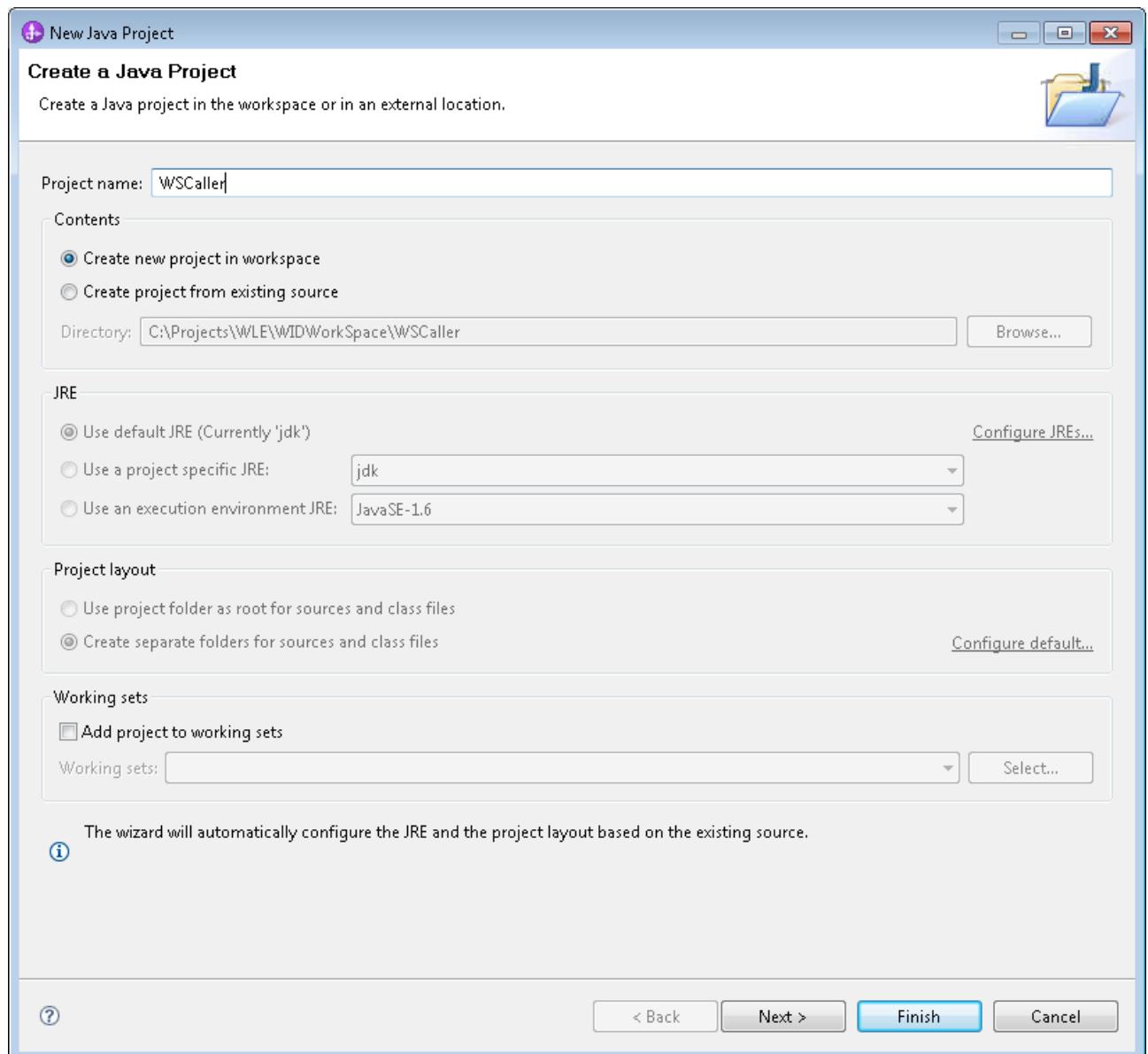
See also:

- Vimeo - [Invoking a BPD through a Web Service call](#)

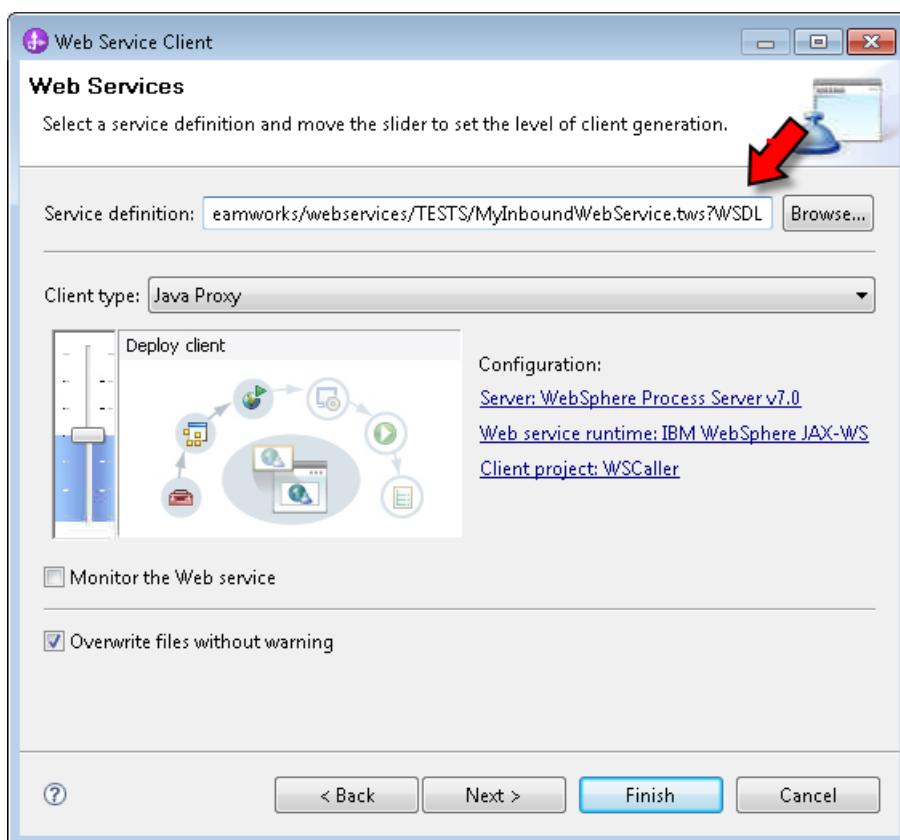
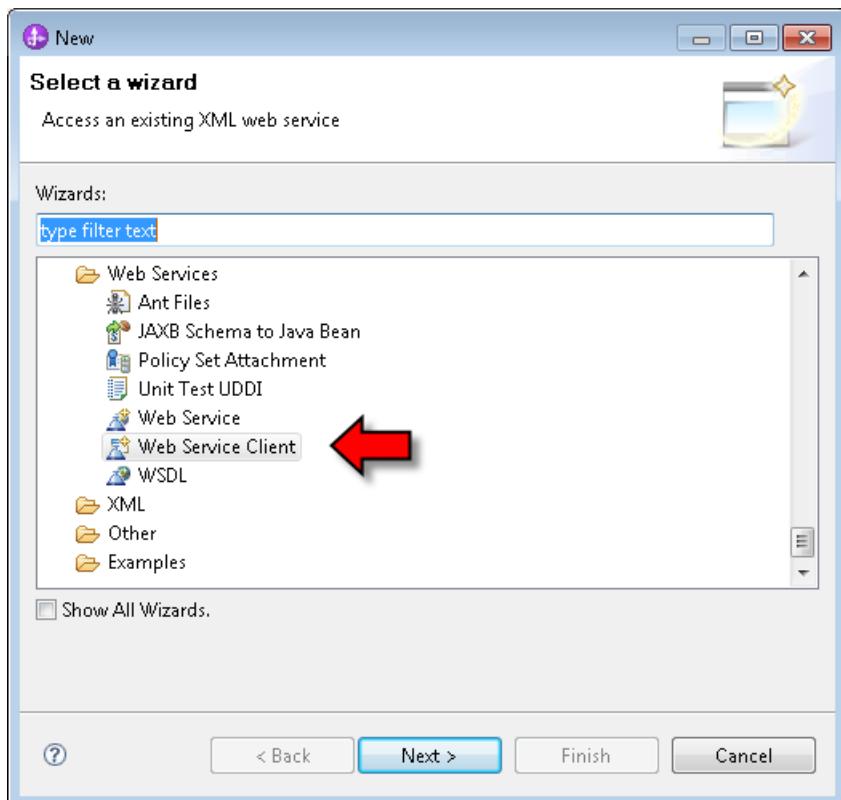
Invoking an IBPM Web Service from a Java POJO

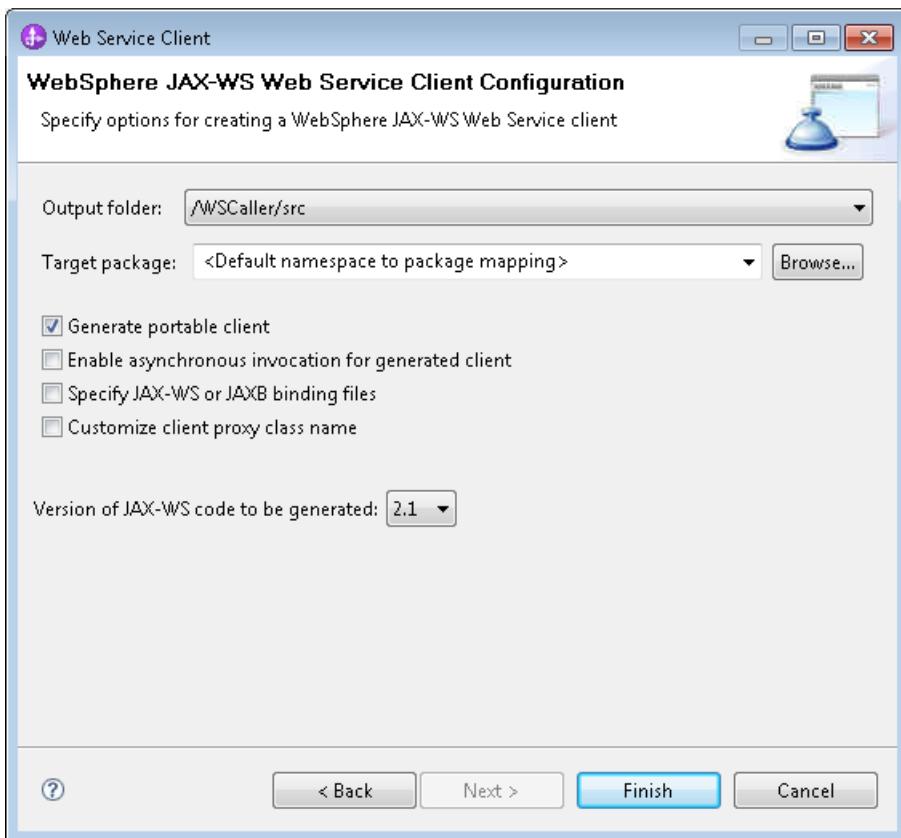
The following is a recipe for building a Java application that calls an exposed IBPM Web Service. Although not strictly related just to IBPM, it is useful to see the steps here to see how simple it is and for future reference. The sample here utilizes IBM's RAD/WID for the Java and Web Service client development.

Create a new Java Project to host the Java code that will call the Web Service.



Using the **File** > **New**, start a wizard for creating a new Web Service Client:





At the conclusion of this step, a series of Java classes will have been created conforming to the JAX-WS specification. In these classes will be one named for the service.

Create an instance of that class and then use the getter to retrieve the SOAP object. That object has methods on it corresponding to the operations on the service.

A test caller can then be written:

```
import  
win7_x64._19086.teamworks.webservices.tests.myinboundwebservice.MyInboundWebServ  
ice;  
public class Main {  
    public static void main(String[] args) {  
        MyInboundWebService miws = new MyInboundWebService();  
        miws.getMyInboundWebServiceSoap().op1("Hello");  
    }  
}
```

Web Services and data types

When we talk about passing data to and from a Web Service, we are implicitly talking about converting data types from one format to another. In IBPM, we have simple data types and complex data types. In Web Services, we have similar concepts but the data types are described by XML Schema Definitions (XSDs). The question now raised is just how compatible are these two environments? Are there data types in IBPM that don't map to XSDs and/or visa versa. In the following section we work through the different permutations of data of interest to us and execute tests against Web Services that expose those types and see how IBPM handles them.

In these tests, we have a variety of back-end Web Service that we wish to call from IBPM. Each Web Service differs from the other by the data types it returns.

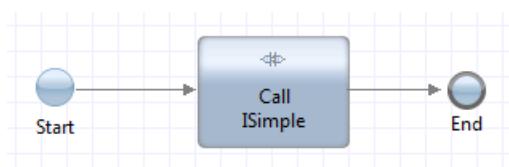
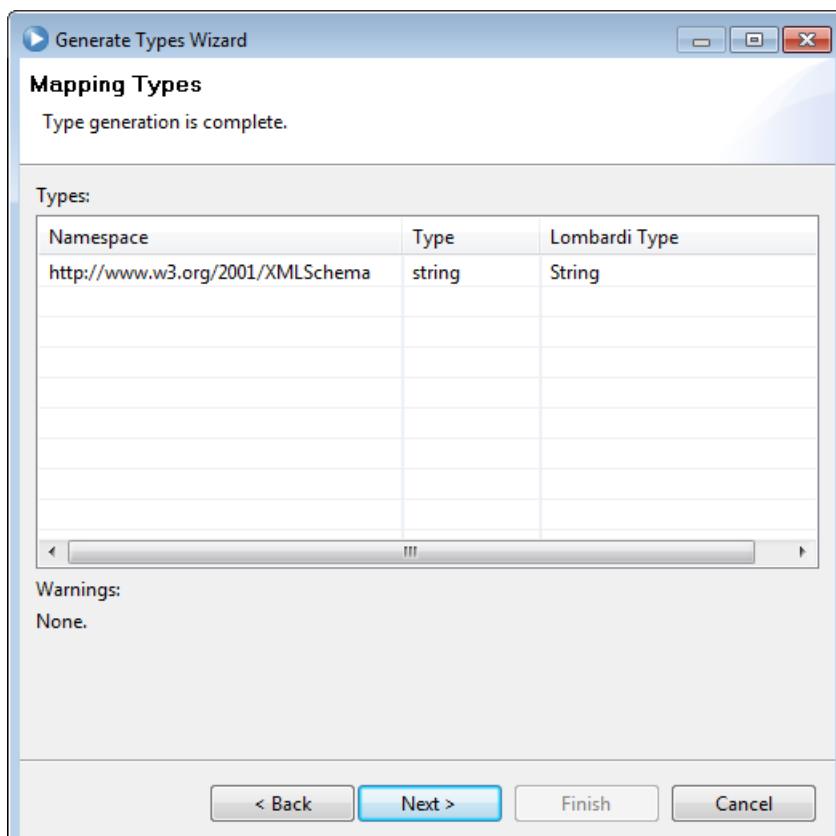
Simple Data Type

Interface: ISimple

Endpoint: http://localhost:9080/WLEDATypeTestsWeb/sca/ISimpleExport

Name	Type
opISimple	
iSimpleIn	string
iSimpleOut	string

Generated data types:



Results

Namespace: local		
Name	Type	Value
inVar	String	ISimple Input
outVar	String	ISimple Value

Conclusion

No issues at all with a simple data type. Surprised to see "String" being offered as a data type mapping as it seems to say that will create a new data type ... but it appears that the table shows mappings as well as new types to be created.

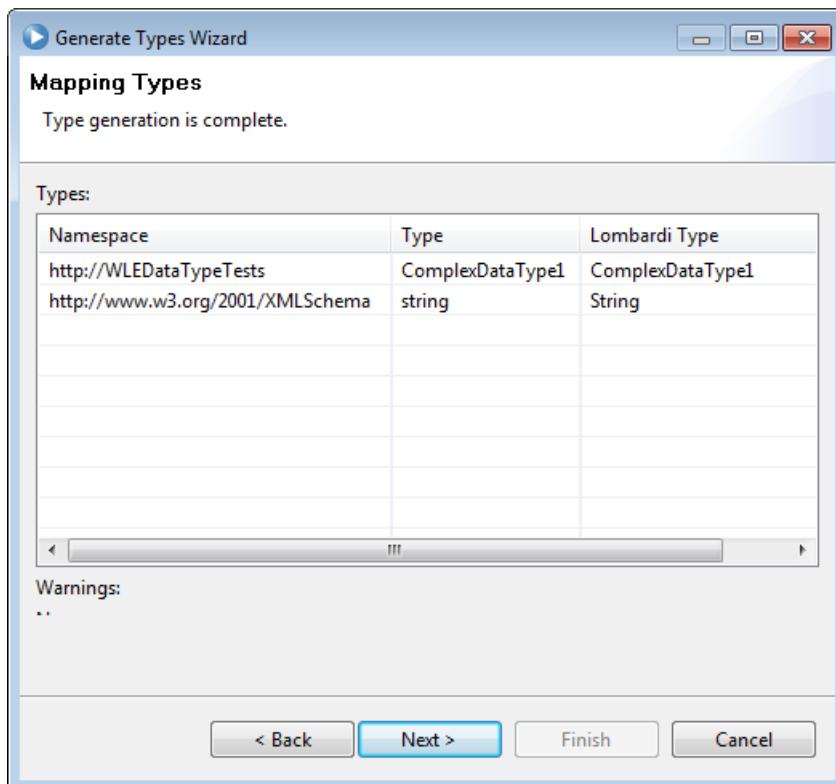
Complex Data Type

Interface: IComplex

Endpoint: http://localhost:9080/WLEDatatypeTestsWeb/sca/IComplexExport

		Name	Type
▼	opIComplex		
Inputs	iComplexIn		string
Outputs	iComplexOut		ComplexDataType1
		ComplexDataType1	
		<Click to filter...>	
		e field1 string	
		e field2 string	
		e field3 string	

Generated data types:



Results:

Namespace: local		
Name	Type	Value
inVar	String	Test for IComplex
outVar	ComplexDataType1	<object type="ComplexDataType1"> <property name="field1" type="String">IComplex Value 1</property> <property name="field2" type="String">IComplex Value 2</property> <property name="field3" type="String">IComplex Value 3</property> </object>

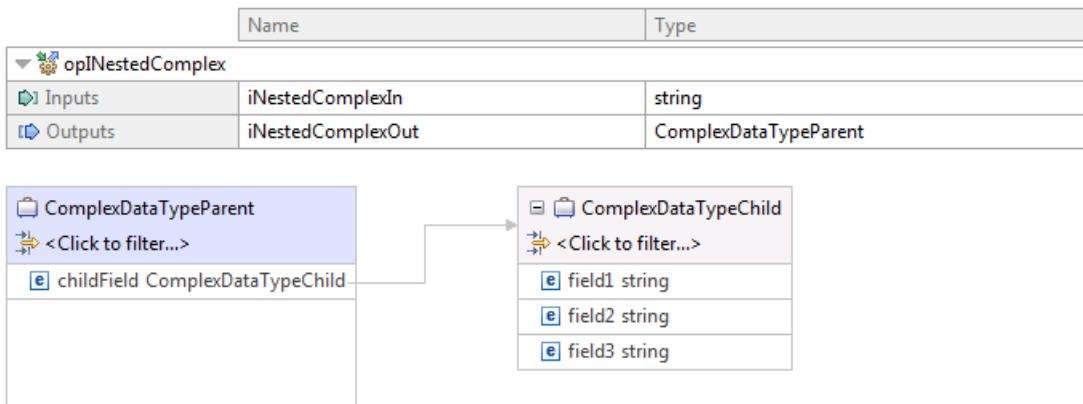
Conclusion:

No issues. Complex data type created as expected.

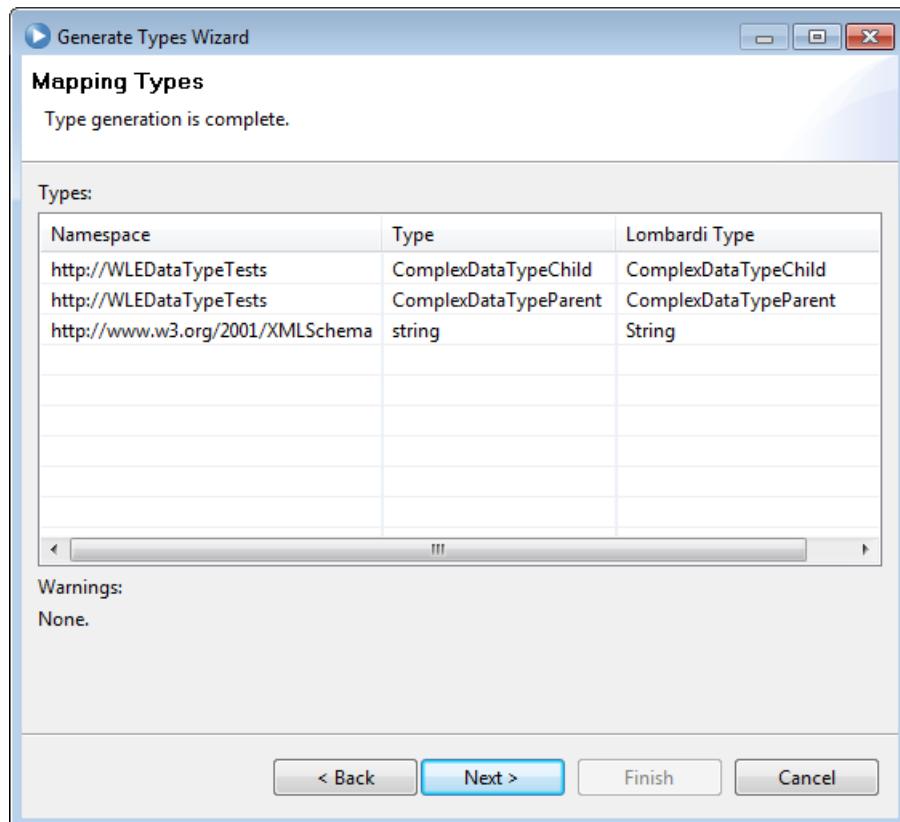
Nested Complex Data Type

Interface: INestedComplex

Endpoint: http://localhost:9080/WLEDatatypeTestsWeb/sca/INestedComplexExport



Generated data types:



Results:

Namespace: local		
Name	Type	Value
inVar	String	Hello INestedComplex
outVar	ComplexDataTypeParent	<object type="ComplexDataTypeParent"><property name="childField" type="ComplexDataTypeChild"><property name="field1" type="String">Child Field 1 Value</property><property name="field2" type="String">Child Field 2 Value</property><property name="field3" type="String">Child Field 3 Value</property></property></object>

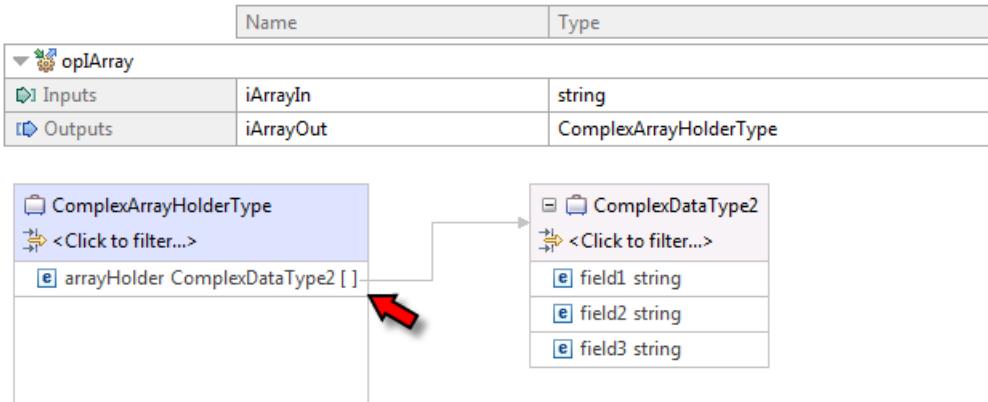
Conclusion:

No issues ... again all as expected.

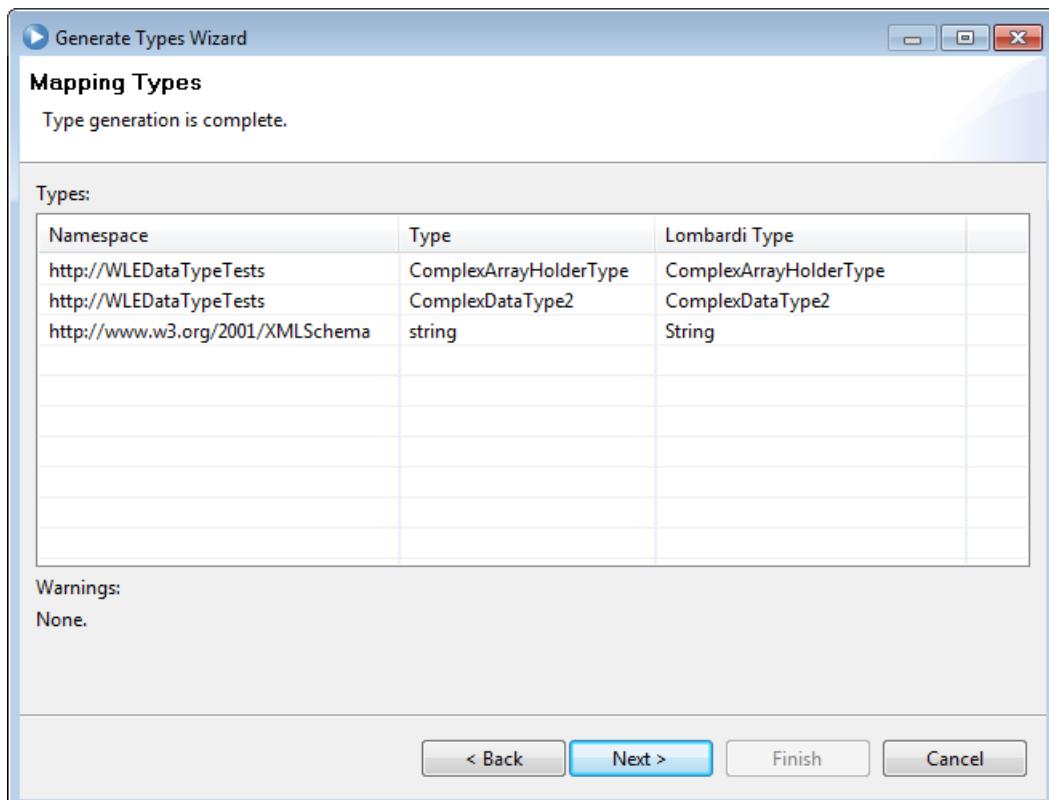
Array of Complex Data Type

Interface: IArray

Endpoint: http://localhost:9080/WLEDatatypeTestsWeb/sca/IArrayExport



Generated data types:



Results:

An error was received.

```
Runtime error in script ("Process: 'WLE IArray' ProcessItem: 'Call IArray' Type: 'ITEM'" 1:0).Internal Script error:  
com.lombardisoftware.core.TeamWorksRuntimeException: Property field1 in class  
ComplexArrayHolderType is not declared. It must be declared to be used.  
Script (line 1):
```

```
1 : tw.local.outVar = __teamworks_temp_wsconnector_variable_;
```

The full stack trace was:

```
com.lombardisoftware.core.TeamWorksException: Runtime error in script ("Process: 'WLE IArray' ProcessItem: 'Call IArray' Type: 'ITEM'" 1:0).Internal Script error: com.lombardisoftware.core.TeamWorksRuntimeException: Property field1 in class ComplexArrayHolderType is not declared. It must be declared to be used.
Script (line 1):
 1 : tw.local.outVar = __teamworks_temp_wsconnector_variable_;

      at
com.lombardisoftware.core.TeamWorksException.asTeamWorksException(TeamWorksException.java:129)
      at
com.lombardisoftware.core.RegexExceptionRewriter.rewrite(RegexExceptionRewriter.java:73)
      at
com.lombardisoftware.core.ExceptionHandler.returnProcessedException(ExceptionHandler.java:305)
      at
com.lombardisoftware.servlet.ControllerServlet.doError(ControllerServlet.java:146)
      at
com.lombardisoftware.servlet.ControllerServlet.doCommon(ControllerServlet.java:409)
      at
com.lombardisoftware.servlet.ControllerServlet.doPost(ControllerServlet.java:130)
      at javax.servlet.http.HttpServlet.service(HttpServlet.java:738)
      at javax.servlet.http.HttpServlet.service(HttpServlet.java:831)
      at
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1657)
      at
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1597)
      at
com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:131)
      at
com.lombardisoftware.servlet.ClearThreadCachesFilter.doFilter(ClearThreadCachesFilter.java:24)
      at
com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:188)
      at
com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:116)
      at
com.lombardisoftware.servlet.CrossSiteScriptingFilter.doFilter(CrossSiteScriptingFilter.java:29)
      at
com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:188)
      at
com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:116)
      at
com.lombardisoftware.servlet.SetCharacterEncodingFilter.doFilter(SetCharacterEncodingFilter.java:35)
```

```

        at
com.ibm.ws.webcontainer.filter.FilterInstanceWrapper.doFilter(FilterInstanceWrapper.java:188)
        at
com.ibm.ws.webcontainer.filter.WebAppFilterChain.doFilter(WebAppFilterChain.java:116)
        at
com.ibm.ws.webcontainer.filter.WebAppFilterChain._doFilter(WebAppFilterChain.java:77)
        at
com.ibm.ws.webcontainer.filter.WebAppFilterManager.doFilter(WebAppFilterManager.java:908)
        at
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:934)
        at
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:502)
        at
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest(ServletWrapperImpl.java:179)
        at
com.ibm.ws.webcontainer.servlet.CacheServletWrapper.handleRequest(CacheServletWrapper.java:91)
        at com.ibm.ws.webcontainer.WebContainer.handleRequest(WebContainer.java:864)
        at
com.ibm.ws.webcontainer.WSWebContainer.handleRequest(WSWebContainer.java:1583)
        at
com.ibm.ws.webcontainer.channel.WCChannelLink.ready(WCChannelLink.java:186)
        at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleDiscrimination(HttpInboundLink.java:445)
        at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.handleNewRequest(HttpInboundLink.java:504)
        at
com.ibm.ws.http.channel.inbound.impl.HttpInboundLink.processRequest(HttpInboundLink.java:301)
        at
com.ibm.ws.http.channel.inbound.impl.HttpICLReadCallback.complete(HttpICLReadCallback.java:83)
        at
com.ibm.ws.tcp.channel.impl.AioReadCompletionListener.futureCompleted(AioReadCompletionListener.java:165)
        at
com.ibm.io.async.AbstractAsyncFuture.invokeCallback(AbstractAsyncFuture.java:217)
        at
com.ibm.io.async.AsyncChannelFuture.fireCompletionActions(AsyncChannelFuture.java:161)
        at com.ibm.io.async.AsyncFuture.completed(AsyncFuture.java:138)
        at com.ibm.io.async.ResultHandler.complete(ResultHandler.java:204)
        at
com.ibm.io.async.ResultHandler.runEventProcessingLoop(ResultHandler.java:775)
        at com.ibm.io.async.ResultHandler$2.run(ResultHandler.java:905)
        at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1563)
Caused by: [TeamworksException name='Process: 'WLE IArray' ProcessItem: 'Call IArray' Type: 'ITEM'', message='Internal Script error:
com.lombardisoftware.core.TeamWorksRuntimeException: Property field1 in class ComplexArrayHolderType is not declared. It must be declared to be used.', line=1, pos=0 nested=]

```

```
    at
com.lombardisoftware.core.script.js.JavaScriptRunner.execute(JavaScriptRunner.java:283)
    at
com.lombardisoftware.core.script.js.JavaScriptRunner.evalExpression(JavaScriptRunner.java:338)
    at
com.lombardisoftware.component.common.workflow.ExecutionContext$3.call(ExecutionContext.java:563)
    at
com.lombardisoftware.component.common.workflow.ExecutionContext.doWithParams(ExecutionContext.java:620)
    at
com.lombardisoftware.component.common.workflow.ExecutionContext.evaluateJSScript(ExecutionContext.java:561)
    at
com.lombardisoftware.component.common.workflow.ExecutionContext.evaluateJSScript(ExecutionContext.java:584)
    at
com.lombardisoftware.component.common.workflow.ExecutionContext.executeJSScript(ExecutionContext.java:548)
    at
com.lombardisoftware.component.wsconnector.worker.WSConnectorWorker.bindOutput(WSConnectorWorker.java:164)
    at
com.lombardisoftware.component.wsconnector.worker.WSConnectorWorker.doJob(WSConnectorWorker.java:125)
    at
com.lombardisoftware.component.common.workflow.ExecutionJob.doJob(ExecutionJob.java:399)
    at
com.lombardisoftware.server.ejb.workflow.EJBWorkflowManagerBean.doResumeWorkflowEngine(EJBWorkflowManagerBean.java:942)
    at
com.lombardisoftware.server.ejb.workflow.EJBWorkflowManagerBean.resumeProcess(EJBWorkflowManagerBean.java:337)
    at
com.lombardisoftware.server.ejb.workflow.EJSRemoteStatefulEJBWorkflowManager_82478d70.resumeProcess(Unknown Source)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:48)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:600)
    at com.ibm.rmi.util.ProxyUtil$2.run(ProxyUtil.java:528)
    at java.security.AccessController.doPrivileged(AccessController.java:251)
    at com.ibm.rmi.util.ProxyUtil.invokeWithPrivilege(ProxyUtil.java:514)
    at com.ibm.CORBA.iiop.ClientDelegate.invoke(ClientDelegate.java:1156)
    at $Proxy93.resumeProcess(Unknown Source)
    at
com.lombardisoftware.server.ejb.workflow._EJBWorkflowManagerInterface_Stub.resumeProcess(_EJBWorkflowManagerInterface_Stub.java:518)
    at
com.lombardisoftware.component.common.workflow.EJBWorkflowManagerDelegateDefault.resumeProcess(EJBWorkflowManagerDelegateDefault.java:142)
    at
com.lombardisoftware.component.common.workflow.EJBWorkflowManagerDelegateWebSphere$6.run(EJBWorkflowManagerDelegateWebSphere.java:84)
    at
```

```

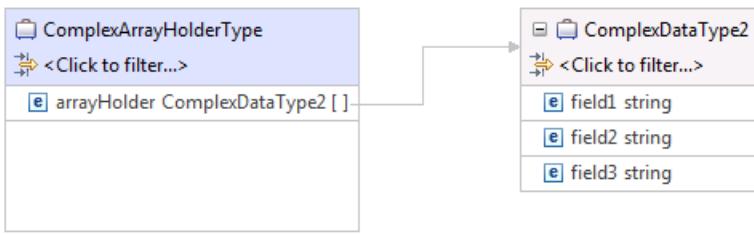
com.lombardisoftware.client.delegate.common.WebsphereDelegateHelper.doAsCurrentSubjectContextSensitive(WebsphereDelegateHelper.java:128)
    at
com.lombardisoftware.client.delegate.common.WebsphereDelegateHelper.doAsCurrentSubjectContextSensitive(WebsphereDelegateHelper.java:116)
    at
com.lombardisoftware.component.common.workflow.EJBWorkflowManagerDelegateWebSphere.resumeProcess(EJBWorkflowManagerDelegateWebSphere.java:82)
    at
com.lombardisoftware.component.common.web.WebWorkflowManager.callEJBWorkflowManager(WebWorkflowManager.java:686)
    at
com.lombardisoftware.component.common.web.WebWorkflowManager.processRequest(WebWorkflowManager.java:238)
    at
com.lombardisoftware.servlet.ControllerServlet.doCommon(ControllerServlet.java:316)
    ... 36 more
Caused by: com.lombardisoftware.core.TeamWorksRuntimeException: Property field1 in class ComplexArrayHolderType is not declared. It must be declared to be used.
    at
com.lombardisoftware.core.TWOObject.getDeclaredPropertyByName(TWOObject.java:548)
    at com.lombardisoftware.core.TWOObject.setTWClass(TWOObject.java:344)
    at com.lombardisoftware.core.TWOObject.cast(TWOObject.java:317)
    at com.lombardisoftware.core.TWOObject.setArrayItemTWClass(TWOObject.java:434)
    at com.lombardisoftware.core.TWOObject.cast(TWOObject.java:315)
    at com.lombardisoftware.server.core.SymbolTable.set(SymbolTable.java:366)
    at
com.lombardisoftware.core.script.js.AbstractTWSymbolTableScriptable.put(AbstractTWSymbolTableScriptable.java:170)
    at
org.mozilla.javascript.ScriptableObject.putProperty(ScriptableObject.java:1729)
    at
org.mozilla.javascript.ScriptRuntime.setObjectProp(ScriptRuntime.java:1557)
    at
org.mozilla.javascript.ScriptRuntime.setObjectProp(ScriptRuntime.java:1547)
    at org.mozilla.javascript.Interpreter.interpretLoop(Interpreter.java:3036)
    at org.mozilla.javascript.Interpreter.interpret(Interpreter.java:2487)
    at
org.mozilla.javascript.InterpretedFunction.call(InterpretedFunction.java:164)
    at org.mozilla.javascript.ContextFactory.doTopCall(ContextFactory.java:398)
    at org.mozilla.javascript.ScriptRuntime.doTopCall(ScriptRuntime.java:3070)
    at
org.mozilla.javascript.InterpretedFunction.exec(InterpretedFunction.java:175)
    at com.lombardisoftware.core.script.js.JSScript.exec(JSScript.java:56)
    at
com.lombardisoftware.core.script.js.JavaScriptRunner$2.execute(JavaScriptRunner.java:242)
    at
com.lombardisoftware.core.script.js.PreparedScope.executeWithScope(PreparedScope.java:199)
    at
com.lombardisoftware.core.script.js.JavaScriptRunner.execute(JavaScriptRunner.java:240)
    ... 66 more

```

Investigation of this issue turned up an interesting situation. Let us break down the scenario into more detail.

The Web Service we are calling returns a complex data type that I called "ComplexArrayHolderType". This data type contains a single field called "arrayHolder" which is

defined as a 0..* (an array/list) of data types of type ComplexDataType2. In WID as a Business Object, this looks as follows:



From a raw XSD perspective, it looks like:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://WLEDATATypeTests"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:bons0="http://WLEDATATypeTests">
  <xsd:include schemaLocation="ComplexDataType2.xsd"></xsd:include>
  <xsd:complexType name="ComplexArrayHolderType">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="arrayHolder"
        type="bons0:ComplexDataType2" maxOccurs="unbounded">
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
  
```

The XSD is consistent with the model Business Object representation model.

When a request is made to the Web Service, we see the following SOAP message being returned:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body xmlns:axis2ns8="http://WLEDATATypeTests/IArray">
    <io:opIArrayResponse xmlns:io="http://WLEDATATypeTests/IArray"
      xmlns:io2="http://www.w3.org/2003/05/soap-envelope"
      xmlns:io3="http://www.ibm.com/websphere/sibx/smo/v6.0.1"
      xmlns:io4="http://www.ibm.com/xmlns/prod/websphere/mq/sca/6.0.0"
      xmlns:io5="http://schemas.xmlsoap.org/ws/2004/08/addressing"
      xmlns:io6="http://www.ibm.com/xmlns/prod/websphere/http/sca/6.1.0"
      xmlns:io7="wsdl.http://WLEDATATypeTests/IArray"
      xmlns:io8="http://www.w3.org/2005/08/addressing"
      xmlns:io9="http://WLEDATATypeTests" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <iArrayOut>
        <arrayHolder>
          <field1>Index 1 Value 1</field1>
          <field2>Index 1 Value 2</field2>
          <field3>Index 1 Value 3</field3>
        </arrayHolder>
        <arrayHolder>
          <field1>Index 2 Value 1</field1>
          <field2>Index 2 Value 2</field2>
          <field3>Index 2 Value 3</field3>
        </arrayHolder>
        <arrayHolder>
          <field1>Index 3 Value 1</field1>
          <field2>Index 3 Value 2</field2>
          <field3>Index 3 Value 3</field3>
        </arrayHolder>
      </iArrayOut>
    </io:opIArrayResponse>
  </soapenv:Body>
</soapenv:Envelope>
  
```

```

</soapenv:Body>
</soapenv:Envelope>

```

Again, this appears consistent. The tag called <iArrayOut> is the *name* of the output variable which does indeed appear to be an instance of "ComplexArrayHolderType" as evidenced by the repeating occurrences of the tag called "arrayHolder" which is the only field in "ComplexArrayHolderType".

Looking for a IBPM perspective, we find that the generation of types from IBPM created two IBPM types. The first is called ComplexArrayHolderType and matches the XSD. This data type, as expected, has a single field called "arrayHolder" that is defined as a list of ComplexDataType2.

The screenshot shows the 'ComplexArrayHolderType' variable type configuration in the IBM BPM perspective. The window is titled 'ComplexArrayHolderType'. It contains several sections:

- Common:** Name: ComplexArrayHolderType, System ID: guid:56d35d2f221c8d0e:-79df2b84:12ebcf358ea:-7853, Modified: tw_admin (March 16, 2011 12:51:38 PM CDT), Documentation: (empty).
- Behavior:** Definition Type: Complex Structure Type.
- Parameters:** A list containing 'arrayHolder (ComplexDataType2) (List)' with three items: 'field1 (String)', 'field2 (String)', and 'field3 (String)'. There are buttons for Add, Remove, Up, and Down.
- Parameter Properties:** A table for the 'arrayHolder' parameter:

Name:	arrayHolder
Is List:	<input checked="" type="checkbox"/>
Variable Type:	ComplexDataType2
Documentation:	(empty)
- Advanced Properties** and **Advanced Parameter Properties** buttons at the bottom.

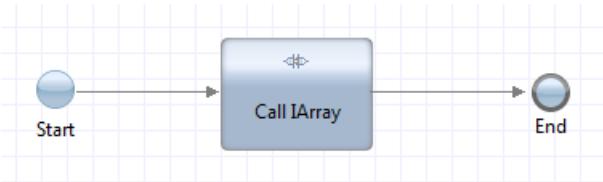
ComplexDataType2 looks as follows:

The screenshot shows the 'ComplexDataType2' variable type configuration in the IBM BPM perspective. The window is titled 'ComplexDataType2'. It contains several sections:

- Common:** Name: ComplexDataType2, System ID: guid:56d35d2f221c8d0e:-79df2b84:12ebcf358ea:-7853, Modified: tw_admin (March 16, 2011 12:51:38 PM CDT), Documentation: (empty).
- Behavior:** Definition Type: Complex Structure Type.
- Parameters:** A list containing 'field1 (String)', 'field2 (String)', and 'field3 (String)'. The 'field2 (String)' item is currently selected. There are buttons for Add, Remove, Up, and Down.
- Parameter Properties:** A table for the selected 'field2 (String)' parameter:

Name:	(empty)
Is List:	<input type="checkbox"/>
Variable Type:	<none>
Documentation:	(empty)
- Advanced Properties** and **Advanced Parameter Properties** buttons at the bottom.

In the IBPM Integration Service, the output parameter of the following integration:



In the data mappings, the following is shown:

The screenshot shows the 'Properties' tab of a BPMN element. Under 'Input Mapping', 'tw.local.inVar' is mapped to 'iArrayIn'. Under 'Output Mapping', 'iArrayOut' is mapped to 'tw.local.outVar'. The 'Data Mapping' tab is selected.

The "outVar" was defined to be of data type "ComplexArrayHolderType". Hovering the mouse over "iArrayOut" shows the expected data type:

The screenshot shows the 'Output Mapping' section. It shows 'iArrayOut' mapped to 'tw.local.outVar'. A tooltip or status bar at the bottom indicates the data type is '{http://WLEDATypeTests}ComplexArrayHolderType'.

Which again does seem to show that it is expecting a ComplexArrayHolderType. So, to recap, in this environment it fails with the previous errors.

Now ... here is where things get interesting. If I click the "autoMap" button for this variable, a variable called "iArrayOut" is created:

The screenshot shows the 'Output Mapping' section. It shows 'iArrayOut' mapped to 'tw.local.iArrayOut'. The 'autoMap' button has been clicked, creating a new local variable.

The data type of **this** variable is:

The screenshot shows the 'Variables' tab. It lists variables under 'Local' and 'Private'. A red arrow points to the 'iArrayOut (ComplexDataType2) (List)' entry, which is highlighted. This entry is described as 'a list of ComplexDataType2'.

Which in description form is "a list of ComplexDataType2".

When this variable and data type are used, the solution executes.

Namespace: local		
Name	Type	Value
iArrayOut	ComplexDataType2[]	<object type="ComplexDataType2[]"> <arrayElement size="3"> <item type="ComplexDataType2"> <property name="field1" type="String">Index 1 Value 1</property> <property name="field2" type="String">Index 1 Value 2</property> <property name="field3" type="String">Index 1 Value 3</property> </item> <item type="ComplexDataType2"> <property name="field1" type="String">Index 2 Value 1</property> <property name="field2" type="String">Index 2 Value 2</property> <property name="field3" type="String">Index 2 Value 3</property> </item> <item type="ComplexDataType2"> <property name="field1" type="String">Index 3 Value 1</property> <property name="field2" type="String">Index 3 Value 2</property> <property name="field3" type="String">Index 3 Value 3</property> </item> </arrayElement> </object>
inVar	String	Hello IArray

Conclusion:

There is a mystery at play here.

Java Message Service – JMS

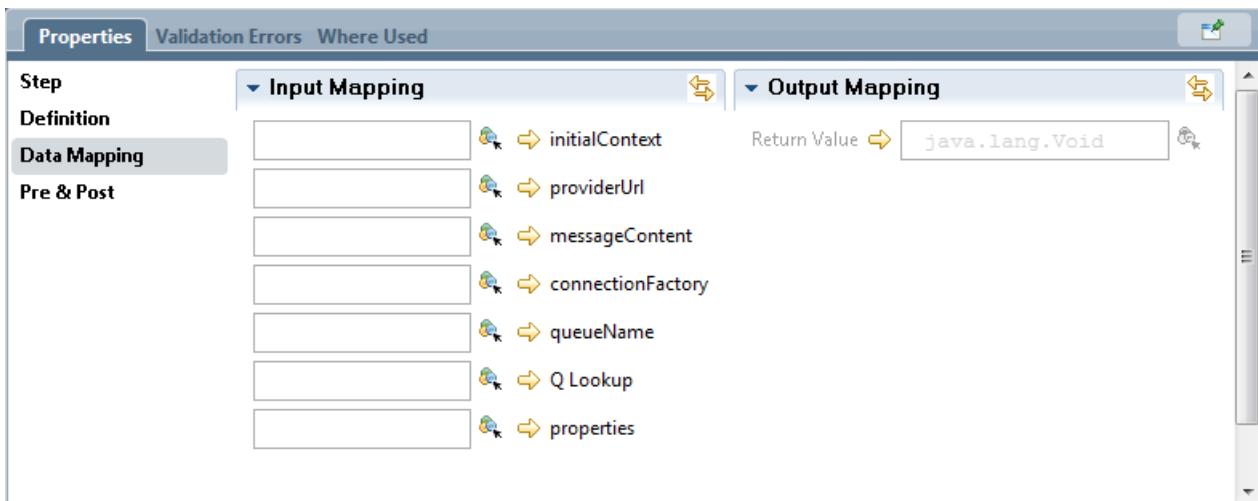
Java Message Service (JMS) is a messaging and queuing system available in the Java world. It allows applications to asynchronously put messages to a queue and for other applications to subsequently retrieve them.

JMS – Sending and receiving from queues

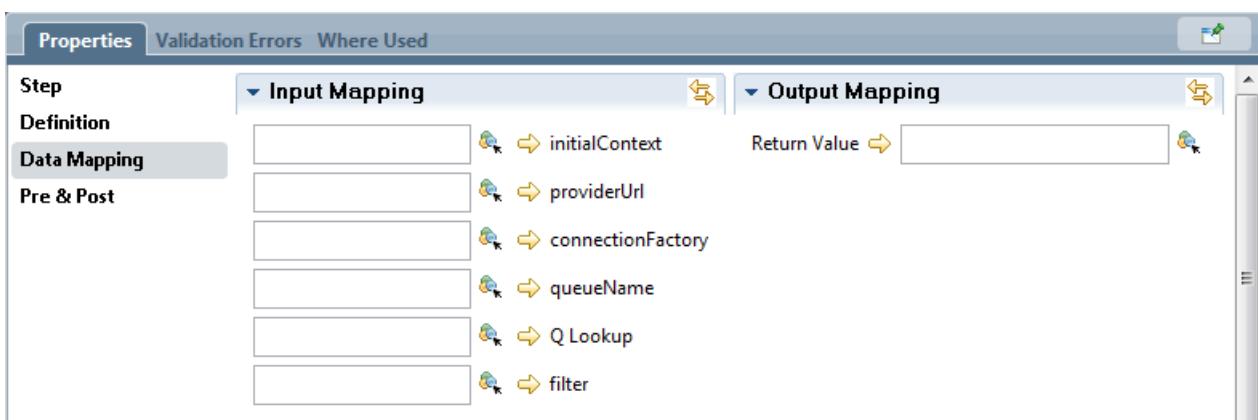
IBPM provides a Java Class as part of the `integration.jar` managed file called `teamworks.JMSMessage`. This class has methods to put and get messages from a JMS queue. To use this capability, create an Integration Service and insert a Java Integration step. For the implementation of that step, select `teamworks.JMSMessage` from the `integration.jar`. The methods available are:

- `putMessage` – Puts/sends a message to a queue
- `getMessage` – Get/retrieve a message from a queue

The `putMessage` parameters look like:



The getMessage parameters look like:



Now we can start to expand on these parameters.

Parameter	Description
initialContext	This is the name of a Java class that provides the JNDI lookup context. This will be the string: "com.ibm.websphere.naming.WsnInitialContextFactory"
providerUrl	This is the URL to the JMS provider. For IBM's SIBus, this will be: "iiop://<hostname>:<port>". The port number will be the bootstrap port of the WAS server. This can be found in the WAS admin console under the server definition. The default port value for a normal configuration is 2809.

Communications		
	Ports	Details
Port Name	Port	
BOOTSTRAP_ADDRESS	2809	
SOAP_CONNECTOR_ADDRESS	8880	
ORB_LISTENER_ADDRESS	9100	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9401	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9403	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9402	
WC_adminhost	9060	
WC_defaulthost	9080	
DCS_UNICAST_ADDRESS	9353	
WC_adminhost_secure	9043	
WC_defaulthost_secure	9443	
SIP_DEFAULTHOST	5060	
SIP_DEFAULTHOST_SECURE	5061	
SIB_ENDPOINT_ADDRESS	7276	
SIB_ENDPOINT_SECURE_ADDRESS	7286	
SIB_MQ_ENDPOINT_ADDRESS	5558	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5578	
IPC_CONNECTOR_ADDRESS	9633	

messageContent	This is a text string that is sent as the body of the message. Note that only JMS TextMessages can be sent using the putMessage() method. Specifically, the other types of JMS messages such as Stream, Bytes, Map and Object are not available.
connectionFactory	This is the JNDI name of the JMS queue connection factory.
queueName	This is the JNDI name of the JMS queue to be accessed
Q Lookup	This is a boolean value (true/false). If set to true, the queue named in the queueName parameter is looked up in JNDI. If set to false, an attempt is made to create a queue with the given name.
properties	This is a IBPM Map data type that contains a set of name/value properties. These properties will be added to the transmitted TextMessage.
filter	

See also:

- DeveloperWorks - [WebSphere Lombardi Edition V7.1: Integrating with the JMS queue, Part 1: JMS integration with the queue hosted on a local Service Integration Bus](#) – 2010-12-08
- DeveloperWorks - [WebSphere Lombardi Edition V7.1: Integrating with the JMS queue, Part 2: JMS integration with the queue hosted on a remote Service Integration Bus](#) - 2010-12-09

JMS – Triggering a UCA

IBPM can listen on JMS queues that are owned by WAS. When a message arrives on this queue, IBPM will trigger a UCA as a result. This provides an elegant way for a IBPM based solution to be triggered by the arrival of an asynchronous message.

The primary queue that is being watched by IBPM for events is retrievable at the JNDI location called:

jms/eventqueue

The format of the message that arrives on the queue is prescribed by IBPM architecture and **must** look as follow:

```
<eventmsg>
<!-- The process app acronym and event name are required.
```

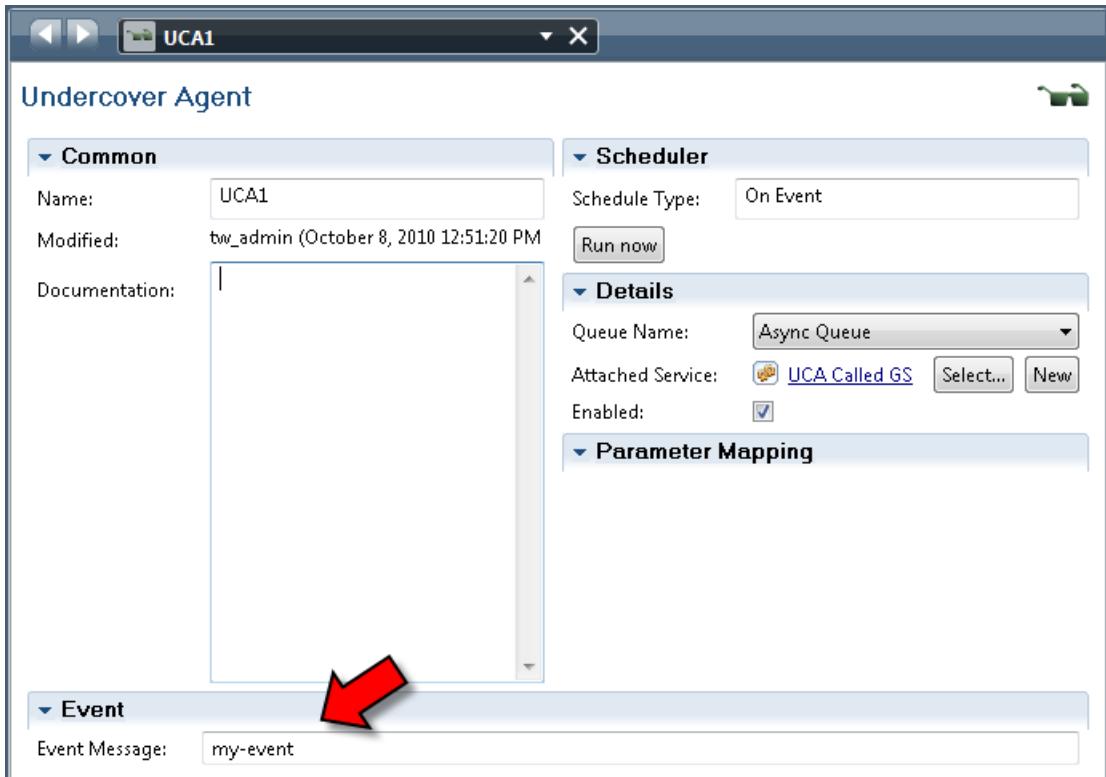
```

The snapshot and UCA name are optional -->
<event
  processApp="acronym"
  snapshot="[snapshot_name]"
  ucaname="[UCA_name]">event_name</event>
<!--Optional: The name of the queue for the event to run in--&gt;
&lt;queue&gt;[queue name]&lt;/queue&gt;
<!--Any parameters the UCA may require-
&lt;parameters&gt;
  &lt;parameter&gt;
    &lt;key&gt;param1&lt;/key&gt;
    &lt;value&gt;! [CDATA[value1]]&lt;/value&gt;
  &lt;/parameter&gt;
&lt;/parameters&gt;
&lt;/eventmsg&gt;</pre>

```

The `processApp` attribute on the `event` element provides the name of the acronym or "short name" of the Process Application. This is required as all artifacts are scoped by the Process App or Toolkit in which they live. Since the message we are going to put in a queue is destined for a UCA which is associated with a Process Application, we need to name the Process Application in which it lives.

When we define a UCA, we give that UCA an attribute called the "Event Message".



This is a poor choice of name for this attribute. What this attribute **does** is correlate an external event such as a message arriving on a JMS queue with the type of UCA to invoke. A Process Application may have multiple UCAs associated with it yet IBPM listens on just one queue. This means that when a message is deposited in the queue, there has to be data in the message to allow IBPM to know which UCA to deliver the message to. The Event Message attribute is the key. By default, IBPM assigns this a uniquely generated value but it is common to change this to a human readable value that describes its nature with more meaning.

The "event_name" element in the JMS XML message contains the name of the Event Message that is used for matching.

Since an incoming event may have data associated with it and this data matches data with the UCA, the Parameters section may have multiple parameter values that are mapped to the message.

See also:

- Undercover Agents (UCAs)
- Starting a UCA from REST

JMS Client Tools

Note: 2012-01-26 - Bizarrely, both of these tools are no longer available from the IBM AlphaWorks web site. Checking with the IBM DeveloperWorks staff, it was found that the reason they are no longer present is that the original IBM authors chose not to continue to support/distribute these tools. Although the tools may have been "dated", they worked beautifully. It is lamentable that they are no longer present. This section will be left in case they should be resurrected.

IBM has a sample JMS client application called the "IBM Client Application Tool for JMS" (what else would it be called). This can be found here:

<http://www.alphaworks.ibm.com/tech/jmstool>

Another useful tool is called Service Integration Bus Explorer. It can be found here:

<http://www.alphaworks.ibm.com/tech/sibexplorer>

For sending messages, I prefer the JMSTool. Here is a quick-start to get it working with IBPM.

First download the tool in a file called `JMS_Client.zip` as well as the SWT libraries as documented at the JMSTool download page. Ensure that the SWT libraries that you download match the platform on which the tool is to be run. For example, download the 64bit libraries if you are going to be running on a 64bit OS. Also make sure that there are no spaces in the paths for either JMS Client, SWT or the WAS App Server.

Extract both the `JMSTool.zip` and the SWT libraries ZIP.

In the `JMS_Client` folder created, find the file called `setenv.bat`. Edit this file with a text editor.

Change the variables for `WAS_HOME` and `SWTJARS` and `JNDI_PROV_URL`

An example resulting file looks like:

```
rem ######
rem #
rem # IBM Client Application for JMS environment script
rem # Copyright (C) IBM Corp. 2006
rem #
rem # This script is run automatically when starting the IBM Client Application
rem # for JMS using the 'JMSSClient.bat' script. The environment variables in this
rem # script should be customized to your system before running the
rem # 'JMSSClient.bat' script.
Rem #
rem #####
rem #####
rem #####
rem # This value should be the location of your WebSphere installation
```

```

rem # (E.g. C:\Program Files\IBM\WebSphere\AppServer)
rem #####set WAS_HOME=C:\IBM\Lombardi7\AppServer
set WMQ_JAR_PATH=%WAS_HOME%\lib\WMQ\java\lib

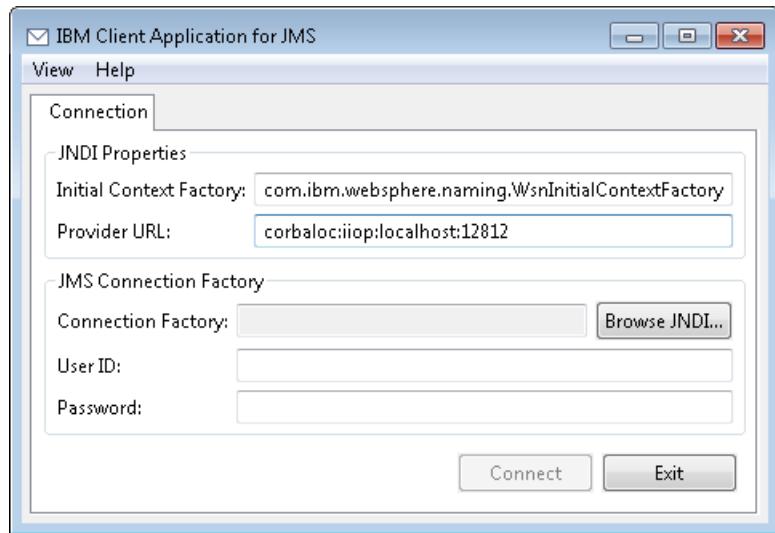
rem #####rem # If you are using the SWT GUI, you need to make sure that the SWT libraries
rem # are available on the classpath. The value of the following environment
rem # should be set to the location of the SWT libraries (I.e. the location of
rem # your 'swt.jar' file and SWT DLL's)
rem #####
set SWTJARS=C:\Projects\WLE\JMS\swt-3.6.1-win32-win32-x86_64

rem #####rem # The JMS connection factory for the default messaging JMS provider has a
rem # dependency on JNDI, performing a lookup for certain WebSphere resources
rem # when attempting to connect to a messaging engine within a Service
rem # Integration Bus. For this reason, a value must be specified for the
rem # JNDI_PROV_URL environment variable that points at a valid Naming Service
rem # within the target WebSphere Application Server cell if you are attempting
rem # to connect to a Service Integration Bus.
Rem #####
set JNDI_PROV_URL=corbaloc:iiop:localhost:12812

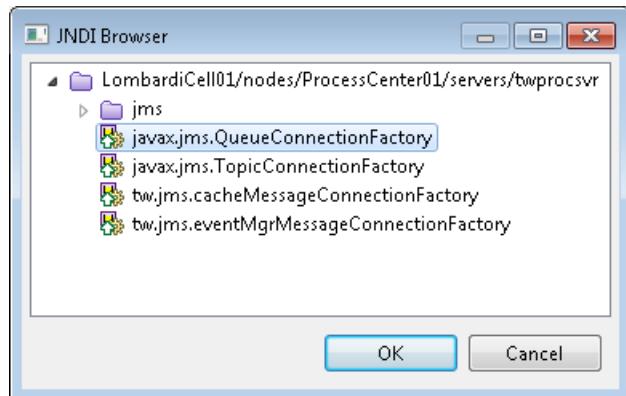
```

To launch the tool, run:

JMSClient.bat -ui SWT

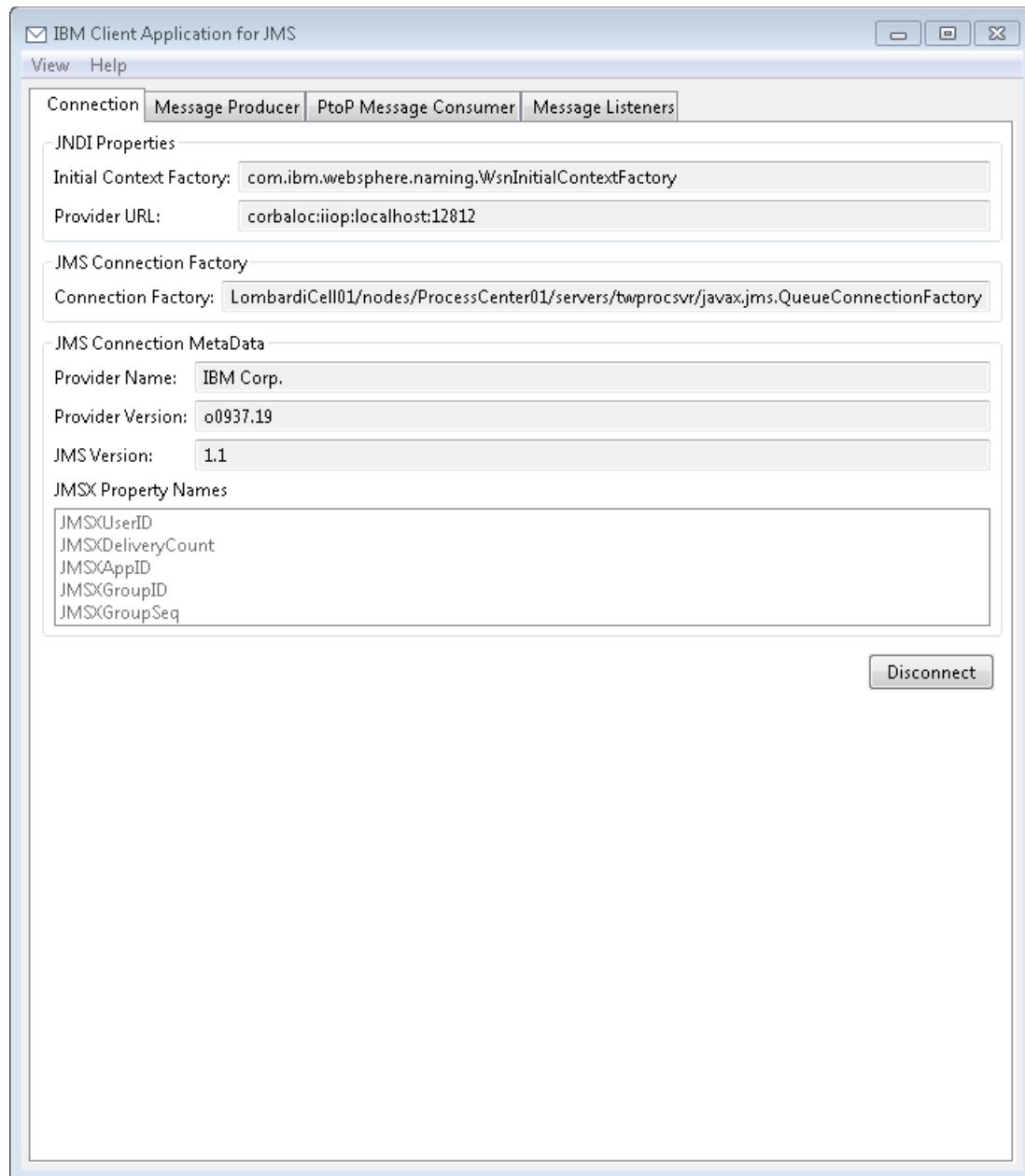


Click the Browse JNDI button to bring up a list of connection factories:

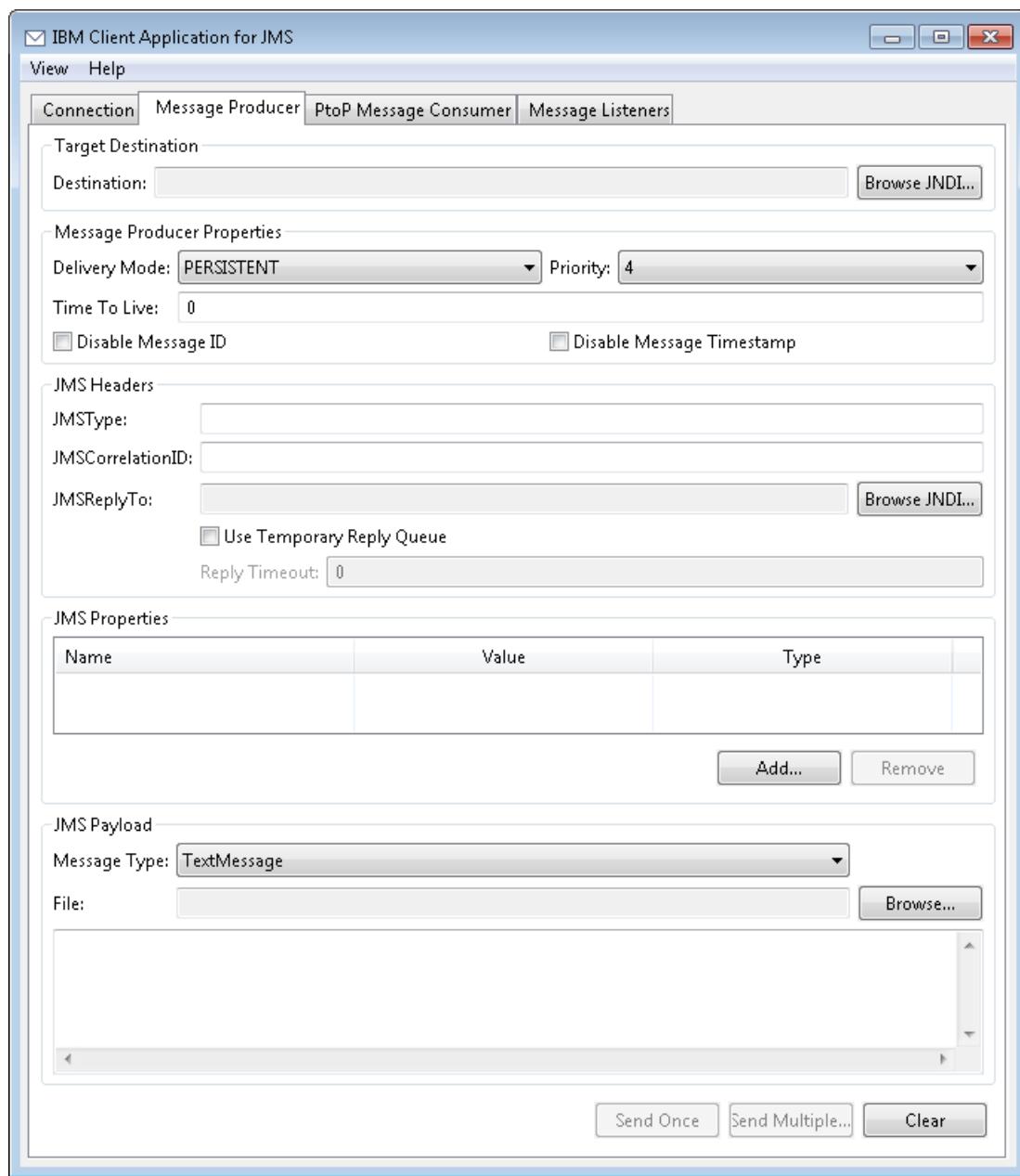


Pick the entry called `javax.jms.QueueConnectionFactory`

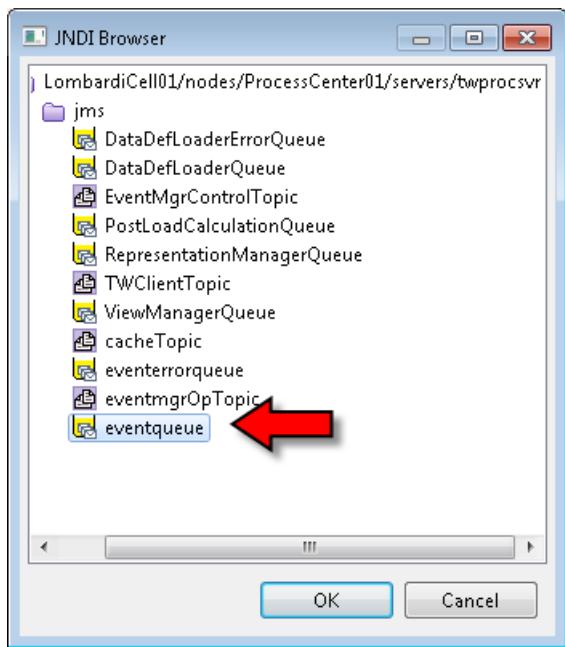
Click Connect to to connect to the WAS Server. The main page of the tool is shown:



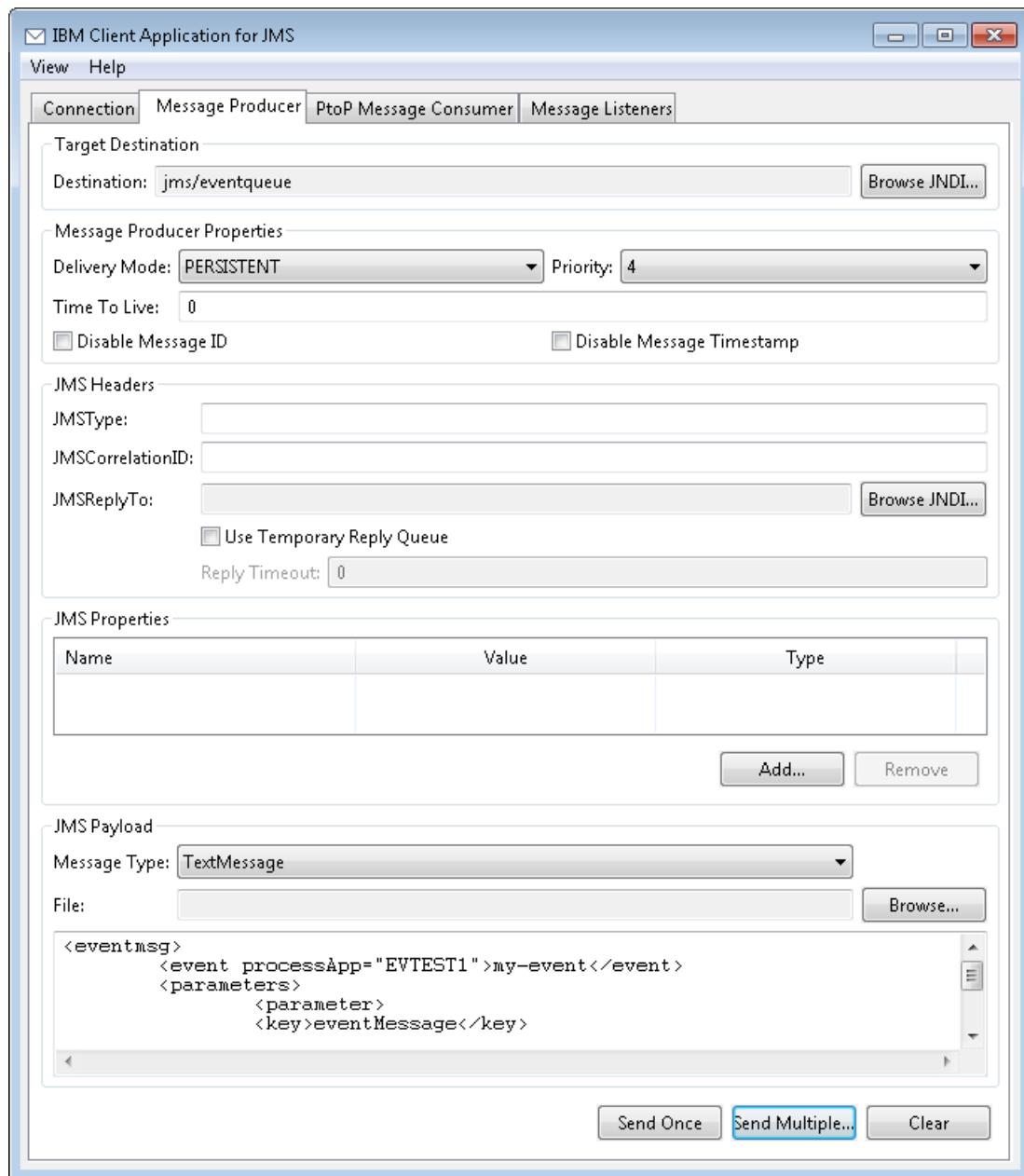
Click on the Message Producer tab to start the task of sending (producing) a message:



Click the **Browse JNDI** button beside the Destination input. This will produce a list of known queues as found by listing the JNDI registry. Pick `eventqueue` from the list:



In the JMS payload section, enter the data that you wish to place on the queue. The content should be XML as previously described. Finally press the Send Once button to send a single instance of the message to the queue.

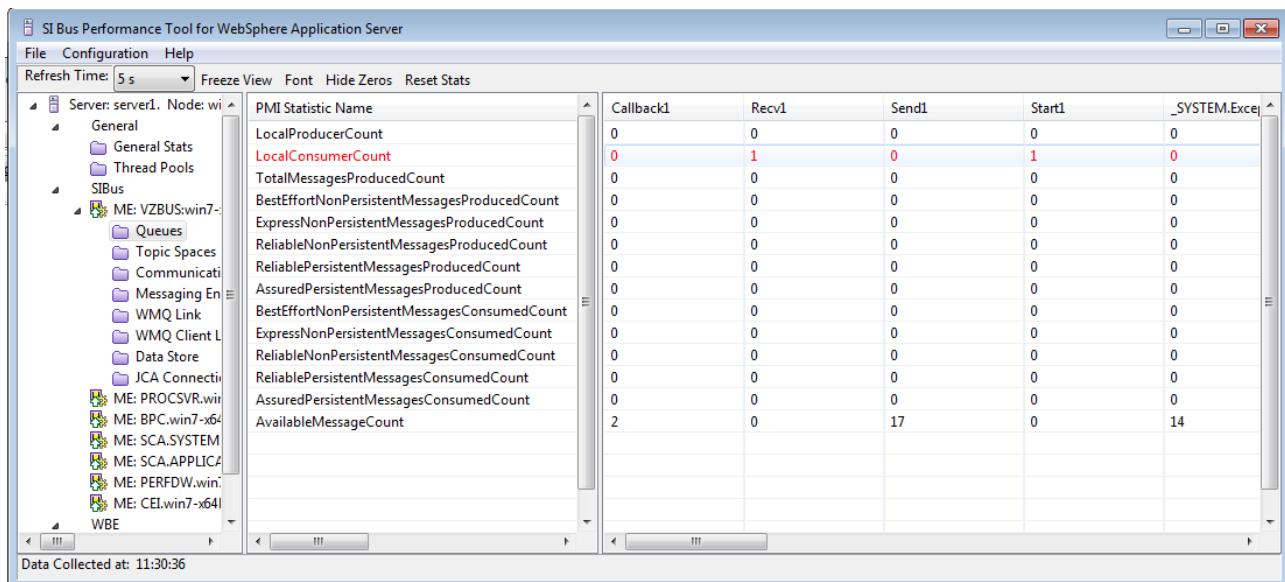


Once the message has been sent, IBPM will retrieve the message and process it.

In addition to the above tools, another tool "SI Bus Performance Tool" can be found here:

<https://www.ibm.com/developerworks/community/groups/service/html/communityview?communityUuid=13a714e5-6379-4325-aeee-c0c88ec3d15b>

This tool is especially useful for seeing if there are readers of SI Bus queues.



See also:

- [IBM Service Integration Bus Destination Handler, Version 1.1](#)

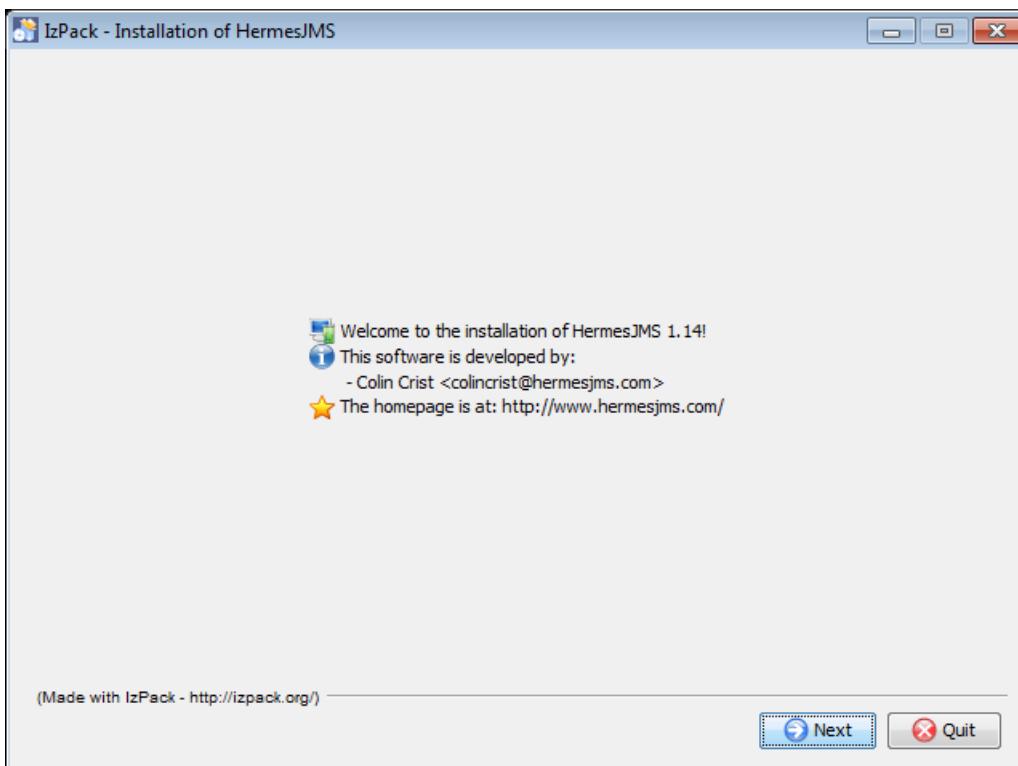
Hermes JMS

Hermes JMS is another good tool for working with JMS messages. It doesn't appear to have been modified since 2009 but it still seems to work. Although not advertised as working with WebSphere SI Bus, it does seem to work reasonably well.

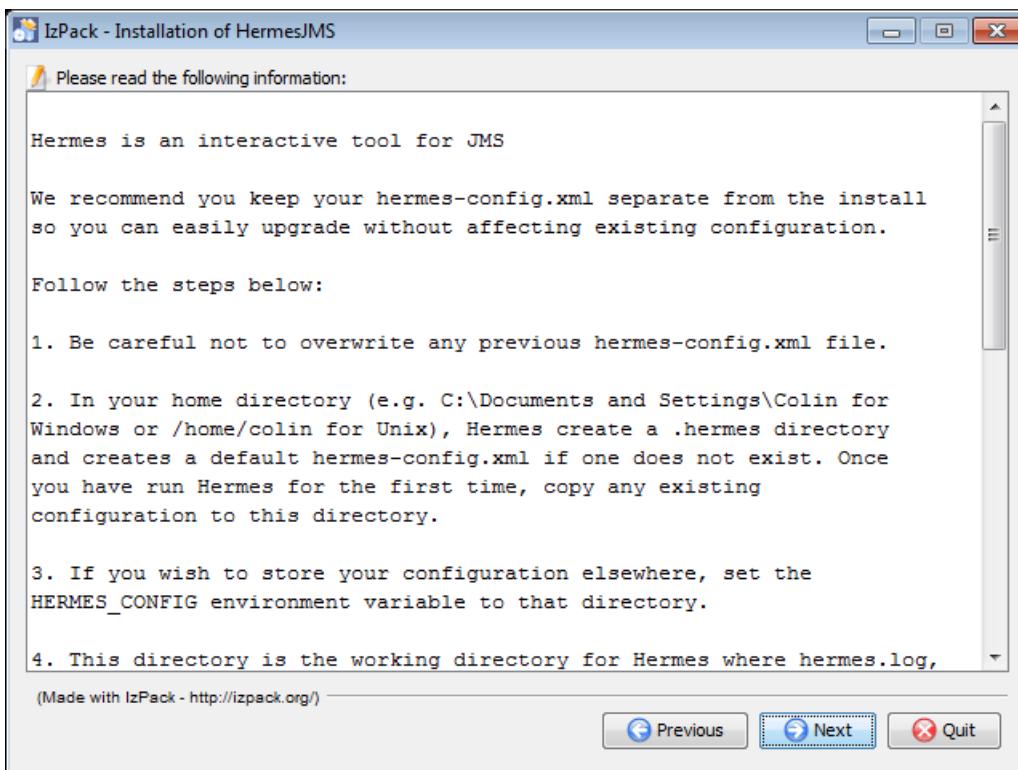
Download the hermes-install-xxx.jar

Run java -jar hermes-install-xxx.jar

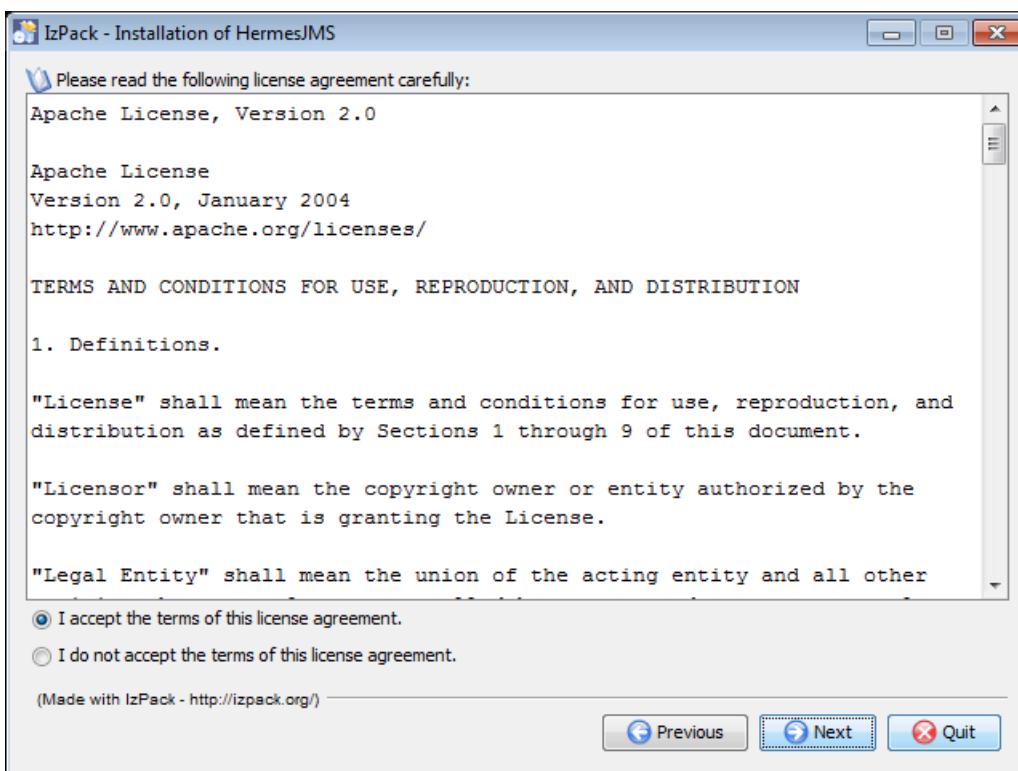
Once run, we see an initial welcome screen.



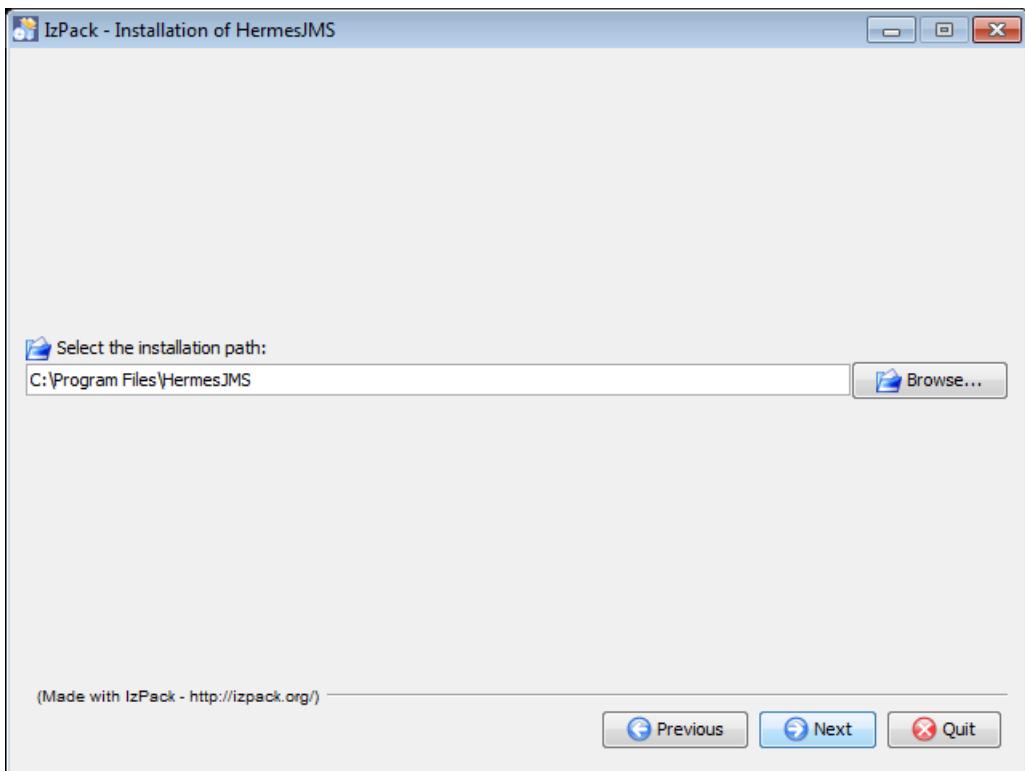
Next is a description of what HermesJMS is all about.



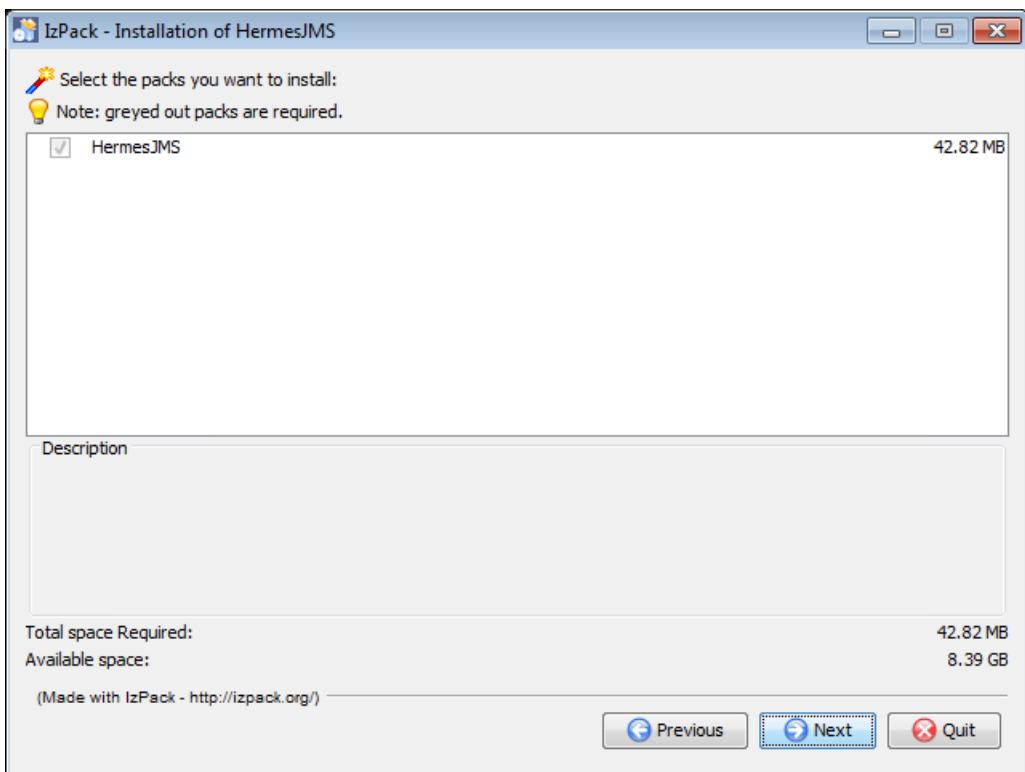
Now the license instructions.



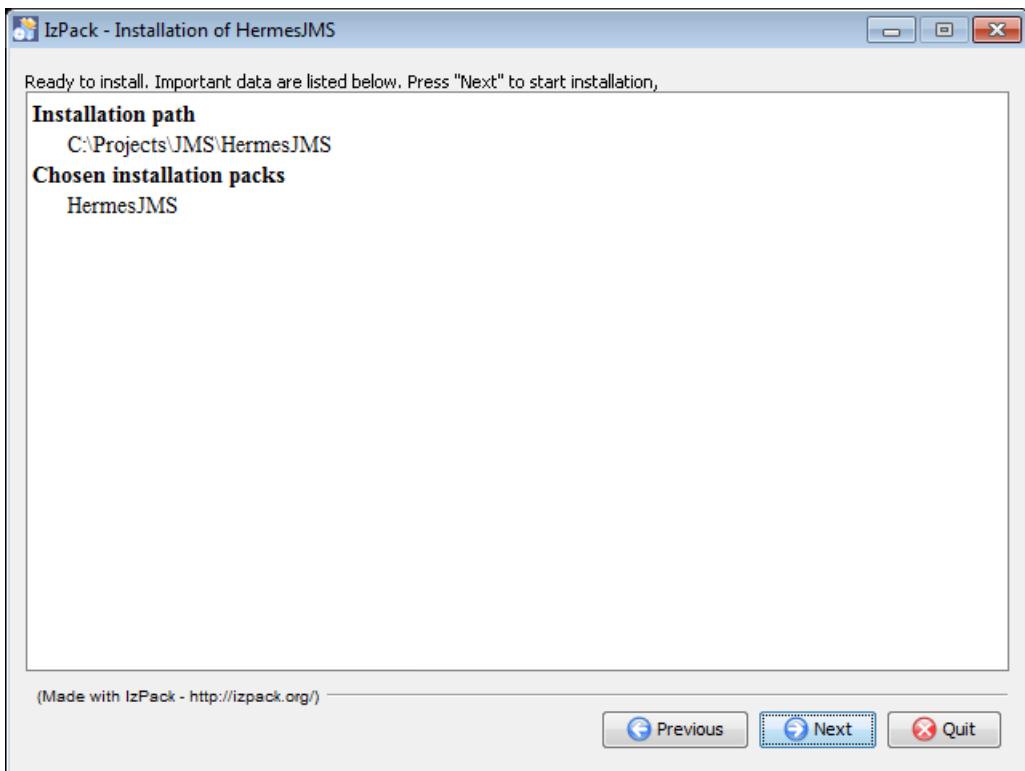
Next the folder into which the tool will be deployed.



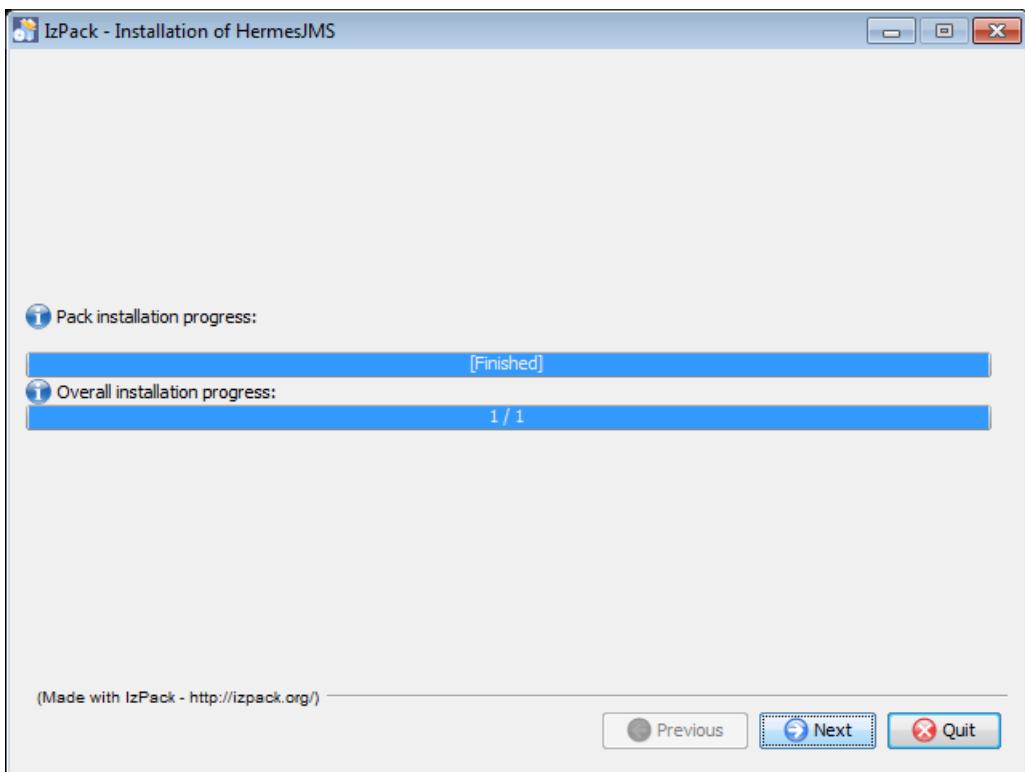
Now a prompt about what is going to be installed.



Now a final confirmation.



And a progress step.



In my experience running on Windows 7, this last window did not terminate and I was forced to quit. However, the tool was installed and seemed to run correctly. It appeared that the problem was caused by the tool trying to create a desktop icon which failed.

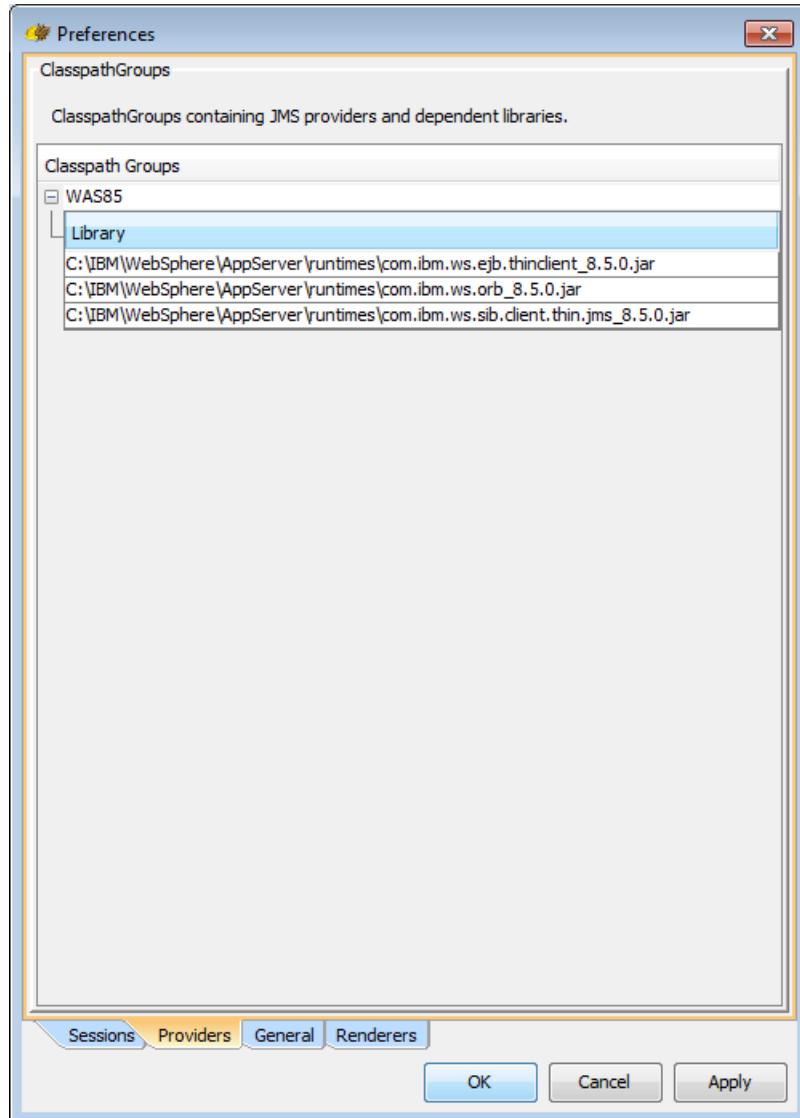
It is absolutely vital that Hermes run using the IBM JVM and not any other JVM. The easiest way to do this is to edit the "hermes.bat" and insert:

```
set JAVA_HOME=C:\IBM\WebSphere\AppServer\java
```

near the top of the file. Make sure that the path matches what you expect.

We also need to add the JAR files from WAS to allow connection. Open Options > Configuration and switch to the Providers tab. Right click in the background and select "Add Group". Set the "Classpath group name" to be "WAS85". Right click "Library" and select "Add JAR(s)". Add the following JAR files from the "<WAS Root>/runtimes/" folder:

- com.ibm.ws.ejb.thinclient_8.5.0.jar
- com.ibm.ws.orb_8.5.0.jar
- com.ibm.ws.sib.client.thin.jms_8.5.0.jar



In the target WAS server, we need to change RMI/IOP security settings. Open the WAS Admin console and go to Security > Global Security

Open the "CSIV2 inbound communications":

Authentication

Authentication mechanisms and expiration

- [LTPA](#)
- [Kerberos and LTPA](#)
- [Kerberos configuration](#)
- [Authentication cache settings](#)
- + [Web and SIP security](#)
- [RMI/IOP security](#)
 - [CSIv2 inbound communications](#)
 - [CSIv2 outbound communications](#)
- + [Java Authentication and Authorization Service](#)
- [Enable Java Authentication SPI \(JASPI\)](#)
- [Providers](#)
- [Use realm-qualified user names](#)

Change the "Transport" property to "SSL – supported"

CSIv2 Transport Layer

Client certificate authentication

Supported

Transport

SSL-supported

Repeat the above for the "CSIv2 outbound communications" setting.

If you fail to do the above, an error similar to the following will be received

```
'javax.naming.NamingException: Error getting WsnNameService properties [Root exception is org.omg.CORBA.TRANSIENT: initial and forwarded IOR inaccessible vmcid: IBM minor code: E07 completed: No]
```

<http://www-01.ibm.com/support/docview.wss?uid=swg21614221>

Next we need to create a new context. The parameters that need set are:

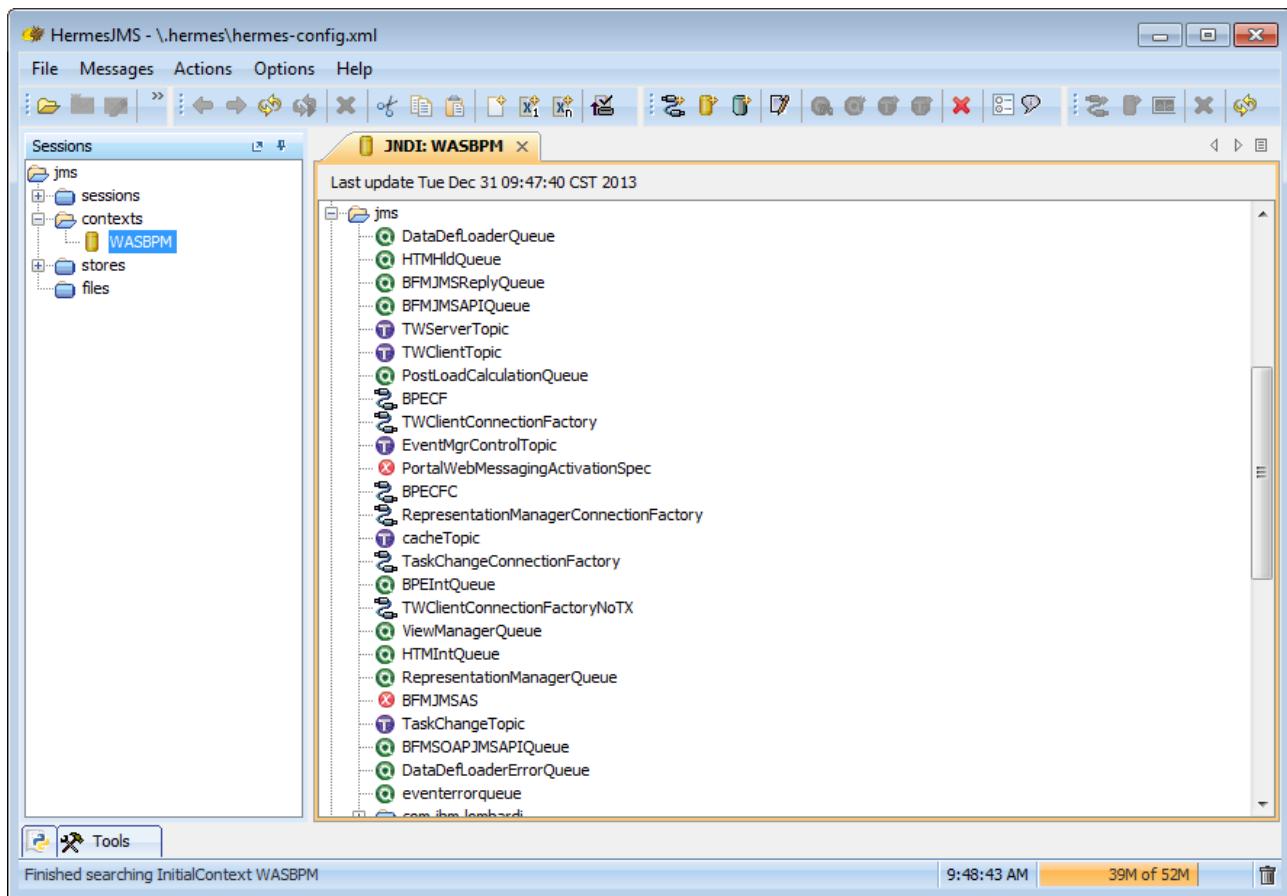
- loader – Set this to WAS85 to load the correct JARs
- providerURL – This will be set to iiop://<hostname>:<bootstrapPort>
- initialContextFactory – This will be set to com.ibm.websphere.naming.WsnInitialContextFactory

Also had to fiddle with IIOP settings when I received:

```
'javax.naming.NamingException: Error getting WsnNameService properties [Root exception is org.omg.CORBA.TRANSIENT: initial and forwarded IOR inaccessible vmcid: IBM minor code: E07 completed: No]
```

<http://www-01.ibm.com/support/docview.wss?uid=swg21614221>

Double clicking on the new context will show a list of JNDI entries. In the jms folder, we will find connection factories, queues and topics.



When the WAS server hosting SI Bus is configured, a port is defined for SI Bus client connections. The WAS logical name for this is "SIB_ENDPOINT_ADDRESS". When we configure a WAS queue connection factory, we must also define the endpoint providers property to allow the client to connect to the correct provider. For example:

```
Provider endpoints
localhost:7278:BootstrapBasicMessaging
```

In the WAS Bus > Security settings, change the permitted transports to work over all types:

Buses > MONITOR.PCCell1.Bus > Security for bus MONITOR.PCCell1.Bus

Configure the security settings for your service integration bus.

Configuration

Launch Bus Security Wizard

General Properties

Enable bus security

Inter-engine authentication alias
MonitorBusAuth

Permitted transports

Allow the use of all defined transport channel chains
 Restrict the use of defined transport channel chains to those protected by SSL
 Restrict the use of defined transport channel chains to the list of permitted transports



Use the Server ID when running mediations

Experience seems to be showing that after adding a QCF or CF we need to restart the server for it to show up in JNDI.

In a QCF definition, we also need to provide a "prover endpoint"

localhost:7281:BootstrapBasicMessaging

WebSphere Default Messaging

The JMS implementation supplied by WAS is called Default Messaging or sometimes System Integration Bus (SI Bus).

See also:

- [Redbook - WebSphere Application Server V7 Messaging Administration Guide](#)

REST Integration

From IBPM 7.5 onwards, a REST interface has been formally provided for the product. REST is a protocol where client requests can be transmitted over an HTTP connection to achieve queries and work on the server. The detailed documentation for the REST APIs can be found in the IBM online InfoCenter.

http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/topic/com.ibm.wbpm.ref.doc/rest/bpmrest/rest_bpm_wle.htm

Although the majority of function can be achieved through the product supplied REST capabilities, many appear to be missing that one would have assumed to be present. Read and study the available method calls as carefully as you can.

When making a REST request, HTTP provides an option to supply the verb or command. These are prescribed by the HTTP protocol and include:

- GET
- POST
- PUT
- DELETE
- others

Take care to use the correct REST HTTP command that is associated with the function that is to be performed. Typically commands that retrieve data use the HTTP GET while commands that update or create some BPM resource use POST or PUT.

Note that some requests, specifically PUT and DELETE may not be allowed to pass through firewalls. A solution to this is to add an HTTP header to the REST request with:

x-method-override = {PUT|DELETE}

or

x-http-method-override = {PUT|DELETE}

And send the request over a POST request which will be allowed to pass. For BPD related REST requests, a URI parameter can be added ... eg.

<http://....?&x-method-override=PUT>

For some REST command, the command may simply be too long if passed as a URL. A solution to this is to send the REST request as a POST, set the x-http-method-override HTTP header to be the command and place the parameters of the REST request in the body of the POST HTTP request. The HTTP Content-Type header should be set to "application/x-www-form-urlencoded".

Yet another way of sending data to the server is to use POST and set the content to be multipart/form-data

For example, setting the Content-Type to be:

multipart/form-data; boundary=-----4827543632391

then a body could be:

```
-----4827543632391
Content-Disposition: form-data; name="instanceId"
123
-----4827543632391
```

```

Content-Disposition: form-data; name="name"
myname
-----4827543632391
Content-Disposition: form-data; name="docType"
file
-----4827543632391
Content-Disposition: form-data; name="data"; filename="MyDoc.txt"
Content-Type: text/plain
Hello
-----4827543632391--

```

To use this multipart/form-data style can be quite tricky. However, if called from Java, the Apache HTTP Components may come to our aid.

A module in this offering is called `MultipartEntity`. An instance of this object can have multiple "parts" added to it with the `"addPart ()"` method. A part appears to be of data type `ContentBody` which has two interesting concrete implementations for us:

- `StringBody` – A string value
- `ByteArrayBody` – An array of bytes

From this, it appears that we can build a `MultipartEntity` object and add to that a series of "parts". Once done, we can then use that `Multipart` entity as the payload of a REST request. Each part has an associated "name" which is supplied in the `addPart()` method.

See also:

- DeveloperWorks - [Integrating a business process application in IBM Business Process Manager V7.5.1 with an external system using the REST API](#) - 2012-02-08
- DeveloperWorks - [Using the REST APIs in IBM Business Process Manager V7.5](#) – 2011-08-31
- Making REST calls from GWT
- [Apache HTTP Components](#)

The REST Functions

The API exposed by the product includes a rich set of capabilities:

Process Model: get	
Process Applications: get	
Process Applications: post	Create a new Process Application from a BPMN 2.0 archive
Toolkits: get	
Toolkits: post	
Process: start	
Process: send message	
Process Instance: get	
Process Instance: suspend	
Process Instance: resume	
Process Instance: terminate	
Process Instance: update process instance	
Process Instance: update due date	
Process Instance: start adhoc	
Process Instance: delete document	
Process Instance: add comment	

Process Instance: fire timer	
Process Instance: add document	
Process Instance: update document	
Process Instance: delete document	
Process Instance: evaluate JavaScript	Execute JavaScript within the context of a process
Process Instance Queries: get	
Process Instance Query Attributes: get	
Process Instance Query Entity List: get	
Process Instance Query Entity List Count: get	
Service Model: get	
Service: start	
Service: currently running	
Service: get data	
Service: stop	
Service: resume	
Service: set data	
Service: evaluate JavaScript	
Task Template: get	
Task Template Client Settings: get	
Task Template Queries: get	
Task Template Query Attributes: get	
Task Template Query Entity List: get	
Task Template Query Entity List Count: get	
External Activity Model: get	
Task: bulk claim	
Task: bulk cancel	
Task: get next	
Task Actions: get	
Task Instance: get	
Task Instance: start task	
Task Instance: assign task	
Task Instance: update	
Task Instance: finish	
Task Instance: claim	
Task Instance: cancel	
Task Instance Client Settings: get	
Task Instance Queries: get	
Task Instance Query Attributes: get	
Task Instance Query Entity List: get	
Task Instance Query Entity List Count: get	
Exposed Items: get	

Search Metadata: get	
Search: perform custom search	
Performance Query: get	
Performance Instance Query: get	
Users: get	
User: get	
User: put	
Groups: get	
Group: get	
Systems Metadata: get	
Asset List: get	
Branch Snapshot List: get	
Project Named Snapshots List: get	
Snapshot Change History: get	
Compare History: get	
Process Application Settings: get	

Handling REST errors

Not all requests to IBM BPM will always succeed. Errors can be returned. The format of an error is as follows:

```
{
  status: "error"
  Data: {
    errorMessage: "Text of message",
    errorMessageParameters: [ // array of parameters // ],
    errorNumber: "Error message number",
    exceptionType: "Java exception name"
    responses: // unknown //
  }
}
```

The REST API Tester

Baked into IBPM is a fantastic tool for testing REST APIs. It is a web based application that can be reached at:

<http://<hostname>:<port>/bpmrest-ui>

When launched from a browser, it shows a list of the categories of the different REST requests:

Business Process Manager | REST API Tester

Result Type:
JSON (Javascript format)

Select API Call:

- + REST Interface for BPEL-related Process Resources
- Business Process Manager REST APIs
 - + Process API
 - + Service API
 - + Task API
 - + Task Template API
 - + Search API
 - + Business Object API

Selecting an entry shows the details of that entry and allows values for the parameters to be supplied. From there, an Execute Call button can be pressed which executes a live call to the server and returns the data from the server for examination.

Business Process Manager | REST API Tester

IBM.

Result Type: JSON (Javascript format)

Select API Call:

- Show Adhoc Events
- + **Service API**
- **Task API**
 - External Activity Model**
 - Task Details
 - Start Task
 - Assign Task To User
 - Assign Task To Group
 - Assign Task To Me

External Activity Model
Retrieves the model data for an external activity

Task ID: 157

Request: http://localhost:9080/rest/bpm/wle/v1/externalactivity/157/model

Status: 200 - OK

Header:
Content-Type: application/json Content-Encoding: gzip Content-Language: en-US Transfer-Encoding: chunked Date: Tue, 14 Jun 2011 14:51:42 GMT Server: WebSphere Application Server/7.0

Result:

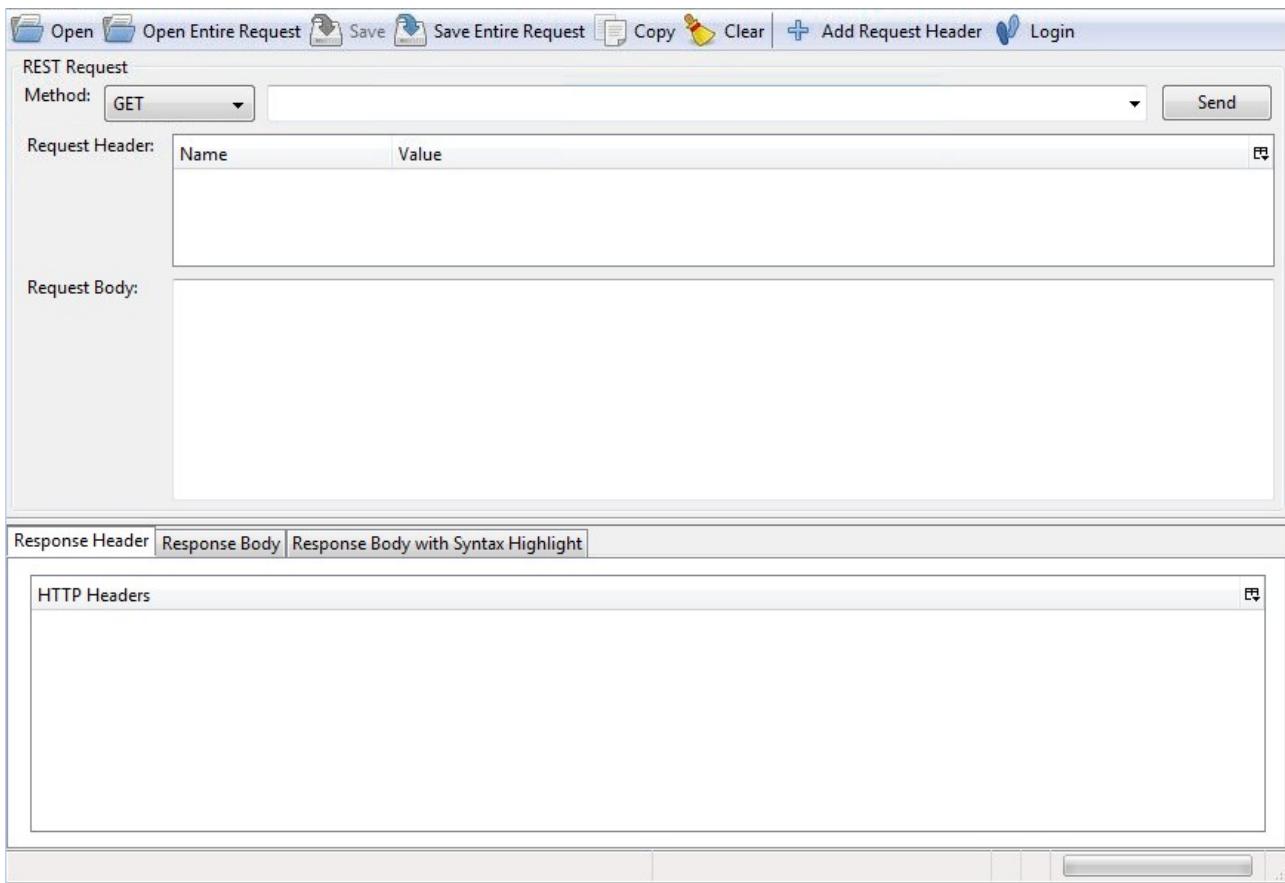
```
{
  status: "200",
  data: {
    name: "ExtAct1",
    customProperties: {},
    inputs: {
      in1: {
        isList: false,
        type: "String"
      }
    },
    outputs: {
      out1: {
        isList: false,
        type: "String"
      }
    },
    validations: {
      String: {
        type: "string",
        description: "String Type"
      }
    },
    ID: "60.73a55a58-1cd4-4534-8951-c31ec49114c3"
  }
}
```



The data can be selected to be returned as either JSON or XML.

Alternatives to the REST API Tester tool include the popular soapUI and the FireFox plug-in called [RESTClient](#).

After installing RESTClient, it can be opened in the FireFox environment and shows a window that looks as follows:



Working with REST Search Queries

One of the more interesting REST APIs is called "Search" and is generically available at:

`/rest/bpm/wle/v1/search/query`

This request is available through both PUT and POST HTTP REST commands but surprisingly not through GET even though this seems like an ideal and logic candidate for a GET request.

The operation can take a number of parameters as query strings. These include:

Name	Description
columns	See description below. A comma separated list of column names.
condition	See description below.
sort	
secondSort	
organization	There appears to be two distinct query return type organizations. One returns task instances and the other returns process instances. This parameter declares which type of search should be performed. The choices are "byTask" or "byInstance".
saveAsName	If this search is to be saved for later re-use then a name value may be supplied here.
run	
shared	

The `columns` is a comma separated list of column names of data that can be returned. The list of columns that can be supplied is itself available from a REST request called:

`/rest/bpm/wle/v1/search/meta/column`

As of IBPM version 8.5 the list looks as follows:

- assignedToUser
- assignedToRole
- instanceName
- instanceId
- bpdName
- instanceDueDate
- instanceCreateDate
- instanceModifyDate
- instanceStatus
- instanceSnapshot
- instanceProcessApp
- taskDueDate
- taskPriority
- taskSubject
- taskStatus
 - Comment
 - Received
 - Closed
- taskReceivedDate
- taskClosedDate
- taskSentTime
- taskReadTime
- taskReceivedFrom
- taskClosedBy
- taskActivityName

In addition to the column names listed above, there are a set of additional columns that are **commonly** returned. Note, do not include these columns in the search columns. These are:

Property	Description
taskId	The ID of the task
taskAssignedTo	Who is the ?? assigned to. Can be null or not present.
assignedToUser	Who is the ?? assigned to. Can be null or not present,
taskAttachedExtActivityRef	
taskAttachedInfoPathFormRef	

The format of a returned JSON structure looks as follows:

```
{  
    status: "200",  
    data:  
    {  
        data:  
        [  
            {  
                <columnName>: <columnValue>,  
                <columnName>: <columnValue>,  
                ...  
                <columnName>: <columnValue>,  
            },  
            ...  
            {  
                <columnName>: <columnValue>,  
                <columnName>: <columnValue>,  
                ...  
                <columnName>: <columnValue>,  
            }  
        ]  
    }  
}
```

```

        }
    ],
    organization: {"byTask"|"byInstance"),
    autoColumns:
    [
        "assignedToUser",
        "instanceId",
        "instanceStatus",
        "taskAssignedTo",
        "taskAttachedExtActivityRef",
        "taskAttachedInfoPathFormRef",
        "taskCreatedByBpdFlowObjectId",
        "taskCreatedByBpdId",
        "taskId"
    ]
}
}
}

```

As can be seen, the most interesting aspect of this is the `data` field which is an array of objects where each object is a set of properties corresponding to the retrieved "columns". The "autoColumns" are the names of additional columns returned beyond those explicitly requested.

The condition part of the REST request allows us to filter or constrain the records returned on a query. A condition takes the form:

`<columnName>|<operation>|<value>`

The allowable operations are:

- Equals
- NotEquals
- Contains
- StartsWith
- LessThan
- GreaterThan

multiple conditions can be supplied and these logically "ANDed" together. There is no deeper query mechanism than this (for example logical "ORs"). Not all column names are supported for condition evaluation. A REST request of the form:

`/rest/bpm/wle/v1/search/meta/constraintColumn`

returns the list of column names that may be used in conditions. As of 8.5 these are:

- assignedToUser
- assignedToRole
- bpdName
- instanceCreateDate
- instanceDueDate
- instanceId
- instanceModifyDate
- instanceName
- instanceProcessApp

- instanceStatus
- instanceSnapshot
- taskActivityName
- taskDueDate
- taskPriority
- taskStatus
- taskSubject

An example condition might be:

- `instanceProcessApp|Equals|A110822` – True if the Process App is "A110822"
- `bpdName|Equals|<Name>` and `taskActivityName|Equals|<Name>` – True for a specific task type within a given BPD type

Another REST exposed service that can be used in conjunction with searches is called Search Meta Data and is exposed at:

`GET /rest/bpm/wle/v1/search/meta/type`

The type parameter defines what type of search data is desired. As of 7.5.1, the following types and their content are defined:

- `column` - The names of the columns that can be included as columns to be returned
- `constraintColumn` - The names of columns that can be included in constraint expressions
- `taskStatus` – The different status that a task may have. Currently these are:
 - `New_or_Received`
 - `Actioned`
 - `Alert`
 - `Answered_Help_Request`
 - `Closed`
 - `Comment`
 - `Deleted`
 - `Followed`
 - `Forwarded`
 - `Help_Request`
 - `Ignored_Help_Request`
 - `New`
 - `Received`
 - `Replied`

- Sent
 - Special
- businessData – The names of business data fields that can be returned as columns
- savedSearch – The names of saved searches
- instanceStatus – The name of statuses of process instances. Currently these are:
 - Active
 - Completed
 - Did_not_Start
 - Failed
 - Suspended
 - Terminated
- priority – The names of allowable priority settings. Currently these are:
 - Highest
 - High
 - Normal
 - Low
 - Lowest

Working with REST Task Instance Queries

This is an area that the author is still researching but one which appears to be of high importance. The concept appears to be that a query can be executed to retrieve task information but the responses seem very different from simple searches. It appears that there is a wealth of possibilities here. The general syntax is:

```
GET /rest/bpm/wle/v1/tasks/query/{queryName}?<parameters>
```

- selectedAttributes – string -
- interactionFilter – String – One of the following values:
 - WORK_ON
 - WORK_ON_ACTIVE
 - ASSESS_AVAILABLE
 - ASSESS_AND_WORK_ON
 - CHECK_COMPLETED
 - BROWSE_ALL
- queryFilter – string -
- searchFilter – string -

- processAppName – string -
- sortAttributes – string -
- offset – integer -
- size – integer -
- filterByCurrentUser – boolean

Each search has attributes/items and each attribute is defined with:

- name
- displayName
- description
- isArray
- sourceAttribute
- content
- sourceQueryTableIdentifier
- isSortable
- isFilterable

An example of an IBM existing saved search is "IBM.DEFAULTALLTASKSLIST_75" which contains:

Name	Column
TAD_DISPLAY_NAME	taskSubject
TAD_DESCRIPTION	taskNarrative
DUE	taskDueDate
ACTIVATED	taskReceivedDate
PRIORITY	priority
STATUS	taskStatus
STATE	taskStatus
ORIGINATOR	taskReceivedFrom
COMPLETED	taskClosedDate
OWNER	assignedToUser
ASSIGNED_TO_ROLE_DISPLAY_NAME	assignedToRoleDisplayName
NAME	taskActivityName
TKIID	taskId
AT_RISK_TIME	taskAtRiskTime
IS_AT_RISK	taskIsAtRisk
PI_PIID	instanceId
CONTAINMENT_CTX_ID	instanceId
PI_NAME	instanceName
PI_DISPLAY_NAME	instanceName

PT_PTID	bpdid
PROCESS_APP_ACRONYM	instanceProcessApp
SNAPSHOT_NAME	instanceSnapshot
SNAPSHOT_ID	instanceSnapshotId
KIND	

Working with REST Task Instances

There are a set of REST requests that will work with instances of tasks. These include the following operations:

- Get the details of a task
- Start a new task
- Assign a task to a user or group
- Update the due date or priority of a task
- Complete/close a task
- Claim a task against a user
- Cancel a task

All of these commands start with the general prefix of:

/rest/bpm/wle/v1/task/{taskId}

The taskId value can potentially be retrieved from a preceding query/search call.

Getting Task and Instance details through REST

The method to get the details of task look as follows:

GET /rest/bpm/wle/v1/task/{taskId} [?parts={string}]

where parts can be one of "data", "all" or "none". The default is "all". An example response is shown next:

```
{
  "status": "200",
  "data": {
    "activationTime": "2011-08-31T18:14:53Z",
    "clientTypes": ["IBM_WLE_Coach"],
    "completionTime": null,
    "containmentContextID": null,
    "description": null,
    "displayName": "Step: Call EA1",
    "dueTime": "2011-08-31T19:14:53Z",
    "kind": "KIND_ORIGINATING",
    "lastModificationTime": "2011-08-31T18:14:53Z",
    "name": "Call EA1",
    "originator": "tw_admin",
    "owner": null,
    "priority": 30,
    "startTime": "2011-08-31T18:14:53Z",
    "state": "STATE_RUNNING",
    "tkiid": "1223",
    "piid": null,
    "status": "Received",
    "priorityName": "Normal",
    "assignedTo": "All Users_T_da7e4d23-78cb-...b2262",
    "assignedToType": "group",
    "data": 
```

```

{
  "bpdToken": {
    "bpdInstanceID": {"data": 1422},
    "tokenID": "2"
  },
  "variables": {
    {
      "in1": {
        "f1": "f1Val",
        "f2": "f2Val",
        "f3": "f3Val",
        "f4": {
          {
            "x": "hi",
            "y": 0,
            "z": false
          }
        }
      }
    }
  },
  "externalActivityID": "60.0b902a52-8df2-44ed-85e2-04bbd980b01d",
  "serviceID": "...",
  "nextTaskId": "...",
  "processInstanceId": "...",
  "isAtRisk": "...",
  "collaboration": ...
}
}

```

Some of the fields in this data structure need explanation:

Field Name	Description
activationTime	
atRiskTime	
clientTypes	
completionTime	
containmentContextID	
description	
displayName	
dueTime	
isAtRisk	A flag that indicates that this task is at risk.
kind	<ul style="list-style-type: none"> • KIND_PARTICIPATING - ??
lastModificationTime	
name	
originator	
owner	
priority	
startTime	
state	<ul style="list-style-type: none"> • STATE_READY – When a Task has been created but not yet claimed, it is in this state. • STATE_CLAIMED – When a Task has been claimed by an individual user, it is in this state. • STATE_FINISHED – When a task has been completed, it is in this state.

tkiid	The Task instance id
piid	The Process instance id
processInstanceName	
status	<ul style="list-style-type: none"> • Received – Task has been received • Closed – Task has been closed
priorityName	
assignedTo	
assignedToDisplayName	
assignedToType	<ul style="list-style-type: none"> • user – The task is assigned to a user • group – The task is assigned to a group
data	
processData	
serviceID	<p>The serviceID field is extremely useful. This is the identifier for the service that created the task. Through this, we can get service description and from this, the expected input and output data types. See also:</p> <ul style="list-style-type: none"> • Getting a Service Model using REST
flowObjectID	
nextTaskId	
collaboration	

One of the most important aspects of the Task data structure is that it contains the variables passed in and/or returned from the task. Ignoring the rest of the structure, these variables can be found at:

```
{
  data:
  {
    data:
    {
      variables:
      {
        varName:
        {
          fieldName: ...
          fieldName: ...
        }
      }
    }
  }
}
```

The field values are pretty much what you would expect. They are strings for String, numbers for numbers and nested JSON objects for deeper data types ... however ... arrays (lists) seem somewhat different.

Consider a data type that looks as follows:

The screenshot shows the 'Business Object' configuration screen. The 'Common' tab displays basic information: Name (BO1), System ID (guid:56d35d2f221c8d0e1e970b64:1324725ef0f), Modified (admin (Sep 8, 2011 12:34:38 PM)), and Documentation (empty). The 'Behavior' tab shows Definition Type as 'Complex Structure Type'. The 'Parameters' tab lists 'f1 (String) (List)' with buttons for Add, Remove, Up, and Down. The 'Parameter Properties' tab shows f1 as a String variable type, selected as a list, with documentation and selection buttons.

This is a data type that contains a field which is an array of strings. If we have a task that has an input variable of this type, what we find is that the Task data returned looks as follows:

```
{
  data:
  {
    data:
    {
      variables:
      {
        in1:
        {
          f1:
          {
            items: "[\"a\", \"b\"]",
            selected: "[]"
          }
        }
      }
    }
  }
}
```

When a query is made to the run-time to retrieve the tasks/process instance, it is made as a specific user. What this means is that **only** those tasks for that user are actually shown. This means that it is not possible (using the REST APIs) to retrieve a list of tasks/processes for all users. The JavaScript API called TWSearch allows for the query of all tasks/instances without regard to who owns or can work on them.

See also:

- Searching for processes and tasks from JavaScript
- Error: Reference source not found

Getting a template for a task

There is a REST call that retrieves a template for a task. At a high level it looks as follows:

```
GET /rest/bpm/wle/v1/taskTemplate/{templateId}
```

What is a templateId?

Finishing a Task

Another Task oriented REST request available to us performs the job of completion of a task. The format of this command is one of:

```
PUT /rest/bpm/wle/v1/task/{taskId}?action=finish&params={json}  
POST /rest/bpm/wle/v1/task/{taskId}?action=finish&params={json}
```

To set the values of the variables, what we want to do is get a list of all the expected output variables of that task. Imagine, for example, a task that has an output variable called "out1" of data type "BO1". The completion of an instance of this task would then be achieved by returning a JSON object that contains a property called "out1" (which is the same name as the variable) where this property contains a JSON child object representation of the data.

An example to complete this task would then be:

```
/rest/bpm/wle/v1/task/1260?action=finish&params={"out1":{"a":"a1", "b":"b1", "c":"c1"}}
```

Note that the `action=complete` is a synonym for `action=finish`.

Experience seems to show that for data types of type List, the data coming back is of the format:

```
{  
  "selected":  
  [  
    ],  
  "items":  
  {  
  }  
}
```

Where selected is an array of integers of ordinal items selected and items is an array of items in the array. When setting a response of a list, experience seems to be showing that items can be set but **not** the selected entries. Attempting to set selected is resulting in errors.

If we have chained tasks, a property called "nextTaskId" contains an array of taskIds. Quite why this is an array and not a single task is still a mystery. If there is no next taskid, the array length is zero.

See also:

- External Implementation

Starting a Task

The Start Task REST request first struck me as an odd thing. It has a mandatory parameter ... a taskId. This confused me for the longest time. Why would we start a task if it already existed? What does it *mean* to start a Task? After a lot of play testing and an awful lot of help from others, something clicked. The idea of "Starting a Task" is synonymous with the idea of "Starting to work on a task". This is opposed to the other interpretation which is "creating a task instance". Starting to work on a task comes into play when one thinks of measuring process metrics such as how long did it take someone to work upon a task.

Consider two distinct tasks both of which complete 4 hours after the process created them. In one task, a user started working on it as soon as it was created and it took him 4 hours to complete the work associated with it. This resulted in a wall clock time of 4 hours from when the task was created to when it was finished. For the second task, the user started working on it 3 hours and 30 minutes after the task was created and finished his task 30 minutes later. Again this resulted in a wall clock time of 4 hours from when the task was created to when it was finished. Should we

consider both tasks the same? The answer is no. The first task took a person 4 hours while the second task took a person 30 minutes. This obviously means that the tasks were not of the same nature. More, if one is looking to improve the process, one would want to focus on tasks that take longer to complete and simple wall-clock time of how long it took for the activity to complete is not always a good measure since we shouldn't assume that the task is worked on as soon as it is created.

The REST API to flag a Task as started is:

```
POST /rest/bpm/wle/v1/task/{taskId}?action=start
```

where the taskId is the identifier of an existing task.

An example return from this call looks as follows:

```
{
  "data": {
    "return": {
      "serviceStatus": "coach",
      "data": "{\"var1\":\"Hello\"}",
      "coach": "<Form xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>\r\n<FormTitle>IBM BPM Coach</FormTitle>\r\n<Layout id='section' styleClass=''\r\n</Layout>\r\n</Form>",
      "step": "Untitled1",
      "coachEvals": [
        {
        },
        "actions": [
          {
            "str": [
              "ButtonGroup0_Button0",
              "ButtonGroup0_Button1"
            ]
          },
          "key": "1456"
        }
      }
    }
  }
}
```

The return data from the Start request is very interesting. The first thing to note is that it returns the XML representation of the Coach itself before being passed through the XSLT conversion to make it HTML. In principle this could then be "parsed" by some GUI generation or interpretation tool to build a new UI. In practice though, my opinion is that this would not be a good idea. The XML representation of the Coach may change in future releases.

Another return is the current value of all the variables in the Service. This appears to be a single String that is JSON encoded.

In the "actions" section is a list of the identities of the buttons which can cause the Coach to complete. These can later be used to cause the Coach to complete through other REST API commands.

Claiming a task

When a task is created, it may be associated with a Team. This is a set of potential owners. When one of those potential owners looks at their task list, they will see the task available to them to work upon. What we want to happen is that when a specific user works on a task, that task should no longer be available to the other potential owners. Another way of saying this is that we want a task to be "claimed" by a particular user. Once it has been claimed, it shouldn't be able to be claimed by another user.

There is a REST API for claiming a task. It is:

```
POST /rest/bpm/wle/v1/task/{taskId}?action=claim
```

Re-assigning a task

A task can be re-assigned to a different user or group of users. This is achieved with the following REST command:

```
PUT (orPOST) /rest/bpm/wle/v1/task/{taskId}?action=assign [&toMe={boolean}] [&back={boolean}]  
[&toUser={string}] [&toGroup={string}] [&parts={string}]
```

The `taskId` is the identifier of the task to be re-assigned.

The `parts` parameter describes what data should be returned from this specific call.

The other parameters describe the new user or groups to which the task should be assigned.

- `toMe=true` – Assign this task to the user that is making the REST call
- `back=true` – Assign this task back to the original owner
- `toUser=string` – Assign this task to the named user
- `toGroup=string` – Assign this task to the named group

There appears to be *mysterious* rules governing who can perform an assignment of a given task and to whom it can be assigned. Read the previous sentence again carefully as it takes some thought to understand that there are actually **two** considerations at play there. Let us look at them both.

Given an arbitrary task X, the first question is "Can I actually perform an assignment upon it?".

The answer to that appears to be "yes, but only if you are either currently assigned to the task or you belong to a group to which the task is assigned".

The next question is "To whom can I assign the task?". The answer to that appears to be "Another user who is also in a group that you yourself are also a member of or to a group that you are a member of".

Getting Task variable values

When a task is active, it has variables associated with it. There is a REST API that can be used to retrieve the values of those variables. At a high level it is:

```
GET /rest/bpm/wle/v1/task/{taskId}?action=getData&fields=fieldName
```

An example instance of a result would be:

```
{  
    "status": "200",  
    "data": {  
        "result": "{\"xyz\":\"Neil\"}",  
        "resultMap": {"xyz": "Neil"}  
    }  
}
```

Getting the client details of a task

When a task is associated with a Coach, we can retrieve some client details for that task. This is itself an additional REST call:

```
GET /rest/bpm/wle/v1/task/{taskId}/clientSettings/IBM_WLE_Coach
```

This returns the following data:

```
{
```

```

    status: "200",
    data: {
      url:"http://localhost:9080/teamworks/process.lsw?
zWorkflowState=1&zTaskId=1749&zResetContext=true"
    }
}

```

The important property is called "url". This contains the URL that can be opened to show the task.

Working with REST Processes

Through the REST APIs we have the ability to work with Processes and instance of processes.

Starting a process from REST

A Process Instance can be started with the Process Resource POST (start) method which generally looks like this:

```
POST /rest/bpm/wle/v1/process?action=start&bpId={string}{ [&snapshotId={string}] | [&processAppId={string}]} [params={string}]
```

Only *one* of snapshotId or processAppId should be supplied. It is mandatory that one of them be supplied to identify which process should be started.

The parameters are GUIDs.

The values for the start REST command can be retrieved from the Exposed Items (with filter) resource:

```
GET /rest/bpm/wle/v1/exposed/process
```

From the exposed details, the fields of interest are:

REST term parameter	Exposed Value	Example
bpId	itemID	25.xxx
snapshotId	snapshotID	2064.xxx
processAppId	processAppID	2066.xxx

There is also a property within the data returned on exposed processes which is called "startURL" which can be used to directly start the process. For example:

```
"/rest/bpm/wle/v1/process?action=start&bpId=25.acbeda41-aa7c-4748-adb0-e3c031bca280&processAppId=2066.06402192-a621-49fc-a9b3-65976c821c3f"
```

An example response from the start request may look as follows:

```
{
  status: "200",
  data: {
    creationTime: "2012-01-07T02:49:09Z",
    description: null,
    executionState: "Completed",
    lastModificationTime: "2012-01-07T02:49:09Z",
    name: "BondProcess:231",
    pid: "231",
    processTemplateID: "25.895c6548-91ef-4862-916b-5cec3460e79d",
    processTemplateName: "BondProcess",
    processAppName: "Surity",
    processAppAcronym: "A111229",
    snapshotName: null,
    snapshotID: "2064.b1e4c8b4-b742-4c0b-a1b6-74a2fb4cc35a",
  }
}
```

```

        dueDate: "2012-01-21T02:49:09Z",
        comments: [
        ],
        tasks: [
        ],
        documents: [
        ],
        actionDetails: null,
        data: "{}",
        variables: {
        },
        executionTree: {
            executionStatus: "2",
            root: {
                name: "BondProcess",
                nodeId: "1",
                children: null,
                createdTaskIDs: null
            }
        },
        diagram: {
        }
    }
}

```

See also:

- Getting Exposed Processes
- REST encoded UUIDs/GUIDs
- TechNote - [Error CWTBG0019E: Unexpected Exception occurs when starting the process using the REST interface with IBM Business Process Manager](#) – 2011-06-22

Getting details of a process instance using REST

The details of a process instance can be retrieved through a REST call if we know the process instanceId. The syntax for the REST request is as follows:

```
GET /rest/bpm/wle/v1/process/{processInstanceId}
```

Within the REST API Tester, this function is known as:

WLE REST APIs > Process API > Current State

```

status: "200",
data:
{
    creationTime: "2012-04-20T15:51:02Z",
    description: null,
    executionState: "Active",
    lastModificationTime: "2012-04-20T15:51:27Z",
    name: "BPD1:943",
    piid: "943",
    processTemplateID: "25.218bd752-b612-4c8b-9fc7-e1a69ac6e925",
    processTemplateName: "BPD1",
    processAppName: "Attachment Security",
    processAppAcronym: "A120420",
    snapshotName: null,
    snapshotID: "2064.0a006243-cb2d-45fa-aa4b-cde8d8c05ecc",
    dueDate: "2012-05-04T15:51:02Z",
    comments: [],
    tasks: [],
    documents: [
    {
        name: "dddd",
        type: "FILE",
        date: "2012-04-20T15:51:22Z",
        length: 6,
    }
]
}

```

```

        ID: "103"
    }],
    actionDetails: null,
    data: "{}",
    variables:
    {
    },
    executionTree:
    {
        executionStatus: "1",
        root:
        {
            name: "BPD1",
            nodeId: "1",
            children: [
            {
                name: "Untitled1",
                nodeId: "4",
                children: null,
                flowObjectId: "bpdid:56d35d2f221c8d0e:-2c8e90b7:136a1cc0df6:-7e2a",
                tokenId: "4",
                createdTaskIDs: ["884"]
            }],
            createdTaskIDs: null
        }
    },
    diagram:
    {
    }
}

```

Let us look at the `executionTree` section. Examination holds the following to be the best understood model of this data:

```

{
    executionStatus: "1"      // Unknown
    root: {
        name: "Process Model Name",
        nodeId: Integer,          // Unknown
        children: {},
        createdTaskIDs:          // Unknown
    }
}

```

The children items appear to have the following format

```

{
    name:
    nodeId:
    children: {},
    flowObjectId:           // Step ID of the BPD step that token is upon
    tokenId: String,        // Id of the token
    createdTaskIDs: Array of String // TaskIds created for this step
}

```

Boiling this down to the commonly asked for function. What if I want to know where within my current process I currently am?

It seems that getting the Process Instance data and drilling down to:

```
executionTree.root.children[*]
```

will return the meat of what we are looking for.

Getting details of a process model from REST

```
GET /rest/bpm/wle/v1/processModel/{bpId}?snapshotId={String}
```

This returns:

- Name of process
- Data model including inputs and outputs

Starting a UCA from REST

A UCA can be *fired* through a REST request. The triggering of a UCA can be used to start a process through a Start Message Event or unblock a process waiting on an Intermediate Message Event.

```
POST /rest/bpm/wle/v1/process?action=sendMessage&message={string}
```

The format of the message is:

```
<eventmsg>
<!-- The process app acronym and event name are required: The snapshot and UCA name are optional -->
<event processApp="[acronym]" snapshot="[snapshot_name]"
ucaname="[UCA_name]">[event_name]</event>
<!--Optional: The name of the queue for the event to run in -->
<queue>[queue name]</queue>
<!--Any parameters the UCA may require -->
<parameters>
  <parameter>
    <key>param1</key>
    <value>! [CDATA[value_for_param1]]</value>
  </parameter>
  <parameter>
    <key>param2</key>
    <value>value_for_param2</value>
  </parameter>
</parameters>
</eventmsg>
```

The response from the POST request indicates whether or not the request message was received:

```
{
  "status": "200",
  "data": {"messageSent": true}
}
```

An example of current usage of this REST request is to start a new instance of a BPD process with data. Even though there is a Start Process REST request, in 7.5.0 this does **not** accept input data to be passed into the process whereas the technique of having a process started with a Start Message Event and a UCA works.

See also:

- Message Start Event
- Invoke UCA
- Undercover Agents (UCAs)
- Java Message Service – JMS

Moving a token within a process using REST

The concept of a token is the step within a BPD process instance that is the "current" step. The REST API provides the capability to "move" tokens from one step to another.

```
POST /rest/bpm/wle/v1/process/{instanceId}?action=moveToken&tokenId={string}&target={string}
```

The tokenId is a simple numeric value (eg. 4) and is obtained from a process instance query. The target is the IBM BPM id of the step in the process (eg. bpdid:56d35d2f221c8d0e:-3cac44a9:13b35102749:-7de2). This can also be determined from a process instance query.

See also:

- Token Management

Retrying a failed instance

```
GET /rest/bpm/wle/v1/process/{instanceId}?action=retry
```

Listing and starting Ad-Hoc Events from REST

The API request called:

```
GET /rest/bpm/wle/v1/process/{instanceId}/actions
```

returns entries describing what can be performed against the identified process instance. This includes whether or not an Ad-Hoc start event is possible. This is called "ACTION_INJECT_TOKEN". A further API can be called which can be used to start such an Ad-Hoc start event. This API is called:

```
POST /rest/bpm/wle/v1/process/{instanceId}?action=adhoc&step={stepName}
```

Working with REST Services

Amongst the primary concepts of IBPM is that of “Services”. Services are the parts of a process that are usually linked together. As an entity, services have their own set of REST APIs available for use.

Starting a service from REST

An API call is available to start a service. It is defined as:

```
POST /rest/bpm/wle/v1/service/{serviceModelId}?action=start...
```

The REST API expects an `serviceModelId` parameter which can be either the `serviceModelId` parameter from the service model ID or it may be of the form

```
"<project-shortname>@<service-name>"
```

The only types of service that can be started are "AJAX" or "Human". The identity of the caller can be found from the `tw.system.user_loginName` property.

The optional parameters to this REST call include:

- `createTask (Boolean)` – Should a task be created in which this service runs?
- `parts (String)` – Which response items should be returned. Values can be:
 - `data`
 - `all`
 - `none`
- `params (String)` – A JSON expression describing the input parameters
- `snapshotId (String)` – The id of the snapshot to be used to start the service
- `callerTaskId (String)` – The id of the task that is associated with starting this service

An example response looks as follows:

```
{  
  "status": "200",  
  "data": {  
    "serviceStatus": "coach",  
    "key": "@42",  
    "step": "Collect MDN",  
    "data": {"allData": null},  
    "coach": "<Form>...</Form>",  
    "coachEvals": {},  
    "actions": ["ButtonGroup0_Button0"],  
  }  
}
```

```

        "actionsMap": {"ButtonGroup0_Button0": "Submit"}
    }
}

```

The values of serviceStatus seems to describe what the next step will be. Values seen include:

- coach – We have reached a coach
- end – The service has ended

Getting data from a Service

An API call is available to retrieve variable values from the running Service. The format of the call is:

```
GET /rest/bpm/wle/v1/service/{instanceId}?action=getData&field={variableName}
```

The variable name should be the full name as in tw.local.{varName}. Simply supplying the name of the variable by itself is not sufficient.

An example response looks as follows:

```
{
  status: "200",
  data:
  {
    result: "{\"xyz\":\"XYZVal\"}",
    resultMap:
    {
      xyz: "XYZVal"
    }
  }
}
```

Note:

The following was written pre 7.5.1 and appears to no longer be true. It is being left here for just now pending more detailed analysis. Experiment and review what works and doesn't work to be sure what is and is not currently applicable Author.

Experience seems to also show that a call to Start a task is required before this API may be used. This is not adequately explained or understood (yet).

Experience also seems to show that the ability to execute JavaScript from a REST call is required. To set this up, edit the 99Local.xml configuration file and change the following:

```
...
<enable-javascript-execution>true</enable-javascript-execution>
</common>
```

Again it is not yet known why this has to be changed and there may be security implications.

Setting data into a Service

Data variable values can be changed inside the content of a running service. This will most likely be a Human Service step.

```
GET /rest/bpm/wle/v1/service/{instanceId}?action=setData&field={variableName}
```

Getting a Service Model using REST

An API is available that returns the model of a Service. This is the declaration of what the interface of the service looks like. The API looks as follows:

```
GET /rest/bpm/wle/v1/serviceModel/{serviceId}
```

The serviceId parameter defines the ID of the service to be queried. This can be either the UUID of the service or the service identifier supplied in the format of “<ProcessAppShortName>@<ServiceName>”. The UUID of a service can be obtained from the field serviceID in the details of a returned task. This can then be used to associate a task to a service that created it.

Field	Description
header	Header information
diagram	A diagram of the model
dataModel	Abstract meta data description of the data expected as input and returned from the service

Here is an example of returned data:

```
{
  status: "200",
  data:
  {
    header:
    {
      name: "HS1",
      UUID: "guid:56d35d2f221c8d0e:-5a405e30:134fcb20625:-7f8d"
    },
    diagram:
    {
      step:
      [
        {
          name: "End",
          x: 700,
          y: 100,
          type: "ExitPoint",
          ID: "ProcessItem.548c1796-57aa-4b5c-a287-3a8a31eb5df4"
        }
      ]
    },
    dataModel:
    {
      properties:
      {
        v1:
        {
          isList: false,
          type: "BO1"
        },
        v2:
        {
          isList: false,
          type: "String"
        }
      }
    }
  }
}
```

```
inputs:
{
    v1:
    {
        type: "BO1",
        isList: false
    }
},
outputs:
{
    v2:
    {
        type: "String",
        isList: false
    }
},
validation:
{
    String:
    {
        type: "string",
        description: "String Type"
    },
    BO1:
    {
        properties:
        {
            a:
            {
                isList: false,
                type: "String"
            },
            b:
            {
                isList: false,
                type: "String"
            },
            c:
            {
                isList: false,
                type: "String"
            }
        },
        type: "object"
    }
}
```

Let us now look at the description of the data model. The data model is **not** instance data. It is not associated with a specific instance of a service. Instead, the data model is the *model* of what is expected as inputs and outputs from the service. The data model needs some interpretation so let us slow down and try and define some terms.

First, let us define something called a field. A field has three attributes:

- name – the name of the field
 - type – the data type of the field
 - isList – a boolean indicator that describes if this is a list (array)

A data model starts with three named containers called “properties”, “inputs” and “outputs”. Each of these are a set of fields.

So, effectively what we have is:

```
{  
  properties:  
  {  
    fieldA, fieldB  
  }  
  inputs:  
  {  
    fieldA  
  }  
  outputs:  
  {  
    fieldB  
  }  
}
```

There is a fourth container within the the data model that is called “validation”. This container contains a set of data types. A data type consists of the following properties:

- data type name – The name of the data type
- type – The type of the data type. This appears to be one of the following:
 - string – A String data type
 - object – A Business Object
- properties – If the type of the data type is “object”, then the properties container will be present. This is a set of fields describing the content of the object.

```
{  
  validation:  
  {  
    String:  
    {  
      type: "string",  
      description: "String Type"  
    },  
    BO1:  
    {  
      properties:  
      {  
        a:  
        {  
          isList: false,  
          type: "String"  
        },  
        b:  
        {  
          isList: false,  
          type: "String"  
        },  
        c:  
        {  
          isList: false,  
          type: "String"  
        }  
      },  
      type: "object"  
    }  
  }  
}
```

```
        }
    }
}
```

Working with Exposed Items

Getting Exposed Processes

The list of exposed processes can be retrieved with:

```
GET /rest/bpm/wle/v1/exposed/process
```

An example returned would be:

```
{
  status: "200",
  data:
  {
    exposedItemsList:
    [
      {
        ID: "2015.202"
        branchID: "2063.3321e2c8-3f9b-4240-84cd-918a120e00ee"
        branchName: "Main"
        display: "BPD2"
        itemID: "25.c23430e6-62fe-4794-9f02-4f840a19b2d7"
        itemReference: "/25.c23430e6-62fe-4794-9f02-4f840a19b2d7"
        processAppID: "2066.69fd9a5f-7e91-4d7f-a386-fcdd64f65209"
        snapshotCreatedOn: "2013-02-07T17:32:31Z"
        snapshotID: "2064.c6cf862c-9d56-4420-ab5b-478ebaed10e8"
        snapshotName: "2013-02-07-1"
        startURL: "/rest/bpm/wle/v1/process?action=start&bpdiId=25.c23430e6-62fe-4794-9f02-4f840a19b2d7&processAppId=2066.69fd9a5f-7e91-4d7f-a386-fcdd64f65209"
        tip: true
        type: "process"
      },
      ...
    ]
  }
}
```

Within these records, the following need further explanation:

- itemID – This is the GUID for the BPD model
- processAppID – This is the GUID for the Process App that contains the exposed process.
- snapshotID – This is the Snapshot ID within the Process App for the exposed process.
- ID

Given the information returned from an exposed process, on occasion we wish to know which Process App the exposed process is contained within. We see that we get its GUID code (2066.xxx) but that doesn't tell us the name of the Process App. Using the API to retrieve a list of all Process Apps, we can then match for this processAppID GUID and find the details for that application.

See also:

- Starting a process from REST

Working with REST Users and Groups

Users and groups known to IBM BPM can be queried through REST APIs.

The first API we will look at returns a list of all users:

```
GET /rest/bpm/wle/v1/users[?filter={string}][&parts={string}]
```

The filter parameter provides a suffix matching capability. For example, if the filter were "abc" then the returned users would only include those that start with "abc" (if any). The parts parameter allows us to control how much data is returned. Options are:

- all
- memberships
- none

```
{  
    "status": "200",  
    "data":  
    {  
        "users":  
        [  
            {  
                "userID": 9,  
                "userName": "admin",  
                "fullName": "Administrator",  
                "isDisabled": false,  
                "primaryGroup": null,  
                "emailAddress": null,  
                "userPreferences":  
                {  
                    "Locale": "en",  
                    "Alert On Assign And Run": "true"  
                },  
                "memberships":  
                [  
                    "Debug",  
                    "tw_admins",  
                    "tw_authors",  
                    ...  
                ]  
            }  
            ...  
        ]  
    }  
}
```

A second REST API allows us to obtain the details of a specific named user:

```
GET /rest/bpm/wle/v1/user/{userNameOrID} [?parts={string}]
```

The `userNameOrID` parameter identifies the user to be returned. To retrieve details on the current user, supply "current" as the name of the user.

The parts parameter may be one of:

- all
- none
- memberships

The response for the retrieval of information for a single user is the same as for a list of users but, of course, only one user is returned.

Working with Process Apps

A list of available Process Apps can be retrieved with the following command:

```
GET /rest/bpm/wle/v1/processApps
```

From within the REST API Tester, this command can be found in

Other > Retrieve Process Applications

A typical response looks like:

```
{
  status: "200",
  data:
  {
    processAppsList:
    [
      {
        shortName: "TWP",
        name: "Process Portal",
        description: "",
        lastModifiedBy: "Internal TW Admin user",
        defaultVersion: "Main",
        installedSnapshots:
        [
          {
            name: "7.5.1",
            acronym: "7.5.1",
            active: false,
            ID: "2064.a7b6c684-65c4-420e-b7be-0a504bba24c5"
          }
        ],
        ID: "2066.23d3ecec-6fdb-4033-9c57-e931aa13761f",
        lastModified_on: "2011-11-21T15:15:04Z"
      }
    ...
  ]
}
```

Of high importance is the “ID” field. The ID field is a UUID for the Process App. In some APIs, this is called the *projectId*.

Working with REST and heritage process document attachments

A heritage capability exists in which documents can be attached or associated with process instances. The REST interface provides capabilities to add a new document, update an existing document or delete a document. Notice that there is no API to retrieve a document. To retrieve a document, a separate URL/REST request is available. The list of documents (if any) associated with a process instance can be retrieved by examining the details of a process instance. From this, the document id can be determined which is used as a key in other requests.

Documents as managed by IBPM can be of two types known as "file" and "url". For a "url" type attachment, what is saved is a URL where the data for the attachment can be found. For a "file" type attachment, the data itself is saved by IBPM with the process instance. The name "file" is probably a bad name as it is really "data".

A common use of document attachments is to have the end user select a file from their local file system and have that attached to the process instance against which they are working. This is typically performed through a browser with a file upload button. When using IBM's Portal and Coaches, this is a straight forward task as the Coach provides a file upload ability. When writing a custom UI component, things become a bit more problematic. Because of obvious browser security rules, a browser client application (one running the browser) is not allowed to see the content of a file. Instead, when a file upload is requested, it is sent to a Web Server using an HTTP FORM (POST) mechanism. If we were to desire our own File upload mechanism to attach a document to a process instance, to what would we then send the HTTP FORM request? The answer is that we

ourselves would have to write such a function.

Heritage - Adding a document to a process instance using REST

A command is provided that looks as follows:

```
POST /rest/bpm/wle/v1/process/{processId}?action=addDocument&name={name}&docType={file|url}&{data=data|docUrl=url}
```

In this command, the following attributes are needed:

Parameter	Description
processId	The id of the process instance to which the document is to be attached.
docType	A value of "file" or "url" indicating which of the two possible types of attachment are to be used.
data	If the docType is "file", then this defines the data to be supplied and used in the request.
docUrl	If the docType is "url", then this defines the URL to be associated with the attached document.
name	Name associated with the document

It is likely that this API will be used in the mode of sending a POST request with the parameters supplied in the body of the request.

See also:

- Error: Reference source not found

Heritage – Getting a list of attached documents

Asking a process for its details returns a list of documents attached to it. The documents are attached in an array called "documents" which has the following structure:

```
{
  name: "ddddd",
  type: "FILE",
  date: "2012-04-20T15:51:22Z",
  length: 6,
  ID: "103"
}
```

Working with REST Asset Lists

This request retrieves a list of the assets in a Process Application that can be used to generate documentation. Examples of this API having been seen to be used include deriving the list of BPDs within a Process App.

```
GET /rest/bpm/wle/v1/assets? [&branchId=???] [&snapshotId=???]
```

Categories:

- Human Service – List of human services
- Participant
- Variable Type – List of variable type definitions
- BPD – List of BPDs
- Project Defaults

```
{
  status: "200",
```

```

data:
{
  "Human Service":
  [
    {
      poId:
      poVersionId:
      tags: [],
      name:
      last_modified_by_userId:
      last_modified_by_userName:
      last_modified:
    }
    ...
  ],
  "Participant": ...
  "Variable Type":
  "BPD":
  "Project Defaults":
}
}

```

Each entry in the categories is termed a “Persistence Object” and is composed of the following fields:

- poId
- poVersionId
- tags
- name
- last_modified_by_userId
- last_modified

For example:

```
{
  poId: "25.02ee9f21-5231-4f36-b6b5-8fbd108bcdd3",
  poVersionId: "043c21c4-751b-4287-bb9e-aafed6ed25ea6",
  tags: [],
  name: "BPD1",
  last_modified_by_userId: 9,
  last_modified_by_userName: "Administrator",
  last_modified: 2147483647
}
```

Getting Project branchIds and Snapshots

There is an API called “Get Project Named Snapshots List Resource”. This is a mouthful. It takes as input the “ID” value of a Process App or Toolkit (what the REST API calls a Project ID). Process App IDs start with code “2066.xxx”.

This API looks as follows:

```
GET /rest/bpm/wle/v1/project/{projectId}/snapshots
```

The `projectId` appears to be the ID field of a query of a Process App

What it returns looks like follows:

```
{
  status: "200",
  data:
  {
    branches:
    [
      {
        branchId: "2063.38158f45-7dc9-4293-bf55-e3176a4659ba",
    ]
  }
}
```

```

        name: "Main",
        snapshots:
        [
            {
                snapshotId: "2064.190135b9-9a63-4f1a-8cdb-94998b41916d",
                name: "Post Josh",
                desc: "",
                createdBy_userId: 9,
                createdBy_userName: "Administrator",
                created_on: 2147483647
            },
            {
                snapshotId: "2064.473163b7-f626-42b4-bb87-aa965d53efae",
                name: "v2.0",
                desc: "",
                createdBy_userId: 9,
                createdBy_userName: "Administrator",
                created_on: 2147483647
            }
        ]
    }
}

```

Notice that the response data contains the `branchId` entry. This is used extensively in other REST requests.

A question that may occur to you is “Where do I get the `projectId` from?”. The answer to this is that we can execute a command to retrieve a list of Process Apps. This is described Working with Process Apps. Process App Ids start with “`2066.xxx`”.

See also:

- REST encoded UUIDs/GUIDs
- Working with Process Apps

Getting Business Object definitions

The meta data description of a specific Business Object definition can be retrieved.

```
GET /rest/bpm/wle/v1/businessobject/{poId}?[branchId=...][snapshotId=...]
```

To obtain this detailed description , we need the “Persistence Object ID” of the Business Object that we wish to query. To obtain this we need to query the Asset List of a Process App or Toolkit. Remember, the Business Object is defined within that. In order to examine the Asset List of a Process App, we need the `branchId` of the Process App. That is itself a different REST call. When using that call we need to know the ID value of the Process App.

An example of a response looks like:

```
{
    "status": "200",
    "data":
    {
        "isComplex": true,
        "type": "BO1",
        "isShared": false,
        "properties": [
            {
                "typeClass": "String",
                "typeClassSnapshotId": null,
                "typeClassBranchId": null,
                "isArray": false,
                "name": "a"
            },
            {
                "typeClass": "BO2",
                "typeClassRef": "12.4ad5f41f-27c8-4c20-ad82-bd08c78ef529",
                "typeClassSnapshotId": "2064.0df8478f-e6aa-488d-9842-459ff6e1610f",
                "typeClassBranchId": null,
            }
        ]
    }
}
```

```

        "isArray": false,
        "name": "b"
    },
    {
        "typeClass": "String",
        "typeClassSnapshotId": null,
        "typeClassBranchId": null,
        "isArray": false,
        "name": "c"
    }],
    "name": "BO1"
}
}

```

See Also:

- Getting Project branchIds and Snapshots

Working with REST retrieved diagrams

A number of the REST requests retrieve descriptions of diagrams back from the run-time. These can be used to visualize various parts of the solutions.

Process Visual Model

```
GET /rest/bpm/wle/v1/visual/processModel/<bpdId>?snapshotId or branchId
GET /rest/bpm/wle/v1/visual/processModel/instances?
instanceIds=[<>] &bpdId=<>&snapshotId=<>
```

Return data

```
{
  status: "200",
  data:
  {
    name: {String}           // Name of the BPD
    poId: {String}          // BPD Id
    poVersionId: {String}
    identifier: {String}    // Name of property that is unique
    id: {String}            // A unique ID for this BPD element/step
    width: {Number}          // Width of the diagram
    height: {Number}         // Height of the diagram
    label: {String}          // The label of the image
    items: [],
    links: [],
    properties: [],
    textPositions: [],
    activeTasks: [],          // May be present
    tokens:                  // May be present
  }
}
```

Now let us look at `items`. Items appear to be the visual “things” on the screen. The most important of these is the “`items`” array. Properties seen in them include:

```
{
  lane: {String} // Swimlane that item lives in
  snapshotId:
  branchId:
  successors: [],
  preAssignment: {Boolean}
  postAssignment: {Boolean}
  conditional: {Boolean}
  eventActionType: {Number}
```

```

interrupting: {Boolean}
active: {Boolean}
id: {String}           // Unique id for this item
label:                 // Text label
x: {Number}            // Left position
y: {Number}            // Top position
type: {String}          // Type of item
category: {String}
colorIcon:
bpmn2TaskType:
poType:
poId:
attachedEvents:
loopType:
isSystem:
color:
width:
height:
children:
}

```

The core discriminator appears to be the “type” property.

Type = start or end

```

{
lane:
snapshotId:
branchId:
succesors:
attachedEvents:
preAssignment:
postAssignment:
conditional:
eventActionType:
interrupting:
active:
id:
label:
x:
y:
type:
category:
}

```

Type = activity

```

{
colorIcon:
lane:
bpmn2TaskType:
poType:
serviceType:
poId:
snapshotId:
branchId:
succesors:
attachedEvents:
loopType:
preAssignment:
postAssignment:
conditional:
eventActionType:
}
```

```
interrupting:  
active:  
id:  
label:  
x:  
y:  
type:  
category:  
}
```

Type = swimlane

```
{  
    isSystem:  
    color:  
    width:  
    height:  
    children:  
    id:  
    label:  
    x:  
    y:  
    type:  
}
```

For visualization of these, the important fields are:

- **id** – This is a unique key to the item on the screen.
- **x, y** – This is the X and Y coordinates of the item on the screen but relative to the swim lane.
- **label** – This is the label contained in the item. Experience seems to show that the label is already split using newline characters into logical lines. For example, if a line would be too wide for a visual item, it is split into newline parts.
- **type** – This is the type of item on the screen. Types seen include:
 - start
 - end
 - activity
 - msgStartEventInterrupting
 - adhocStartEvent
 - messageEndEvent
 - errorEndEvent
 - terminate
 - intermedCatchMsgEventNonBoundary
 - intermediateThrowMsgEvent
 - intermedCatchTimerEventNonBoundary
 - intermedCatchTackingEventNonBoundary
 - gateway

- gatewayAnd
- gatewayOr
- gatewayEvent
- group
- intermedCatchErrorEventBoundary
- intermedCatchMsgEventBoundaryInterrupting
- intermedCatchTimerEventBoundaryInterrupting
- category – This is the category of the item
 - StartEvent
 - Activity
 - EndEvent

For type of "activity"

- ServiceType – The type of activity
 - "Human Service"
 - "General System Service"
- bpmn2TaskType – The type of activity
 - "ScriptTask"
 - "UserTask"
 - "DecisionTask"
 - "ServiceTask"
 - "SubProcess"
 - "CalledProcess"
 - "None"

Let us look for a moment at the (x,y) position of an item. Each item seems to have an (x,y) pair of attributes that describes where that item should be positioned on the screen. Unfortunately, it is not quite as easy as that. The (x,y) coordinates are relative to the swim-lane in which the item lives. There is a class of item of type “swim-lane”. These items also have an (x,y) pair. For an arbitrary item that is to be displayed, we first need to find the swim-lane in which it lives and then add the (x,y) pair of the swim-lane to the (x,y) pair of the item to find the screen (x,y) pair.

Another key area within this data is the “links” array. This is an array of “links” which are the wires connecting one item to another. This data structure contains:

```
{
  needDefaultMarker: {boolean} // If true a marker is needed on the line to show that this is the
  default path.
  needDiamondMarker:
  id:
  name:
```

```

start: {id} // The id of the start item
startPosition:
end: {id} // The id of the end item
endPosition:
showName:
showEndState:
linkLabelPosition:
gfx: {String} // This appears to be a String of a JSON object that is an array of intermediate
points
inCriticalPath:
}

```

For visualization of these, the important fields are:

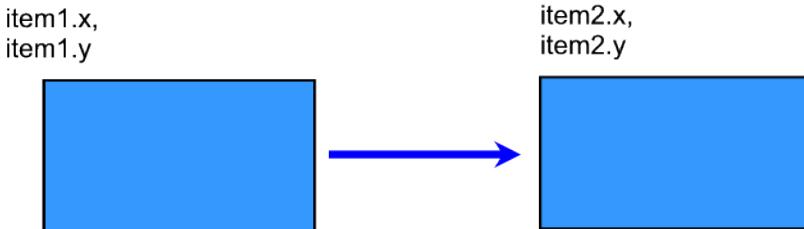
- `startPosition`, `endPosition` – Where on the item does the line start or end. Entries here include:
 - `rightCenter`
 - `leftCenter`
 - `leftTop`
 - `leftBottom`
 - `bottomLeft`
 - `bottomCenter`
 - `bottomRight`
 - `rightBottom`
 - `rightTop`
 - `topRight`
 - `topCenter`
 - `topLeft`
- `start`, `end` – Ids for the Items representing the start and end objects
- `gfx` – A string representation of a JSON object that is an array of points. The general format appears to be:

```
{
  intermediatePoints:
  [
    {
      x: {Integer},
      y: {Integer}
    }
  ]
}
```

But remember ... the `gfx` field appears to be a String and **not** an expanded JSON object. It has also been found that if the start and ends of the links are in different swim lanes then an offset of 24 pixels must be added to the "x" coordinate of the `intermediatePoints`. It appears that this is most likely a bug but is now unlikely to be corrected because of the potential negative side effects to existing technologies. It also seems that if the start and end items are in different swimlanes then the `intermediatePoints` are in absolute coordinates but if the start and end items are in the SAME swimlanes then we need to add the "y" offset of the swimlane.

The core model for links is that each link identifies the item from which the link should start and the item to which the link should end. These items are identified in the link data by the fields "start" and "end". These are uuids that correspond to the "id" field in the items.

So ... what then is the algorithm for drawing a link from one item to another?



First, we look at the `link.start` value. This will reference the `id` of an item. Getting to that item, we then know the `item.x` and `item.y` position. Similarly, looking at the `link.end` value, we get to a second item and learn its `item.x` and `item.y` position.

A directory can be found on the server which contains icons for all the different aspects of the product as shown in Process Designer. The directory is:

```
<Profile>\installedApps\<Cell>\IBM_BPM_Teamworks_{Node_Server}.ear\webviewer.war\n\widgets\com\ibm\bpm\wpd\images\BPD
```

Type	Size
activity	95x70
start	24x24

A closer look at tokens

If the "tokens" property is present, it has the following structure:

```
"tokens":\n{\n    "bpdid:56d35d2f221c8d0e:-1ff5e133:13895e09ee1:-7fe6":\n        [\n            {\n                "instanceId": "456",\n                "createTime": "2012-07-19T04:28:15Z",\n                "createTimeLong": 1342672095246,\n                "active": true,\n                "tokenId": "2"\n            }\n        ]\n}
```

Basically tokens is a list of items in the diagram keyed by bpdid.

Executing JavaScript in the context of a process from a REST request

An unusual REST API is exposed that allows an arbitrary fragment of JavaScript to execute within the context of a process. The API is defined as

```
POTS /rest/bpm/wle/v1/process/{instanceId}?action="js"&script=script
```

In order to be able to execute this command, the `enable-javascript-execution` flag must be set to true. If it is not set to true, the REST request returns "CWTBG0529E:Authorization Error, Javascript Expression execution is disabled". This flag can be found in 99Local.

An example use of this call would be to set the value of a process variable.

It was mentioned that this is an unusual API request as it allows for arbitrary code execution within the context of a process. It is not yet known what security implications this exposes. For example, who is allowed to execute such a request? Is it based upon an administrative permission or is it a function of participant groups associated with the process instance?

In general, I would recommend staying away from this API. The opportunity for breaking a process is high, there is no obvious auditing or tracking on this command and reviewing a BPMN diagram to understand what the process should do is "deviated" by using this API. If arbitrary code can be executed in the context of a process, working back in the path of such a process from the diagram won't show you how a result was reached.

Calling REST from Java

Although typically REST APIs are called from browser environments running JavaScript, there is absolutely nothing to prevent a Java client application from invoking REST directly. Here is an example application that makes a REST request. The core of the sample uses the Java supplied class `HttpURLConnection`. This class owns the ability to build and transmit correctly formatted HTTP requests. The sample also adds basic HTTP authentication data to ensure that the IBPM runtime knows the identity of the caller.

The response data is retrieved as a String. To use that data we will have to parse it as either a JSON data structure or an XML data structure depending upon the return type. It is strongly recommended to "play" with an individual REST call for a while before attempting to code up a solution. This will prepare you for understanding the data expected to be received by the runtime as well as the response sent back.

The HTTP security mechanism expects the userid and password to be Base64 encoded. Oddly, the Java 6 libraries don't appear to have a Base64 encoder supplied with them however the [Apache Commons "Codec"](#) package which is available for free download provides just what is needed.

```
package com.kolban.wle;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class Main {

    public static void main(String args[]) {
        try {
            URL url = new URL("http://localhost:9080/rest/bpm/wle/v1/systems");
            HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
            String wasUserName = "admin";
            String wasPassword = "admin";
            String authorization = "Basic " + new
String(org.apache.commons.codec.binary.Base64.encodeBase64(new
String(wasUserName + ":" + wasPassword).getBytes()));
            httpURLConnection.setRequestProperty("Authorization", authorization);
            httpURLConnection.setRequestMethod("GET");

            //httpURLConnection.setDoOutput(true);
            //httpURLConnection.setRequestProperty("content-type", "text/xml");

        /*

```

```

        * String data = "<data to write>"; OutputStreamWriter wr = new
        * OutputStreamWriter( httpURLConnection.getOutputStream());
        * wr.write(data); wr.flush();
        */

        System.out.println("Response: " + httpURLConnection.getResponseCode() +
": " + httpURLConnection.getResponseMessage());
        InputStreamReader inReader = new
InputStreamReader(httpURLConnection.getInputStream());

        BufferedReader br = new BufferedReader(inReader);
        String line = br.readLine();
        while (line != null) {
            System.out.println("Line is: " + line);
            line = br.readLine();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

To take advantage of passing the context data from an authenticated application, the following Java code fragment can be added:

If the caller is running in an authenticated context, then an LTPA token can be passed. Adding the following before connecting to the server will achieve this:

```

Set s = WSSubject.getCallerSubject().getPublicCredentials(WSCredential.class);
WSCredential creds = (WSCredential) s.iterator().next();
byte token[] = creds.getCredentialToken();
httpURLConnection.setRequestProperty("cookie", "LtpaToken=" +
Base64.encode(token));

```

When using JSON from a Java environment, there are basically two options. One can install the IBM freely distributed Feature Pack for Web 2.0 into the WAS server which will provide the [com.ibm.json.java](#) package or alternatively you can utilize the free open source package found at [www.json.org](#).

REST encoded UUIDs/GUIDs

Within the REST APIs, we appear to have a lot of “ids”. These appear to be long UUIDs. However, they seem to have encoding at their beginning that gives us an indication of their purpose. Here are some of the ones found so far:

Code	Description
2015	Process Template ID
2063	Branch ID
2064	Snapshot ID
2066	Process App ID
2072	Process Instance ID
2078	Task ID
1	Service Persistence Object

12	Variable Type Persistence Object
25	BPD Persistence Object
63	Project Defaults Persistence Object

We can convert a string that is encoded to one that is not encoded using:

```
value.replace(/^2072\./, "")
```

REST input and output data types

Processes and services can have input and output data. We can execute a query on a service or a process for the description of the data expected as input or returned as output. From this description, we can dynamically build an input structure. The format of the response of what input data looks like is interesting. Generically, it looks as follows:

```
dataModel:
{
  properties: {object},
  inputs: {object},
  outputs: {object},
  validation: {object}
}
```

The `inputs` object has a property for each of the inputs of the service. Each one of those properties is itself an JSON object that looks as follows:

```
{
  isList: boolean,
  type: string
}
```

The name of the property corresponds to the name of the parameter of the service. Let us now focus a little more on the `type` field. The `type` field describes the data type of the parameter. This is where the `validation` property comes into play. The `validation` property is again an Object. Each of its properties corresponds to a type in the `inputs` and `outputs` objects. These objects have the following form:

```
{
  type: string,
  description: string,
  properties: {object}
}
```

REST Security

When a request is made to the BPM server using REST, the caller must be authenticated to permit the appropriate action. Since the REST request is arriving over HTTP, WAS based HTTP security applies. There are a number of ways that authentication may be achieved. The first is the use of the technique called basic HTTP authentication. This means sending a userid/password pair with each request. Although this works, it is not the best nor most secure mechanism. With basic HTTP authentication, the password is passed in a simple Base 64 encoding which means that it is trivially de-codable. If basic authentication is chosen, then it is recommended to send the HTTP requests using SSL transport security to ensure that the content can not be examined. Instead of using basic authentication, LTPA tokens can be utilized. The concept here is that a user authenticates with the WAS server once and a token (a large string of characters) is returned. That token is then passed

back from the browser with each subsequent REST request. Since the token was generated by WAS and is known only to the browser that performed the initial login, this token is trusted by WAS and allows WAS to know the caller's identity based on the previous successful login that generated the token. The token is passed back implicitly to the WAS server through a browser held cookie.

To perform a login to WAS, the following simple web page can be used. This page is called "login.html".

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login</title>
</head>
<body>
    <h2>Form Login</h2>
    <form METHOD=POST ACTION="j_security_check">
        <p>
            <font size="2"> <strong> Enter user ID and password:</strong>
            </font> <br> <strong> User ID</strong> <input type="text" size="20" name="j_username"> <strong> Password </strong> <input type="password" size="20" name="j_password"> <br> <br>
            <font size="2"> <strong> And then click this button:</strong>
            </font> <input type="submit" name="login" value="Login">
        </p>
    </form>
</body>
</html>
```

When displayed, it looks as follows:

Form Login

Enter user ID and password:

User ID Password

And then click this button:

If the user fails to authenticate, a second page is needed to show an error. This file is called "error.html":

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <h1>A Form login authentication failure occurred</h1>
    <p>Authentication may fail for one of many reasons. Some possibilities include:
    <ol>
        <li>The user-id or password may be entered incorrectly; either misspelled or the wrong case was used.
        <li>The user-id or password does not exist, has expired, or has been disabled.
    </ol>
</body>
</html>
```

The final step is to update the Java EE WAR deployment descriptor to state that "FORM" based security is to be used to authenticate the user:

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>MYREALM</realm-name>
    <form-login-config>
        <form-login-page>/login.html</form-login-page>
        <form-error-page>/error.html</form-error-page>
    </form-login-config>
</login-config>
```

See also:

- [Basic Authentication](#)

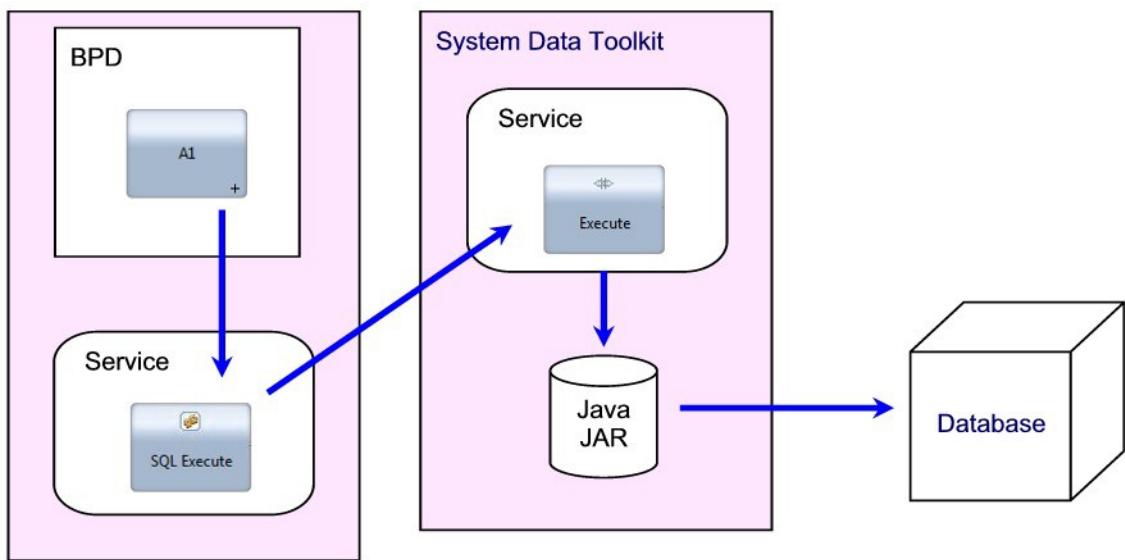
Database Integration

A commonly needed ability is the integration of data held in databases with a process. A database may hold prices, stock availability or other important information that may be desired to be used by a process. IBPM provides "connectors" to allow access to the database content held in tables.

Integration with supplied DB connectors

The functions to access databases are found in the System Data toolkit as implementation services. These services are themselves implemented in Java presumably using the Java JDBC bindings.

The overall story looks as follows:



A BPD that wishes to work with data from a Database will invoke a user written Service through an activity. This is business as usual. The called Service will wish to access the database. In order to do this it will leverage one of the IBM supplied pre-implemented Services found in the System Data Toolkit. When that is invoked, that supplied service calls native Java code supplied in a JAR packaged with the System Data Toolkit. That Java code utilizes JDBC to interact with the target database.

The list of available functions includes:

Name of Service	Description
SQL Blob to File	Read a Blob from the database and write the data to a file.
SQL Blob to File (SQL Statement)	
SQL Call Multiple Stored Procedures	Execute multiple store procedures.
SQL Call Multiple Stored Procedures (SQL Statement)	
SQL Call Stored Procedure	Call a single stored procedure
SQL Call Stored Procedure (SQL Statement)	
SQL CLOB to File	Read a CLOB from the database and write the data to a file.
SQL CLOB to File (SQL Statement)	
SQL Execute Multiple Statements	Execute multiple SQL statements.

SQL Execute Multiple Statements (SQL Result)	
SQL Execute Script (SQL Result)	
SQL Execute Statement	
SQL Execute Statement (SQL Result)	
SQL File to BLOB	Read binary data from a file and write it to the database.
SQL File to BLOB (SQL Statement)	
SQL File to CLOB	Read character data from a file and write it to the database.
SQL File to CLOB (SQL Statement)	
SQL Get Database Type	Determine the type of database.

The parameters of these service calls supplied in the System Data Toolkit are typically the following:

Parameter Name	Description
sql	This is a String representation of the SQL statement/query to be executed against the database.
parameters	These are a list of parameters to be replaced in the SQL expression. Each element of the list should be of type <code>SqlParameter</code> . These parameters are replaced in the SQL statement where '?' is found.
maxRows	This is the maximum number of rows to be returned.
returnType	This is a String representation of the return type. Typically this will be "XMLElement" but a complex data type can also be used.
dataSourceName	This is the WAS defined JDBC name as a String.

If the `returnType` is defined as `XMLElement` then a typical return from a query will be an `XMLElement` object with the following structure:

```
<resultSet recordCount="2" columnCount="3">
    <record>
        <column name="C1">A</column>
        <column name="C2">B</column>
        <column name="C3">C</column>
    </record>
    <record>
        <column name="C1">D</column>
        <column name="C2">E</column>
        <column name="C3">F</column>
    </record>
</resultSet>
```

Using JavaScript, we can then access the fields of the retrieved data as follows:

```
tw.local.result.record[0].column[0].getText() == "A"
```

If we wish, we can convert this `XMLElement` into a JavaScript object as follows:

```
var result = new Array();
var childRecord = tw.local.results.firstChild;
while(childRecord != null){
    var column = childRecord.firstChild;
    var row = new Object();
    while(column != null)
    {
        var name = column.getAttribute("name");
        var myText = column.getText();
        row[name] = myText;
        column = column.nextSibling;
    }
}
```

```

        }
        result.push(row);
        childRecord = childRecord.nextSibling;
    }
}

```

As an alternative to asking for the result to be an `XMLElement`, a Business Object data type name can be supplied. When the SQL is executed, a `List` object is returned where each element in the list is an instance of the Business Object data type. The columns in the returned data are matched to the names of the fields in the complex data type. Where there is a match, the returned column value is used as the value of the corresponding field.

A third solution, an currently my own favorite is to define the response type for the results to be "Record" and supply a "List of Record" as the receiver variable. This has all the advantages of defining a custom BO but does not need the overhead of a BO definition simply to exist to hold the SQL data returned.

See also:

- Data Type – Record

The `SqlParameter` structure used in some of these calls has the following attributes:

Name	Description
<code>value</code>	The value associated with the SQL parameter.
<code>type</code>	The type of the parameter. This is a DB data type such as <code>VARCHAR</code> , <code>DECIMAL</code> , <code>DATE</code> .
<code>mode</code>	The mode of the parameter such as 'IN' or 'OUT'.

It is recommended not to index a list of `SqlParameter` structures by ordinal value. Rather, create a variable and index by the variable. It is just better in the long run.

When working with Database integration, a recommended practice for building this is the following pattern:

1. Validate that the target database can be accessed with JDBC
2. Build SQL queries outside of WLE and validate that these queries work well. Consider using a tool such as [SQuirreL SQL](#) to build these commands.
3. Define JDBC connections in the Process Server WAS environment and test that the connections work.
4. Build out Integration Services that make use of the SQL Services in the System Data toolkit using the JDBC connection and the previously built and tested SQL queries.

See also:

- Operational Databases

Using LiveConnect and JDBC

JavaScript executing within a service runs on the Process Server and can use embedded Java in the form of LiveConnect. This means that JDBC programming can be explicitly included in the story:

```

try
{
    var context = new Packages.java.naming.InitialContext();
    var datasource = context.lookup("jdbc/TEST");
}
catch(myExp)
{
}

```

```

{
    log.error("Exp: " + myExp.message);
}

```

Example – Selecting rows from a table

In this example we will select and retrieve rows from a table. We will assume a table called EMPLOYEE that looks as follows:

Key	Name	Data type	Length	Nullable
EMPLOYEEID	VARCHAR	10	No	
FIRSTNAME	VARCHAR	40	No	
LASTNAME	VARCHAR	40	No	

We create a nested service activity and bind it to SQL Execute Statement. The data mapping for this activity looks like:



The SQL statement is:

```
"SELECT LASTNAME, FIRSTNAME FROM KOLBAN.EMPLOYEE WHERE EMPLOYEEID=?"
```

Note that there is a parameter for the EMPLOYEEID column in the WHERE clause. This is taken from a variable called parm1 which is defined as a list of SQLParameters.

The one and only element of that list variable is coded to be:

```
value = '123456'
type = 'VARCHAR'
mode = 'IN'
```

The result type is an XMLElement and a variable of that type called result is declared to hold the results. After execution, the LASTNAME and FIRSTNAME can be extracted through:

```
// LASTNAME
tw.local.result.record[0].column[0].getText();
// FIRSTNAME
tw.local.result.record[0].column[1].getText();
```

The order of the columns is as defined by the order of the returned items defined in the SELECT statement.

Security with Database Interaction

IBPM interacts with the database using the credentials defined in the security attributes defined in the WAS DataSource definition. It is vital that the tables being accessed have sufficient permissions granted for this userid. If not, an odd error message will be presented:

```
DSRA9110E: Statement is closed.
```

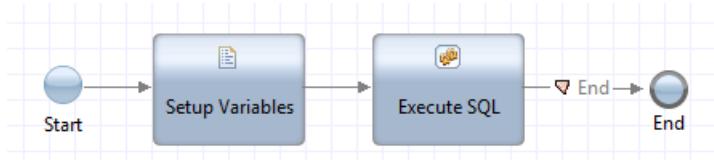
JDBC definitions needed to access databases

In order for a process or service to access a database, a JDBC definition must be made in the underlying WAS server that describes how to access the database. This is fully described at:

- Defining custom databases

Service to Insert a row

Following the recommendation of building wrapping services to perform function, here is an example of a service to insert a row into a DB. Imagine that the DB is called TESTDB and has a table called TAB1 which has columns A, B and C. We have a BO called BO1 that has fields A, B and C. The input to this service is an instance of BO1:



The setup variables step populates a String variable called sql which will contain the SQL code to execute and also it will populate the parms variable which is a list of SQL parameters.

```

tw.local.parms = new tw.object.listOf.SQLParameter();
tw.local.parms[0] = new tw.object.SQLParameter();
tw.local.parms[0].value = tw.local.bo1.A;
tw.local.parms[0].type = "VARCHAR";
tw.local.parms[0].mode = "IN";

tw.local.parms[1] = new tw.object.SQLParameter();
tw.local.parms[1].value = tw.local.bo1.B;
tw.local.parms[1].type = "VARCHAR";
tw.local.parms[1].mode = "IN";

tw.local.parms[2] = new tw.object.SQLParameter();
tw.local.parms[2].value = tw.local.bo1.C;
tw.local.parms[2].type = "VARCHAR";
tw.local.parms[2].mode = "IN";

tw.local.sql = "INSERT INTO KOLBAN.TAB1 (A, B, C) VALUES (?, ?, ?)";
  
```

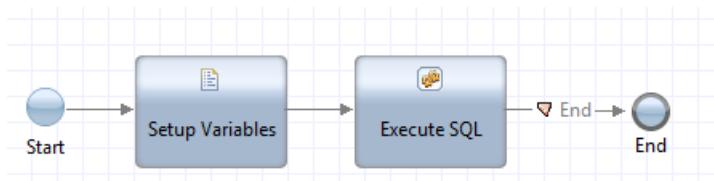
The Execute SQL parameters look as follows:

The screenshot shows the 'Properties' dialog for the 'Execute SQL' step. Under the 'Implementation' tab, the 'Data Mapping' section is selected. It displays the following mappings:

Input Parameter	Value	Type	Output Variable	Type
sql	tw.local.sql	String	results (ANY)	List of S...
parameters	tw.local.parms	List of SQLParameter		
maxRows		Integer		
returnType	"XMLElement"	String		
dataSourceName	"jdbc/TESTDB"	String		

Service to Delete a row

Again with the notion that we want to create re-usable services to perform work, here is one to delete a row. Imagine that the DB is called TESTDB and has a table called TAB1 which has columns A, B and C. We have a BO called BO1 that has fields A, B and C. The input to this service is an instance of BO1 and we want to delete the row where the row matches field A.



The setup variables looks as follows:

```
tw.local.parms = new tw.object.listOf.SQLParameter();
tw.local.parms[0] = new tw.object.SQLParameter();
tw.local.parms[0].value = tw.local.bol.A;
tw.local.parms[0].type = "VARCHAR";
tw.local.parms[0].mode = "IN";

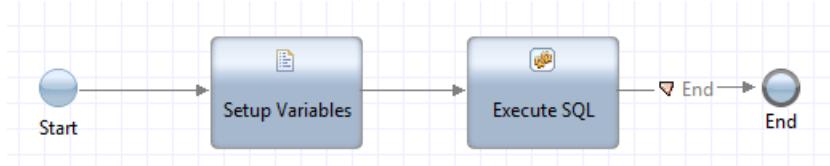
tw.local.sql = "DELETE FROM KOLBAN.TAB1 WHERE A=?";
```

The Execute SQL parameters look as follows:

The screenshot shows the 'Properties' tab of a service configuration. Under the 'Data Mapping' section, there are two main sections: 'Input Mapping' and 'Output Mapping'. In 'Input Mapping', several parameters are mapped: 'tw.local.sql' to 'sql (String)', 'tw.local.parms' to 'parameters(List of S...)', 'maxRows (Integer)' (with a checked checkbox), 'XMLElement' to 'returnType (String)', and 'jdbc/TESTDB' to 'dataSourceName (String)'. In 'Output Mapping', 'results (ANY)' is mapped to 'tw.local.results'.

Service to Update a row

Similar to the previous examples, this example illustrates a service to update a row in a table:



```
tw.local.parms = new tw.object.listOf.SQLParameter();

tw.local.parms[0] = new tw.object.SQLParameter();
tw.local.parms[0].value = tw.local.bol.B;
tw.local.parms[0].type = "VARCHAR";
tw.local.parms[0].mode = "IN";

tw.local.parms[1] = new tw.object.SQLParameter();
tw.local.parms[1].value = tw.local.bol.C;
tw.local.parms[1].type = "VARCHAR";
tw.local.parms[1].mode = "IN";

tw.local.parms[2] = new tw.object.SQLParameter();
tw.local.parms[2].value = tw.local.bol.A;
tw.local.parms[2].type = "VARCHAR";
tw.local.parms[2].mode = "IN";

tw.local.sql = "UPDATE KOLBAN.TAB1 SET B=?, C=? WHERE A=?";
```

This screenshot is identical to the one above, showing the 'Properties' tab with the 'Data Mapping' section open. It displays the same input and output mappings for the Execute SQL step, including the checked checkbox for 'maxRows (Integer)'.

Calling stored procedures

Let us first think about an initial sample stored procedure. Imagine a table that looks as follows:

OURTABLE

COL_A	COL_B
...	...

This is a simple table with two columns. Imagine that we have written a stored procedure that inserts a new row into the table. Duplicates are allowed. When a row is inserted, the number of COL_A rows that have the same COL_A value as the inserted row are returned.

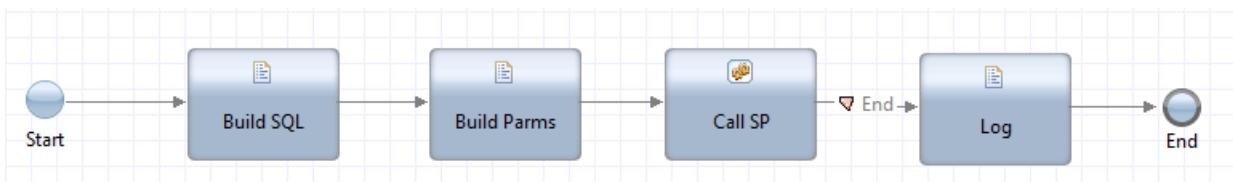
A procedure signature for this might be:

```
ourProcedure(in colAValue varchar(10), in colBValue varchar(10), out numDuplicates integer)
```

The code for this might be:

```
CREATE PROCEDURE OURPROCEDURE (
    IN COLAVALUE VARCHAR(10),
    IN COLBVALUE VARCHAR(10),
    OUT NUMDUPPLICATES INTEGER
)
P1: BEGIN
    INSERT INTO NULLID.OURTABLE VALUES (COLAVALUE, COLBVALUE);
    SELECT COUNT(*) INTO NUMDUPPLICATES FROM NULLID.OURTABLE WHERE COL_A = COLAVALUE;
END P1
```

In IBPM, we can then invoke the System Data toolkit provided Integration Service called "SQL Call Stored Procedure".



The passed in SQL was:

```
tw.local.sql = "CALL DB2ADMIN.OURPROCEDURE(?, ?, ?);"
```

and the parameters that were built were:

```
tw.local.sqlParms = new tw.object.listOf.SQLParameter();
tw.local.sqlParms[0] = new tw.object.SQLParameter();
tw.local.sqlParms[0].type="VARCHAR";
tw.local.sqlParms[0].mode="IN";
tw.local.sqlParms[0].value="ABC";

tw.local.sqlParms[1] = new tw.object.SQLParameter();
tw.local.sqlParms[1].type="VARCHAR";
tw.local.sqlParms[1].mode="IN";
tw.local.sqlParms[1].value="ABC";

tw.local.sqlParms[2] = new tw.object.SQLParameter();
tw.local.sqlParms[2].type="NUMERIC";
tw.local.sqlParms[2].mode="OUT";
```

For MS SQL Server, it appears that the SQL syntax may have to be:

```
{ call schema.procname(?) }
```

(Notice the addition of the curly braces.

See also:

- Cleaning/removing completed processes

Mapping IBM BPM Data types to DB data types

Database data types sometimes need to be mapped to/from IBM BPM data types. This section provides some notes on performing those tasks.

Dates/times

Consider a table which has a column of type TIMESTAMP. We may wish to perform a SQL select comparing against such a value. In IBM BPM we have the Date/TWDate entry. What if we wanted to compare against this?

For example:

```
SELECT * FROM MYTABLE WHERE ORDERDATE == tw.local.myDate
```

How can we achieve that?

SQL provides some conversion functions

- TIMESTAMP – example `TIMESTAMP('20131120190600')`
- DATE – example `DATE('2013-08-25')`
- TIME

These functions can take string parameters and convert the string to a comparable SQL query.

```
myDate.format("yyyy-MM-dd")
```

eg.

```
"SELECT * FROM MYTABLE WHERE ORDERDATE == DATE('" + tw.local.myDate.format("yyyy-MM-dd") + "')"
```

For a SQL TIMESTAMP, we can use:

```
sql = "..." + tw.local.myDate.format("yyyyMMddHHmmss") + "...";
```

See also:

- developerWorks – [DB2 Basics: Fun with Dates and Times](#) – 2003-08-23

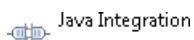
Java Integration

Java integration is the idea that we can invoke the services of custom written Java classes as part of the execution or a process. By dropping into Java, we have a powerful environment at our disposal. This is achieved with the Java Service implementation. This component can be found in an Integration Service.

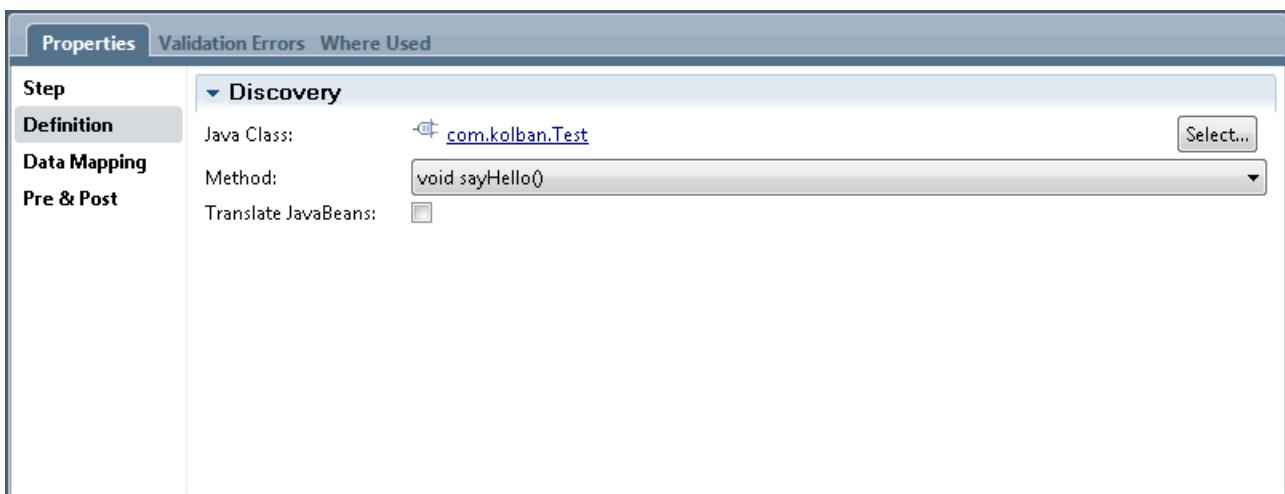
When added into the diagram, a Java service implementation looks as follows:



Note that this looks visually indistinguishable from a Web Service implementation. The Java Service is added from the palette from the component with the icon that looks like:



Once added, the properties sheet provides the options. The following image shows a properties page already completed.



The Select button allows the user to select the Java Class file that is to be called. This class must be part of a JAR file that has been defined as an "external file" to the Process Application (see: Adding managed files). Once the class has been chosen, methods in the class that may be called can be selected from the Method pull-down.

The Translate JavaBeans check box is used to describe how data being returned from the method call is to be handled. If checked, the data is serialized into an XML document and an XMLElement object is returned. This is useful if the returned Java Object is a JavaBean.

If the Translate JavaBeans box is not checked, the return data type MUST be one of the following types:

- java.lang.String
- java.util.Calendar
- java.lang.Long
- java.util.ArrayList
- java.lang.Integer
- java.util.HashMap
- java.lang.Short
- org.jdom.Document
- java.lang.Byte
- org.jdom.Element
- java.lang.Double
- com.lombardisoftware.core.XMLNodeList
- java.lang.Boolean
- com.lombardisoftware.core.TWOObject
- java.lang.Character

In addition to calling Java classes natively, JavaScript "LiveConnect" technologies could be used to call Java from within the context of a JavaScript environment however the use of LiveConnect is discouraged for performance reasons.

If a business object is passed, it appears to receive an instance of a class that implements the interface called `teamworks.TWOObject`.

The `TWOObject` class can be found in the `pscInt.jar` file found in the `<Install>/BPM/Lombardi/lib` folder.

The interface for `TWOObject` looks as follows:

```
public abstract interface TWOObject
{
    public abstract String getTWClassName()
        throws TeamWorksException;
```

```

public abstract Set<String> getPropertyNames();

public abstract Object getPropertyValue(String paramString);

public abstract void setPropertyValue(String paramString, Object paramObject);

public abstract void setPropertyValue(Map<String, Object> paramMap);

public abstract void removeProperty(String paramString);
}

```

Think of the `TWObject` class as a wrapper/holder for a Business Object. The names of all the properties contained in the object can be retrieved with the `getPropertyNames()` method. The value of one of these properties can then be obtained via the `getPropertyValue()` method.

If a list object is passed in as a parameter, the object is an instance of `teamworks.TWList`. The definition of this is:

```

public abstract interface TWList
{
    public abstract String getTWClassName()
        throws TeamWorksException;

    public abstract int getArraySize();

    public abstract void addArrayData(Object paramObject);

    public abstract void addArrayData(int paramInt, Object paramObject);

    public abstract Object getArrayData(int paramInt);

    public abstract void removeIndex(int paramInt);

    public abstract boolean isEmptyArray();

    public abstract List getUnmodifiableArray();
}

```

An interesting item to note is that IBPM has an internal class type called `com.lombardisoftware.core.TWObject`. This class implements **both** the `TWList` and `TWObject` interfaces. When a Java class is called, it is often this object type that is passed in both when a list and a complex object are used as actual parameters. The implication of this is that if one is dynamically processing input, we **can't** use the java "instanceof `TWList`" or "instanceof `TWObject`" to determine whether we have a list or a complex data type as both of these interfaces are implemented by a common class. Looking closer at the IBPM implementation, there is a `getType()` method which returns either `com.lombardisoftware.core.TYPE_ARRAY` or `com.lombardisoftware.core.TYPE_PROPERTIES`.

Any writing to `System.out.println()` will appear in the `SystemOut.log` file. This can be used for debugging and other tasks.

When a Java Class wishes to return a Business Object, it must create an instance of one. The `teamworks.TWObjectFactory` class can be used. It has a static method called `createObject()` which returns an instance of a `TWObject`. This method takes a parameter which is the type name of the Business Object. The type name is the BO's simple name. There is also a corresponding method to create a `TWList` object that will return a list of Business Objects. Again, the type of the Business Objects contained within the list must be specified.

When a Java class is exposed as an IBPM callable service, Process Designer knows only a little about it. Calling a Java method will simply show the following in the tooling:

This doesn't say anything about the parameters. What we would ideally like the result to be is:

A class called a "BeanInfo" class can be created which provides the meta data that can be used by PD to provide more info. The BeanInfo class must be created in the same package as the primary class and have the same name as the primary class appended with "BeanInfo". For example, if the primary class to be called by IBPM is "com.kolban.MyClass" then the BeanInfo class should be called "com.kolban.MyClassBeanInfo".

Next we will describe the creation of a suitable BeanInfo class.

First we create a new class that extends `java.beans.SimpleBeanInfo`. Next we override the function called `getMethodDescriptors`. This method returns an array of `MethodDescriptor` objects. The array will contain an object for each exposed method.

Imagine that we had a method in our class called "transform" that took as input a `HashMap` and a `String`. The following would present these as parameters we choose to call "Map" and "Template":

```
package com.kolban.bpm.velocity;

import java.beans.MethodDescriptor;
import java.beans.ParameterDescriptor;
import java.beans.SimpleBeanInfo;
import java.lang.reflect.Method;
import java.util.HashMap;

public class WLEVelocityBeanInfo extends SimpleBeanInfo {

    @SuppressWarnings("rawtypes")
    private Class beanClass = WLEVelocity.class;
    @SuppressWarnings("unchecked")
    @Override
    public MethodDescriptor[] getMethodDescriptors() {
        System.out.println("getMethodDesriptors called");
        try
        {
            Method method = beanClass.getMethod("transform", HashMap.class,
String.class);
            if (method == null)
            {
                System.out.println("Unable to find method.");
            }
        }
    }
}
```

```

        return null;
    }
    ParameterDescriptor parm1 = new ParameterDescriptor();
    parm1.setShortDescription("Map");
    parm1.setDisplayName("Map");

    ParameterDescriptor parm2 = new ParameterDescriptor();
    parm2.setShortDescription("Template");
    parm2.setDisplayName("Template");

    MethodDescriptor methodDescriptor = new MethodDescriptor(method, new ParameterDescriptor[]
{parm1, parm2});
    return new MethodDescriptor[] {methodDescriptor};
}
catch(Exception e)
{
    e.printStackTrace();
}

return super.getMethodDescriptors();
}
}

```

Experimentation seems to show that adding or changing a BeanInfo class may require a restart of Process Designer before the changes become apparent.

When working with Java Integration, I have found it useful to de-compile the classes provided by the product to examine how some of the supplied classes operate. Personally, I use the "Java De-compiler" called JD found here:

<http://java.decompiler.free.fr/>

The tool has both a stand-alone GUI as well as excellent integration with the Eclipse platform. Truly a top-notch effort here.

See also:

- Java source level Debugging
- [Integration Service](#)
- [Data Type – TWObject](#)
- [Data Type – XMLElement](#)
- Vimeo - [Video on Java Integration](#)
- Vimeo – [Video on Java debugging](#)
- [Java De-compiler package](#)
- DeveloperWorks - [Implementing Java integration components for IBM Business Process Manager V7.5.1](#) - 2012-06-13

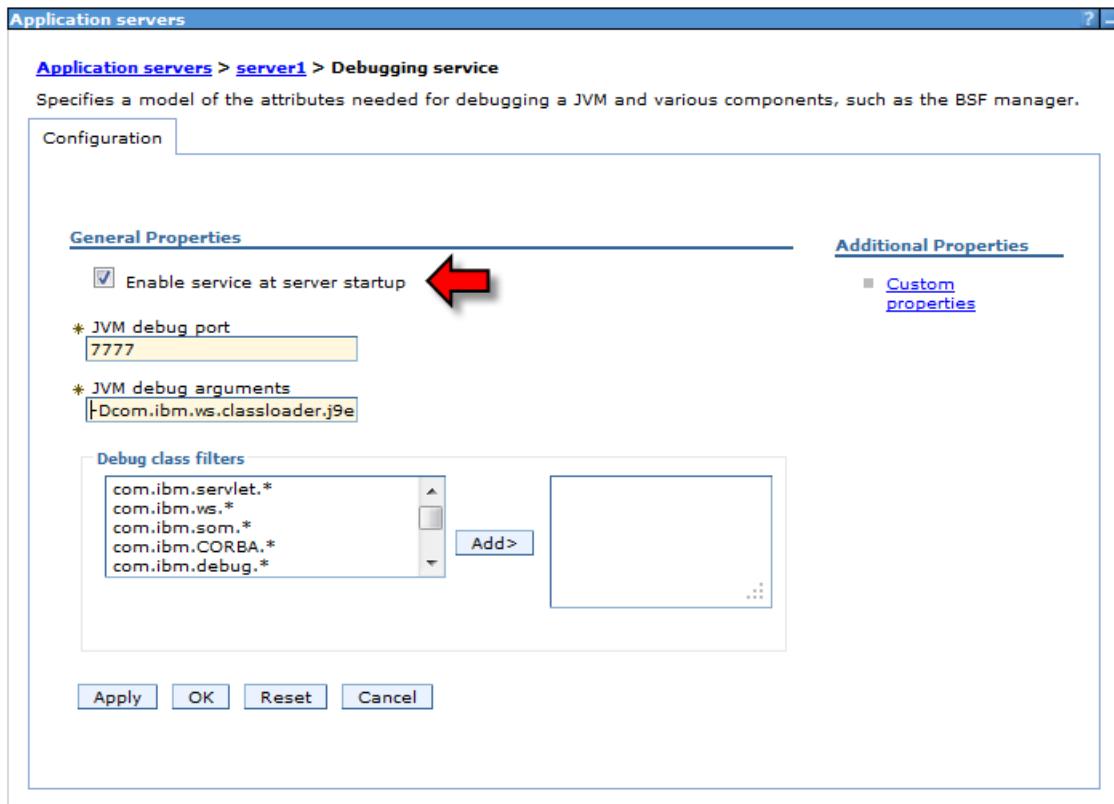
Java source level Debugging

When running Java code as part of the Process Application, we may wish to perform source level debugging against it. Fortunately, this is pretty simple to do. The WAS server hosting IBPM can have debugging enabled for it. Open a WAS Admin Console and navigate to Servers > Server Types > WebSphere application servers > <serverName> > Debugging Service.

Additional Properties

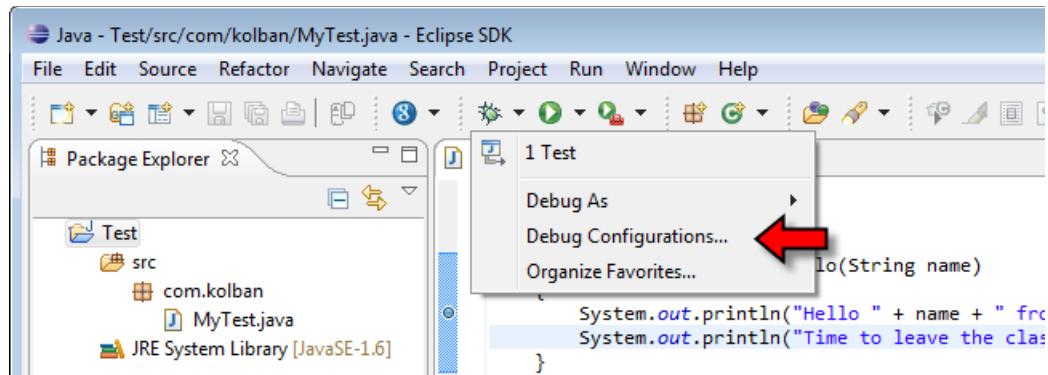
- [Class loader viewer service](#)
- [Endpoint listeners](#)
- **Debugging service** 
- [Thread pools](#)
- [Reliable messaging state](#)
- [Web server plug-in properties](#)

Within that area, there will be a check-box (initially unchecked) called "Enable service at server startup". Check that check box and restart the server. Take note of the "JVM debug port" value which defaults to 7777.

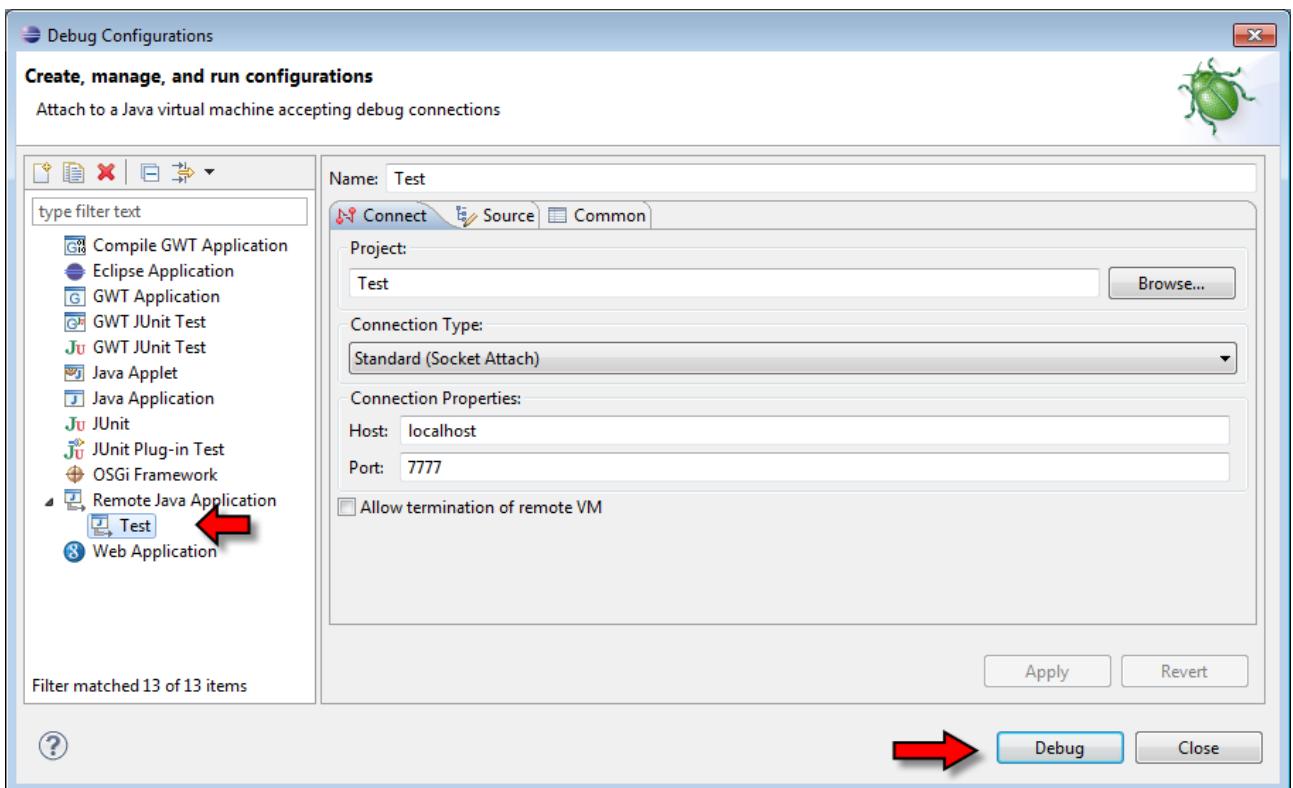


Restart the server for the change to take effect. Once the server has been bounced, it will start listening on port 7777 for incoming debugger attachment requests.

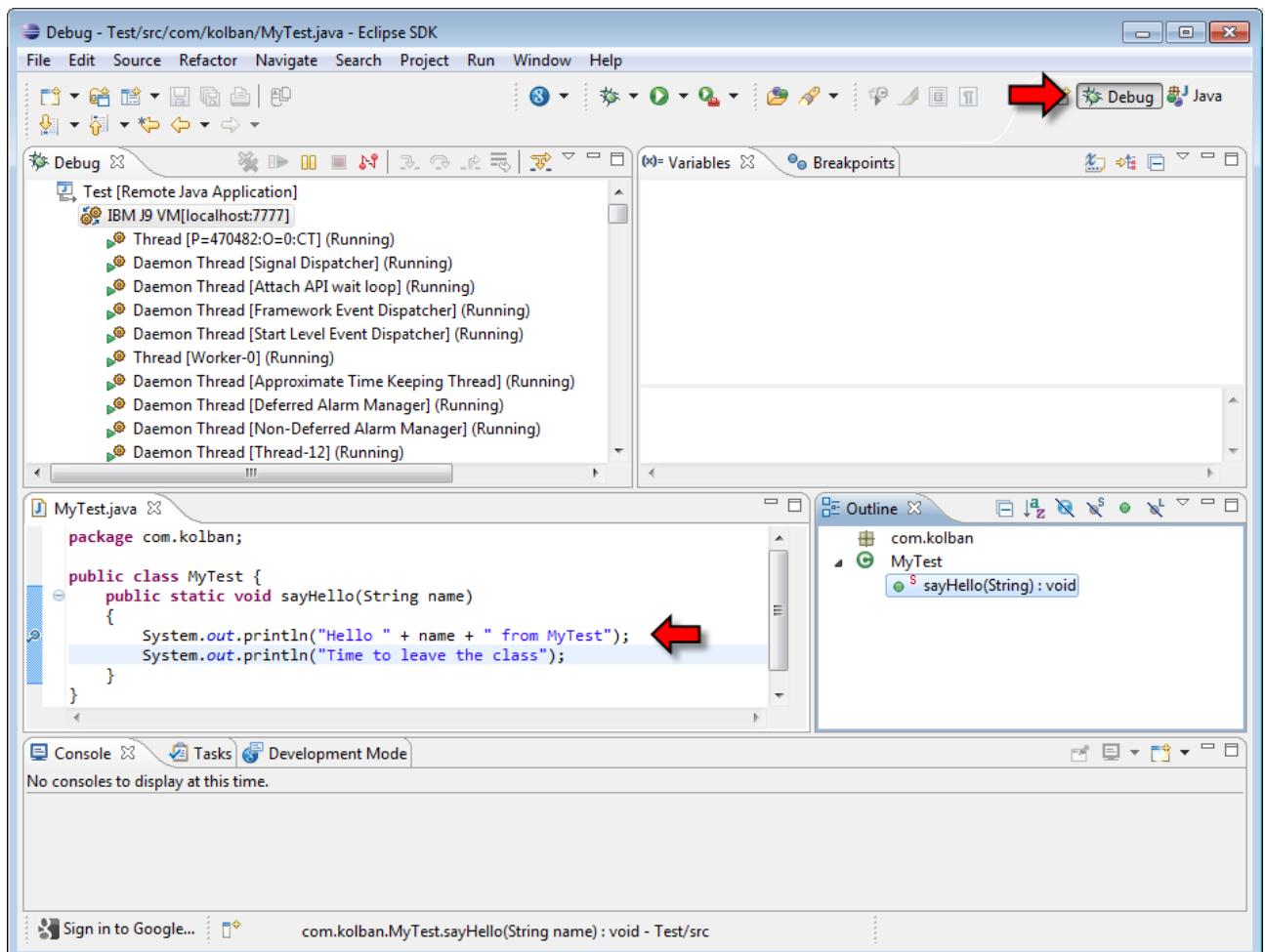
I am assuming that Eclipse is used for Java development. After saving a project which contains the source of the Java code, we can create a Debug Profile in Eclipse. From the debug menu, select "Debug Configurations...."



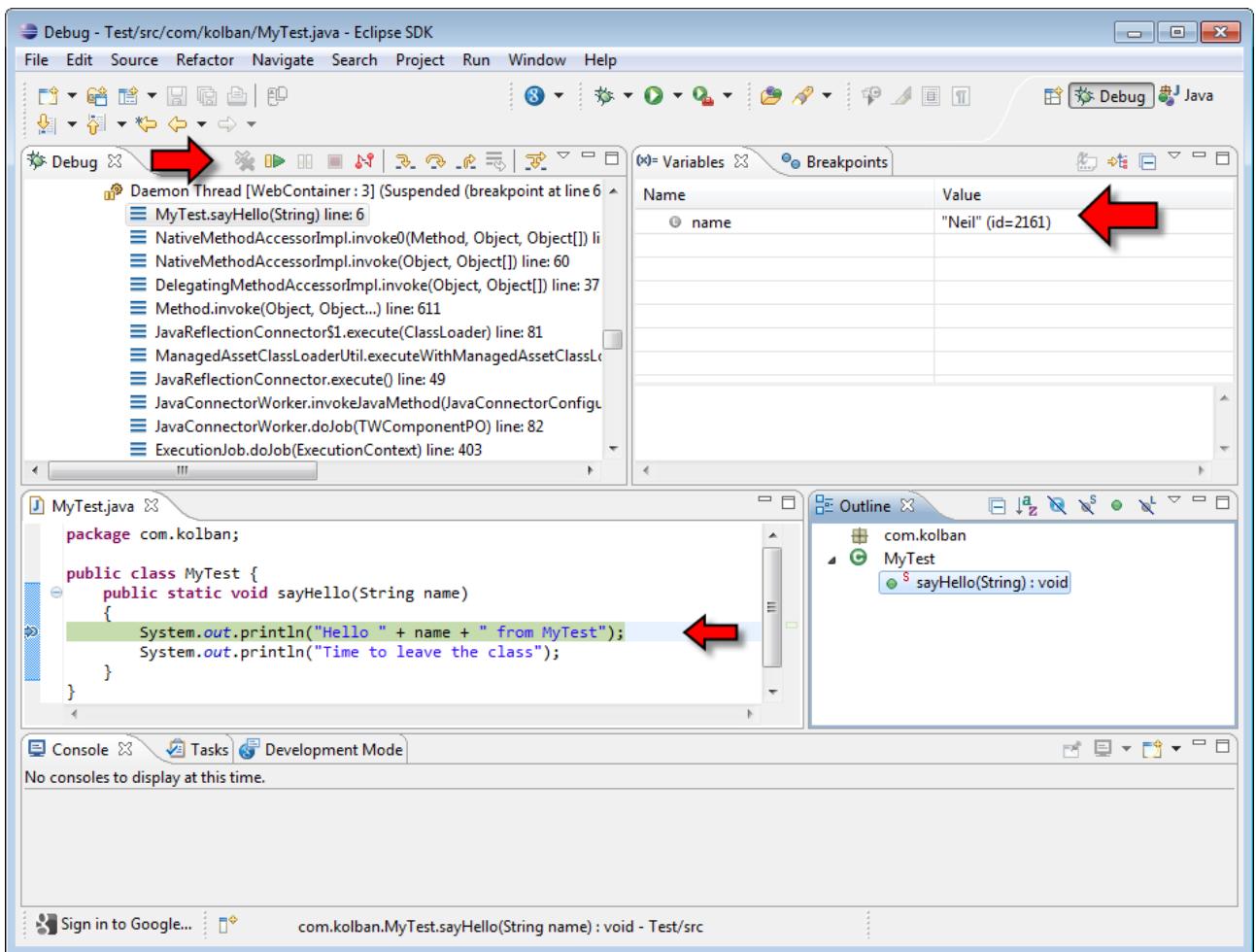
This will open the debug configurations menu. Select/create a new entry under Remote Java Application.



Next click the Debug button. Switch to the Eclipse Debug perspective and set a breakpoint in your source code of your project.



Now when an integration service is execute which invokes the code, a breakpoint will be reached and you can step through the code:



See also:

- Video: [Java Source Level Debugging](#)

eMail

IBPM has the ability to integrate with an external eMail provider. This can be used to originate an email during the execution of a process or receive an email for processing.

Sending an email

The System Data toolkit provides an integration service called "Send E-mail via SMTP". This can be invoked in a service through a Nested Service activity. The parameters to this service look as follows:

<input type="checkbox"/> Use default	"localhost"	smtpHost (String)
<input type="checkbox"/> Use default	"kolban@test.com"	to (String)
<input type="checkbox"/> Use default	"kolban@test.com"	from (String)
<input type="checkbox"/> Use default	"Test Message"	subject (String)
<input type="checkbox"/> Use default	"Hello there"	messageText (String)
<input checked="" type="checkbox"/> Use default		contentType (String)
<input checked="" type="checkbox"/> Use default		replyTo (String)
<input checked="" type="checkbox"/> Use default		cc (String)
<input checked="" type="checkbox"/> Use default		bcc (String)
<input checked="" type="checkbox"/> Use default		importance (String)
<input checked="" type="checkbox"/> Use default		attachmentFileNames ...

Parameter	Description
smtpHost	The hostname of the machine hosting the EMail server to which this new email should be delivered.
to	The account name of the user hosted by the email server to which the email should be delivered.
from	The identity of the person sending the email.
subject	A Subject message that will appear as the subject of the email in the receivers in-box.
messageText	The body of the email message.
contentType	This field describes how the content of the email is to be handled. Choices are "text/plain" (the default) which says that the body of the email is simple text. Another useful setting for this is "text/html" which says that the body of the email is to be treated as an HTML document and formatted by the email client appropriately.
replyTo	The address to which a reply to message should be sent.
cc	Additional recipients of the email (carbon copy).
bcc	Additional recipients of the email that are hidden from other recipients (blind carbon copy)
importance	
attachmentFileNames	

The service is implemented as the Java class called `teamworks.Mail`. This is contained within the "integration.jar" file. Examining the code for `teamworks.Mail`, it seems that:

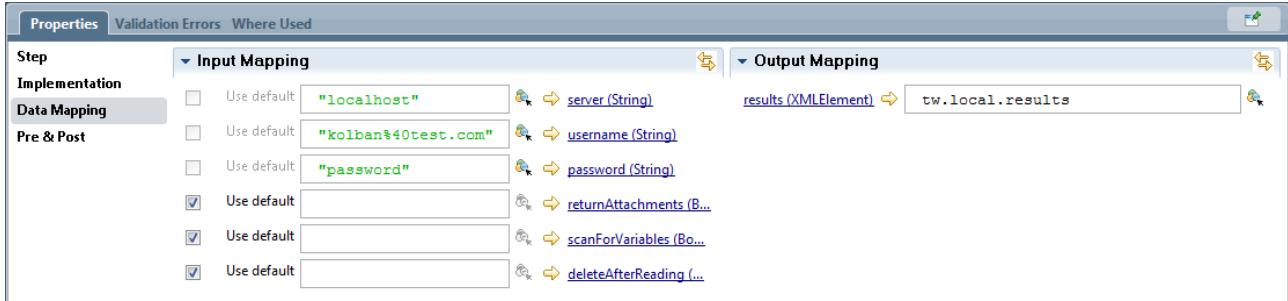
- No support for alternate port numbers are available
- No support for authentication is available

Receiving an email

Similar to sending an email, the System Data Toolkit provides a supplied service called "Read E-mail via POP" and a second service called "Read E-mail via IMAP". These services retrieve one or more emails from an email in-box and return them for processing. If there are no emails to be retrieved then the call returns immediately with no results. One common way of using this feature is to create a service which is scheduled to start every period of time. When the service is started, it polls the email in-box and if no emails are present, it ends and awaits its next start. If

an email is found, then a UCA is used to start a BPD instance to process the email. The start of the BPD is asynchronous and the email service can end before the BPDs have been processed.

The parameters for the service looks as follow:



Their description is:

Parameter	Description
server	The host-name of the email server against which we should attempt to read the email.
username	The account name of the user's email we wish to read. Take care with the use of "@" symbols. Consider coding "%40".
password	The password for the email account.
returnAttachment	??
scanForVariables	??
deleteAfterReading	After retrieving the email, should the email be deleted. The default is 'false'. If the email is left on the server, then the next read will re-read it.

The returned data is an XML data structure. When an email is retrieved, the format of the XML is as follows:

```
<resultSet recordCount="7" columnCount="-1">
<record>
  <column name="MSG_SUBJECT">test</column>
  <column name="MSG_TO">kolban@test.com</column>
  <column name="MSG_FROM">kolban@test.com</column>
  <column name="MSG_DATE">Mon Oct 25 10:50:22 CDT 2010</column>
  <column name="MSG_CC" />
  <column name="MSG_REPLY">kolban@test.com</column>
  <column name="MSG_NARRATIVE">Hello</column>
  <column name="MSG_ATTACHMENTS" />
</record>
...

```

To handle the result, here are some example XML processing fragments.

- To get the count of emails returned, use var.getAttribute("recordCount")

Here is an example of getting data from the emails:

```
var countOfEmails = Number(tw.local.results.getAttribute("recordCount"));
log.info(tw.local.results);
for (var i=0; i<countOfEmails; i++) {
    var subject = tw.local.results.xpath("/resultSet/record[" + (i+1) +
    "]/column[@name='MSG_SUBJECT']").item(0).getText();
    var from = tw.local.results.xpath("/resultSet/record[" + (i+1) +
    "]/column[@name='MSG_FROM']").item(0).getText();
    var body = tw.local.results.xpath("/resultSet/record[" + (i+1) +
    "]/column[@name='MSG_NARRATIVE']").item(0).getText();

    log.info(subject);
    log.info(from);
    log.info(body);
}
```

See also:

- Data Type – XMLElement

Send an email and receive a response

On occasion, a process solution may wish to send an email and then wait for a response to be received before carrying on. This would typically be seen when a process needs to interact with an external party who has no access to the task lists being maintained by BPM. For example, if a process is handling a request from a customer and needs some additional information or confirmation before continuing, email may be the most appropriate way to achieve that.

We have already seen that we can send an email very easily but how then do we block waiting for a response email to arrive?

The solution to this puzzle from a BPM perspective is to utilize a Message Intermediate Event which will block the process until an event is published. That event will be the arrival of a response email. What we will do is set up a scheduler which runs every period of time (say 15 minutes). When it runs, a service is invoked which polls the eMail system and retrieves each of the emails in a particular inbox. For each email received, the subject, sender and content may be examined. From this data, a correlation value will need to be found. An event will then be published using that correlation value.

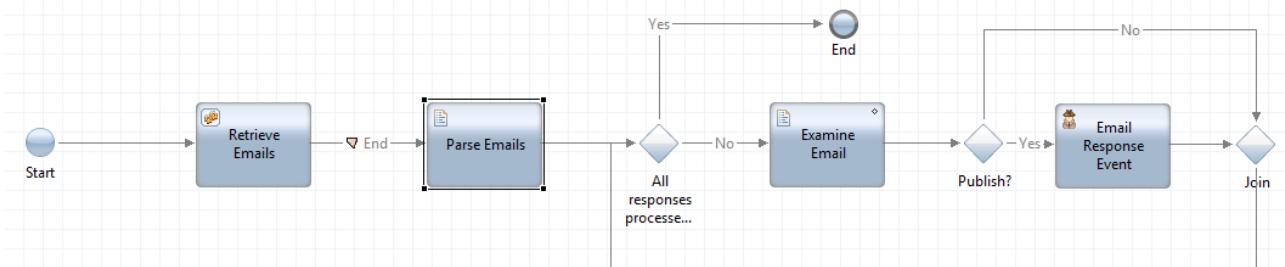
When the original email is sent, imagine that a unique "id" is added to the subject line. It may look like:

```
Subject: [yT1hP] - Request for information
```

It could alternatively be added into the body.

This "tag" should also be found in a response email. By matching the request tag to the response tag, we will have our correlation value.

Here is an example of a service which examines incoming emails and publishes events:



First the set of all the emails is retrieved. These emails are returned in an XML document which isn't that convenient to work with. We first convert each email into a Business Object instance defined as:

- requestId (String)
- success (Boolean)
- subject (String)
- body (String)
- sender (String)

At the time of initial parsing, the sender, subject and body portions are populated.

Next we start looping through each of the emails returned. We examine each email looking for a correlation value (in this example called "requestId"). In this example, the following code was used:

```
// Get the Request ID from the subject .... it will be found in '.... [xxxx] ...'
```

```

// Build a pattern which is:
// o Any number of characters followed by a '[' followed by any number of characters terminated by a
// ']' followed by any number of characters

tw.local.currentEmailResponse = tw.local.responses[tw.local.responseIndex];
tw.local.currentEmailResponse.success = false;
tw.local.readyForPublish = false;

var pattern = /.*\[(.*)\].*/;
var reqId = pattern.exec(tw.local.currentEmailResponse.subject);
if (reqId == null) {
    log.info("No match!");
} else {
    tw.local.currentEmailResponse.requestId = reqId[1];
    if (tw.local.currentEmailResponse.subject.match(/success/i) != null) {
        tw.local.currentEmailResponse.success = true;
    }
    tw.local.readyForPublish = true;
}
log.info("[Examine Emails]: RequestId: " + tw.local.currentEmailResponse.requestId + ", success = "
+ tw.local.currentEmailResponse.success );

// End of file

```

The core code in this is the use of a JavaScript regular expression to look for the tag in the Subject line.

This piece of wonderful JavaScript achieves that task:

```

var pattern = /.*\[(.*)\].*/;
var reqId = pattern.exec(tw.local.currentEmailResponse.subject);

```

It says "Match any string of characters up till a '[' and then find any string of characters up till a corresponding ']' and remember the result". Now search the Subject line using that pattern and retrieve the matched data between the '[' and ']' characters and call that the requested Id.

How we ask the email responder to reply is also a choice that we can make. We can ask them to simply reply to the email and add comments to the Subject or simply pass the body back with the event publish so that it can be examined by the blocked process when it wakes up.

See also:

- Message Intermediate Event
- Undercover Agents (UCAs)

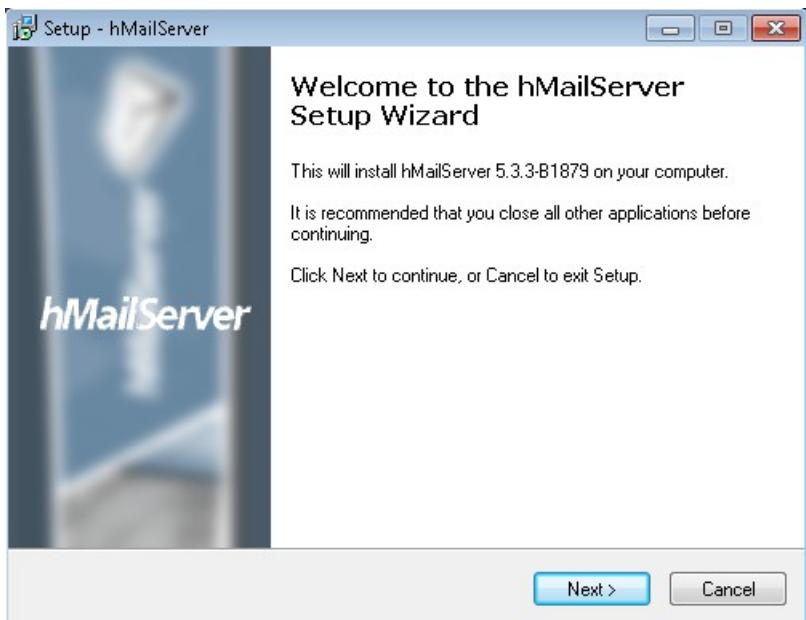
Installing hMailServer as a test EMail provider

When working with emails, it is useful to have a sandbox email server against which to test sending emails. A good and free email server package for this purpose is called **hMailServer** which is available for download:

<http://www.hmailserver.com/>

Installation of the package is very simple as shown in the following walk through.

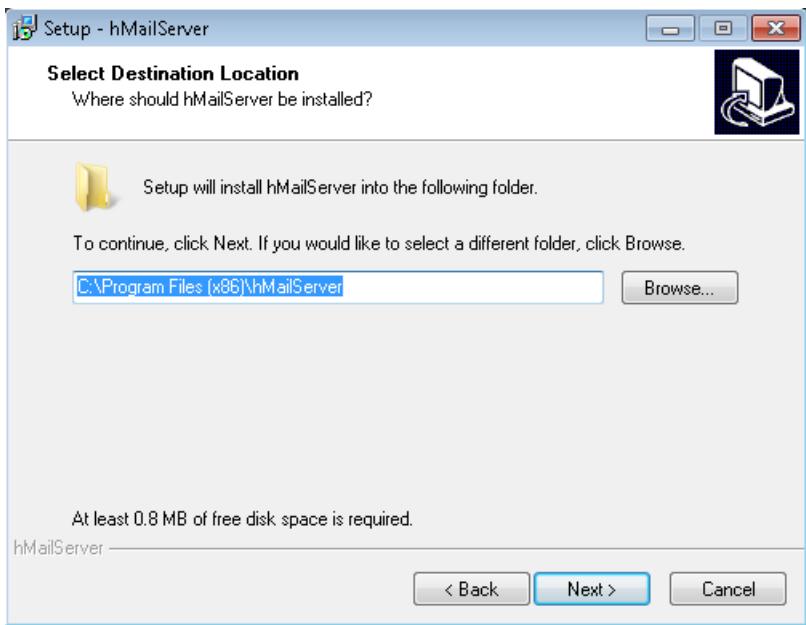
When the installer is first launched, we get the standard installer welcome message.



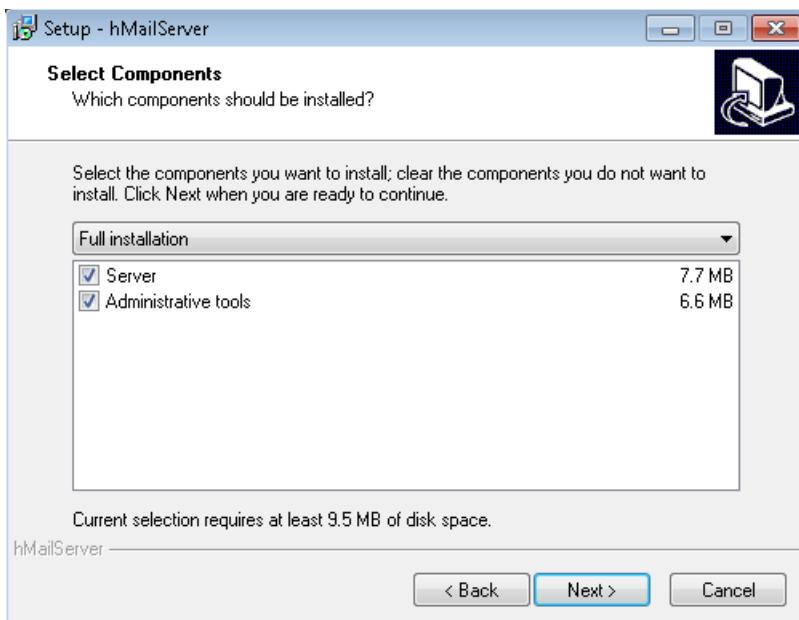
Next, read and review the license agreement and if you concur, accept and continue.



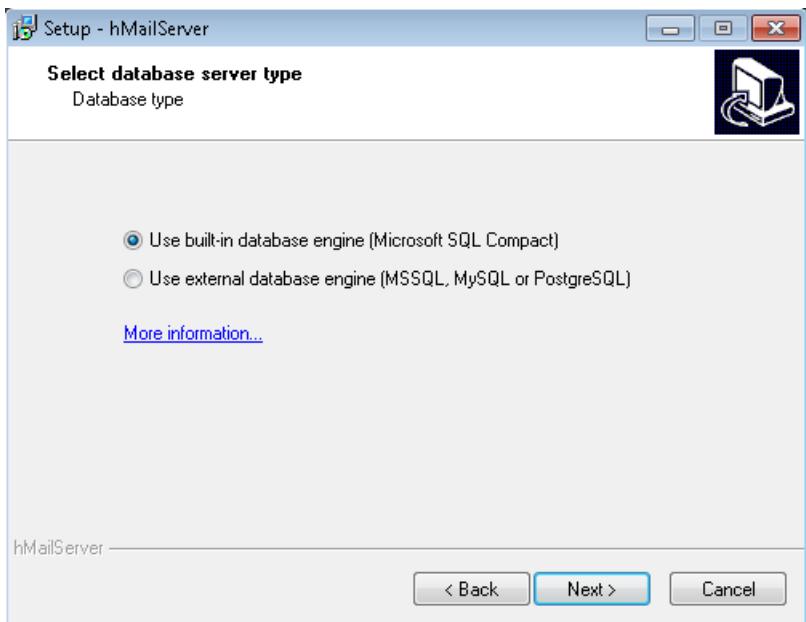
The EMail server will be installed on your local Windows file system. Here you get the opportunity to select the location where the product will be installed.



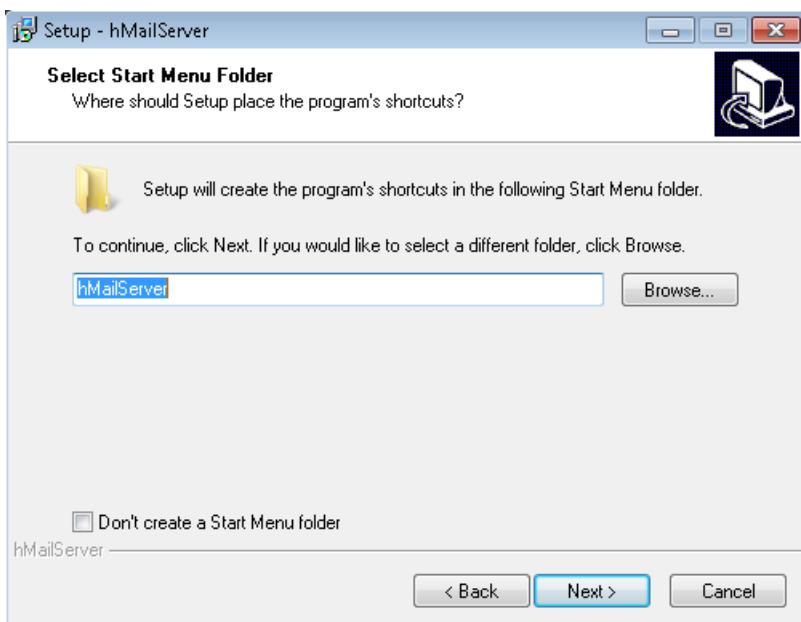
The EMail server has two components. One being the EMail server and the other being its basic Administration tools. Both should be installed.



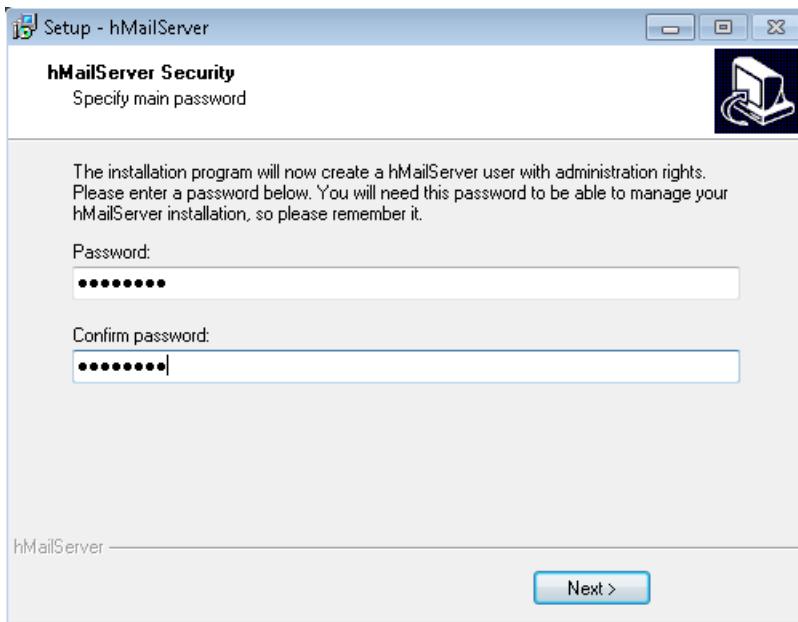
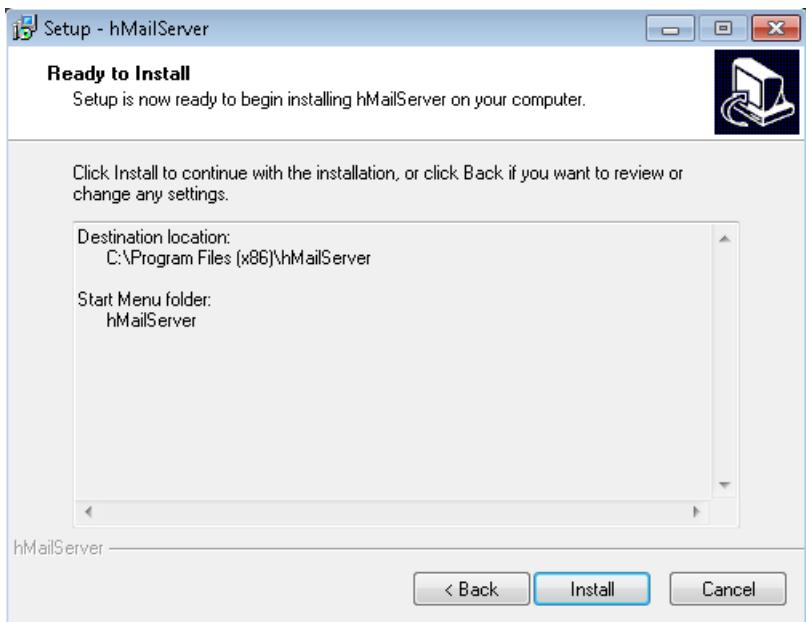
Even though we plan to use the EMail server for unit testing, the EMail server wants to know where to save its emails. Select the in-built Microsoft SQL database engine to minimize further configuration. The footprint is small and no explicit configuration is required.

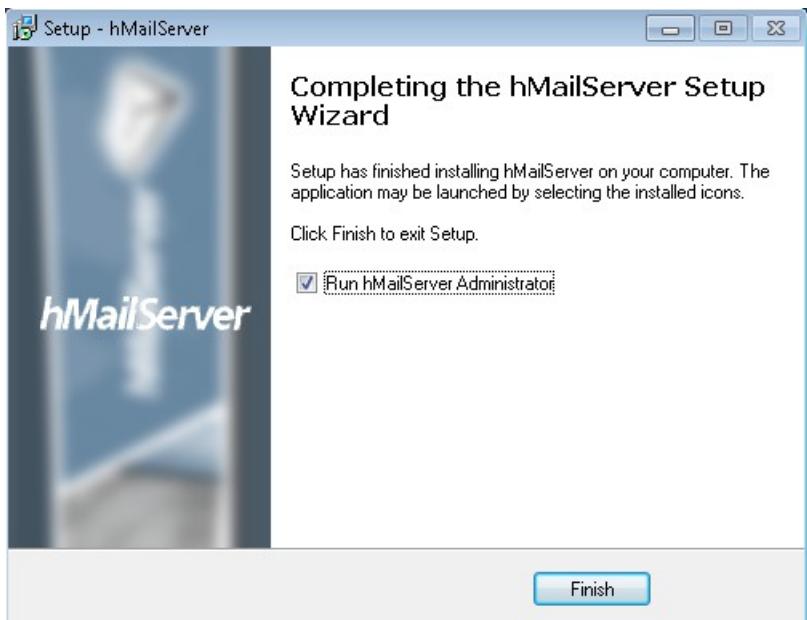


Finally, we are asked what we would like the EMail server to be known as in our start menu.

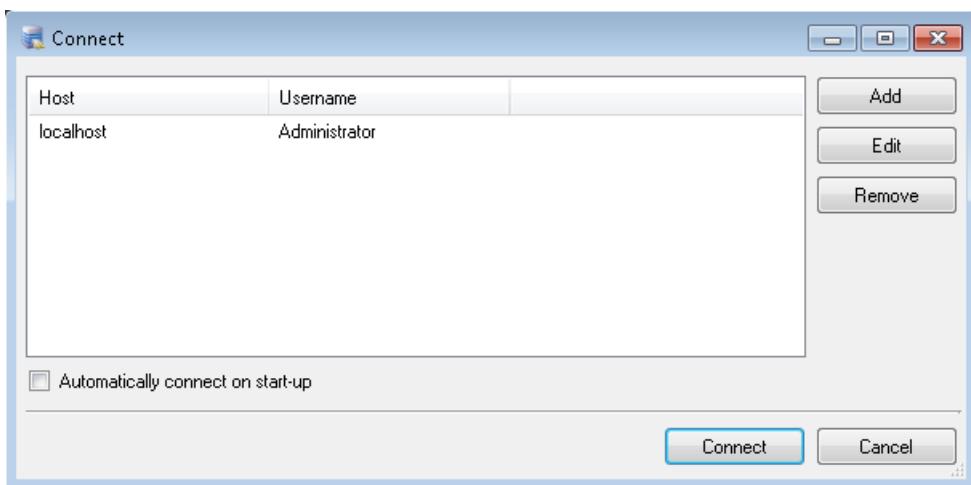


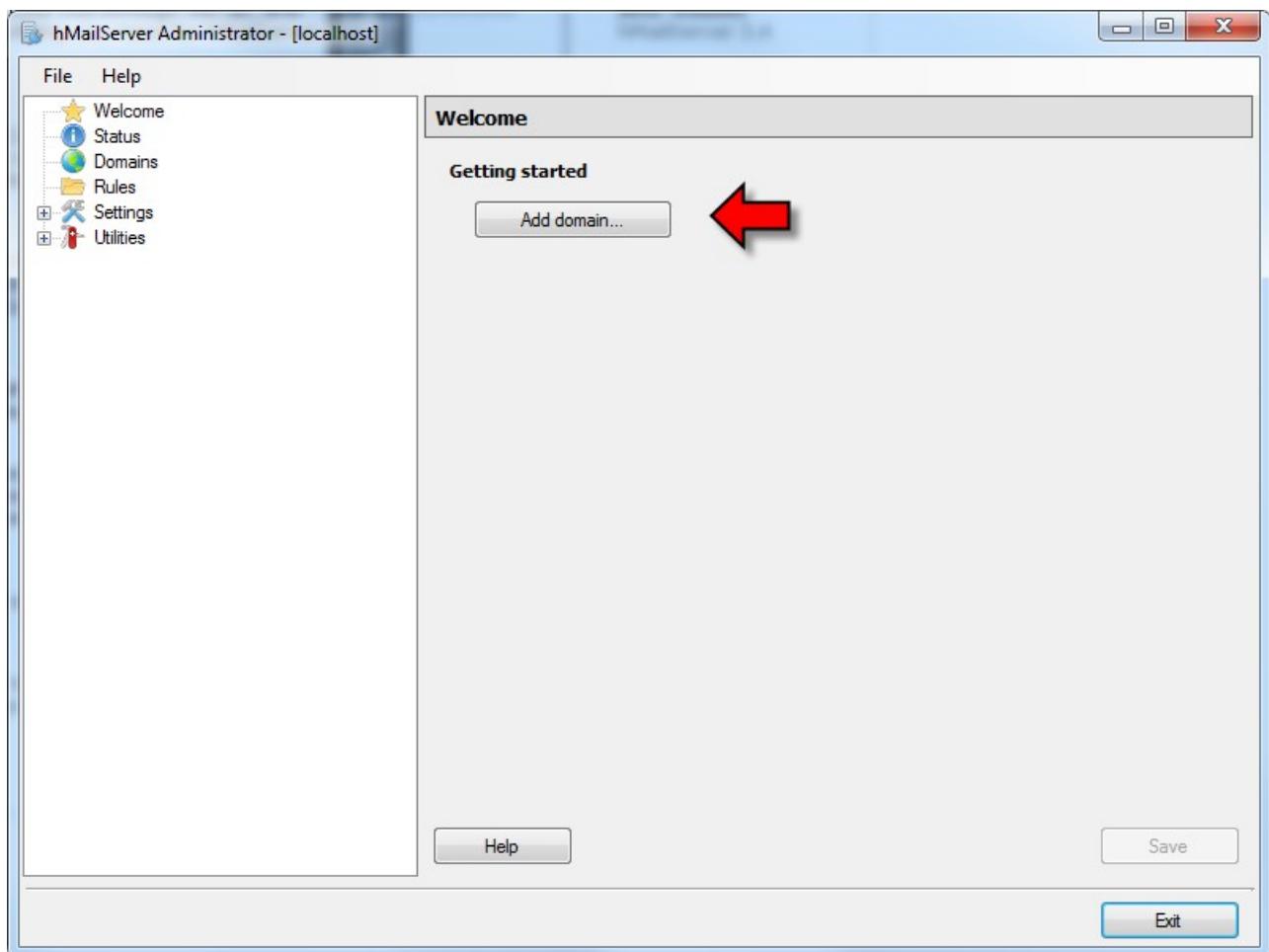
We are shown the final summary screen and then installation progresses.

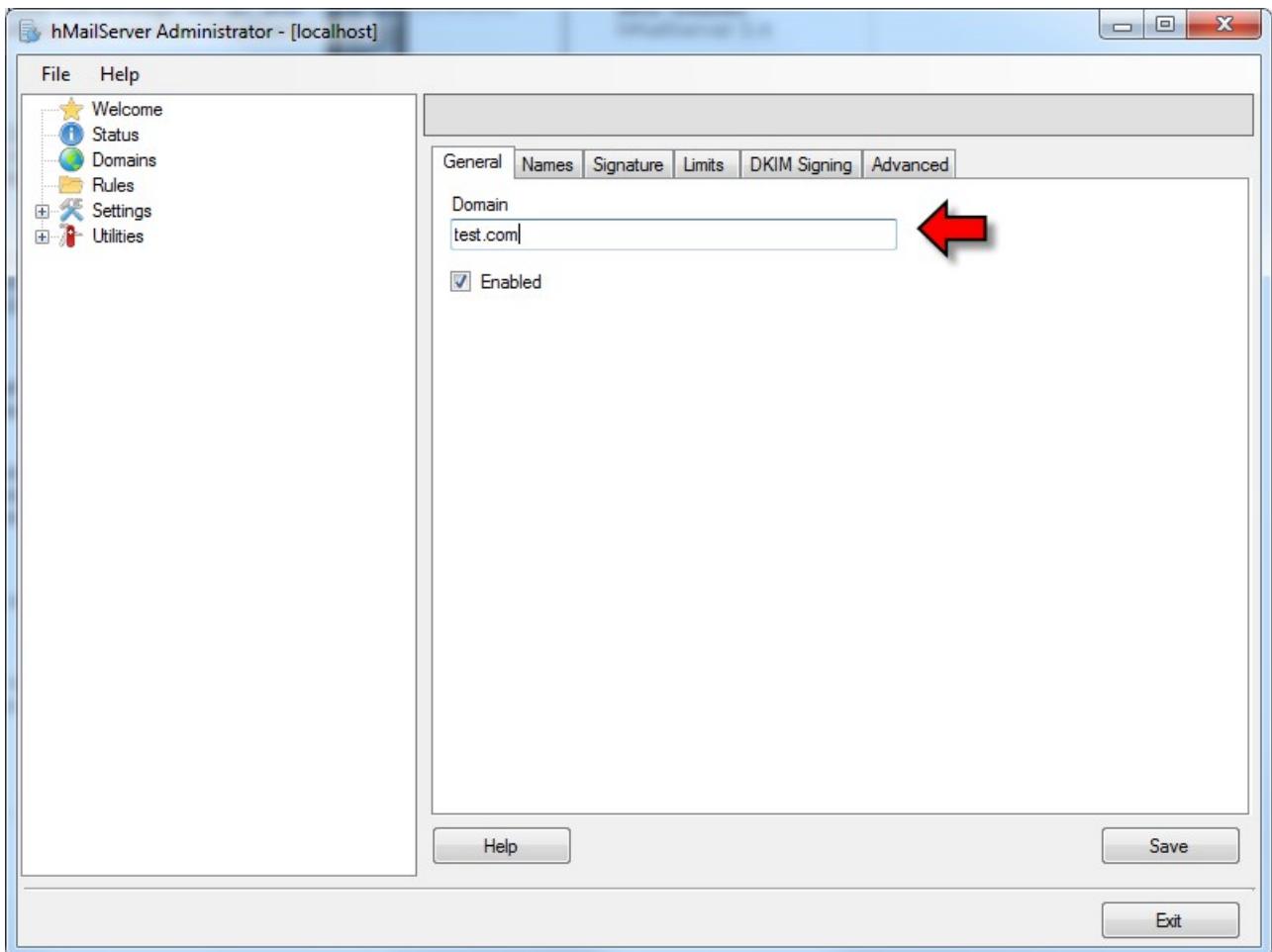




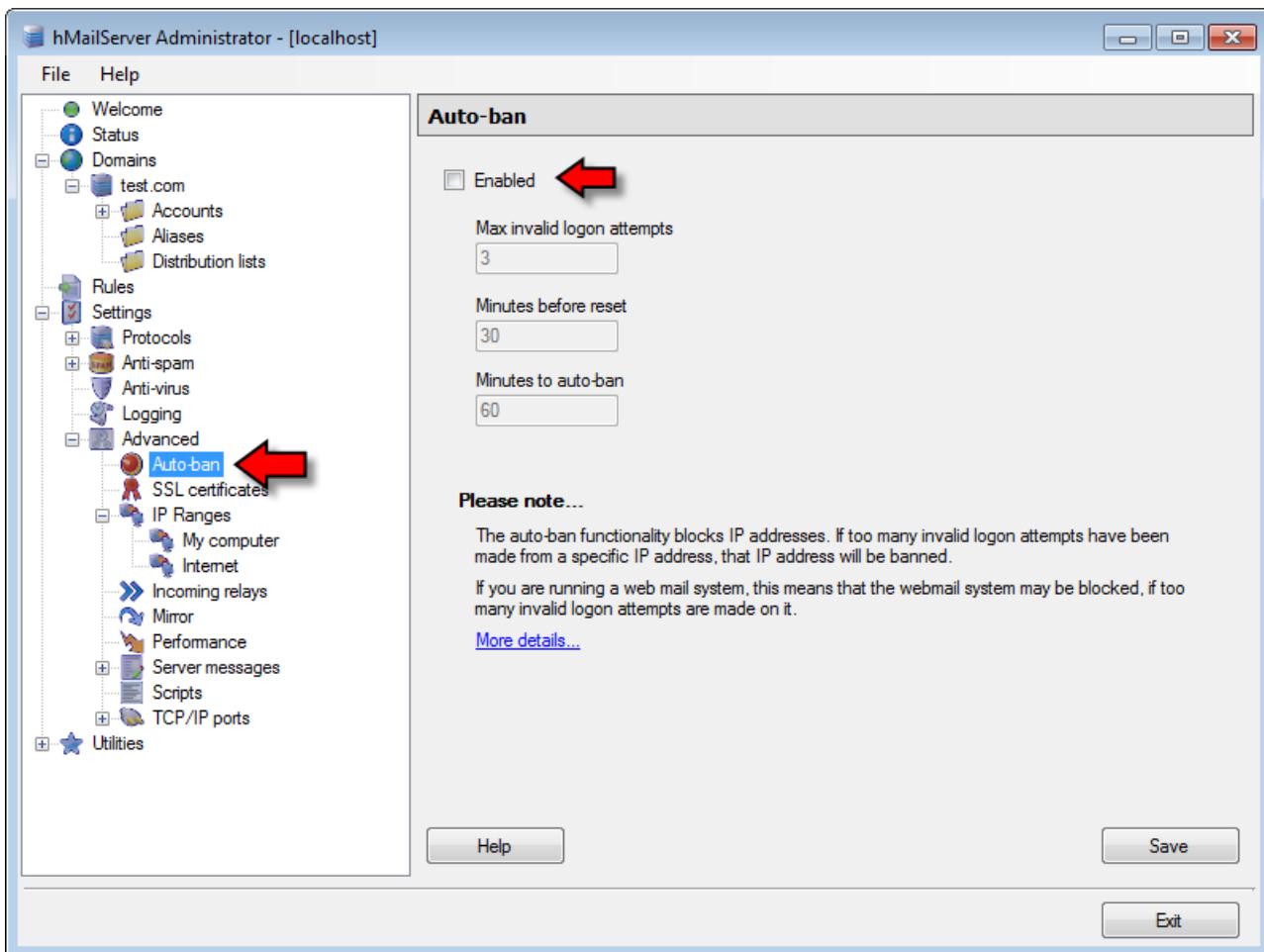
Once installed, the hMailServer administration console can be launched.







When testing with hMailServer, it is a good idea to switch **off** the auto-ban feature. Auto-ban will disable a host from being able to make mail connections because too many failed attempts have been recorded. To switch off auto-ban, uncheck the *enabled* check box in the Auto-ban settings:



If a ban has already happened, you will be able to undo the ban from the IP Ranges menu entry where an entry will be found from the host making the connection requests.

`javax.mail` can be debugged by adding the "mail.debug=true" flag in the System Properties, or so it is claimed on the web however I have not been able to find a recipe to switch this on.

The hMailServer and various other mail servers want to give userids the format "[xxx@yyy](#)" for login and access. Unfortunately, this seems to break the parsing of a userid. A work-around is to code as "xxx%40yyy". The "%40" is the hex escape code for the "@" symbol.

WPS and SCA

Prior to the release of IBPM 7.5, IBM had two separate BPM products. One of those was called WebSphere Process Server (WPS). WPS was a heavy utilizer of the SCA technologies. There may be occasions where an IBPM 7.5 environment may wish to interact with a legacy WPS environment.

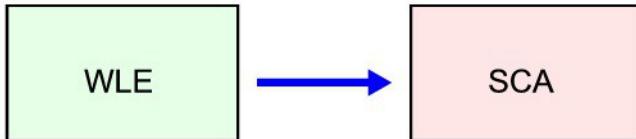
See also:

- DeveloperWorks - [Integrating WebSphere Process Server V7 and Lombardi Teamworks](#) – 2010-06-30

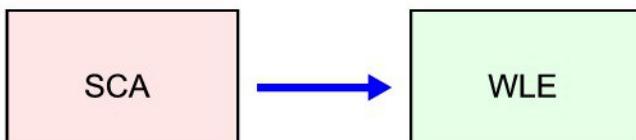
Asynchronous Invocation of an SCA Module

Consider a IBPM process that wishes to invoke an instance of an old WPS or WESB hosted module. That module might take some time to respond. Conceivably, it could take days if a

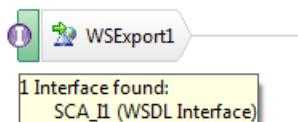
Human Task in WPS is utilized. As such, we can't afford to keep a connection from IBPM to SCA open for that length of time. One possible solution to our problem is leverage two one-way invocations. One from IBPM to SCA to start the module and later, when the SCA module has completed its work, one further invocation back from SCA to IBPM to allow the waiting IBPM hosted process to continue.



Later ...



In this story, we will assume that the SCA Module exposes an interface called SCA_I1 which is a one-way interface that has an input variable called in1 of data type BO1 (a Business Object).



The interface looks like:

The screenshot shows the configuration and operations for the SCA_I1 interface. Configuration details include:

Name	SCA_I1	Refactor name
Namespace	http://WLEtoSCA/SCA_I1	Refactor namespace
Binding Style	document literal wrapped	Change binding style to document literal non-wrapped More...

The Operations section shows an operation named "op1" with an input parameter "in1" of type "BO1".

Once the module has been deployed, we now find that we have an exposed Web Service that when invoked, will start an instance of that module. Looking at the binding of the SCA Export, we find that we have the WSDL URL. However, WPS is pulling a trick on us here. If we added a "?wsdl" to that URL an HTTP redirect would occur which would break IBPM. So what we do is manually add a "?wsdl" to the URL in a browser and get the **real** URL to the WSDL.

Export: WSExport1 (Web Service Binding)

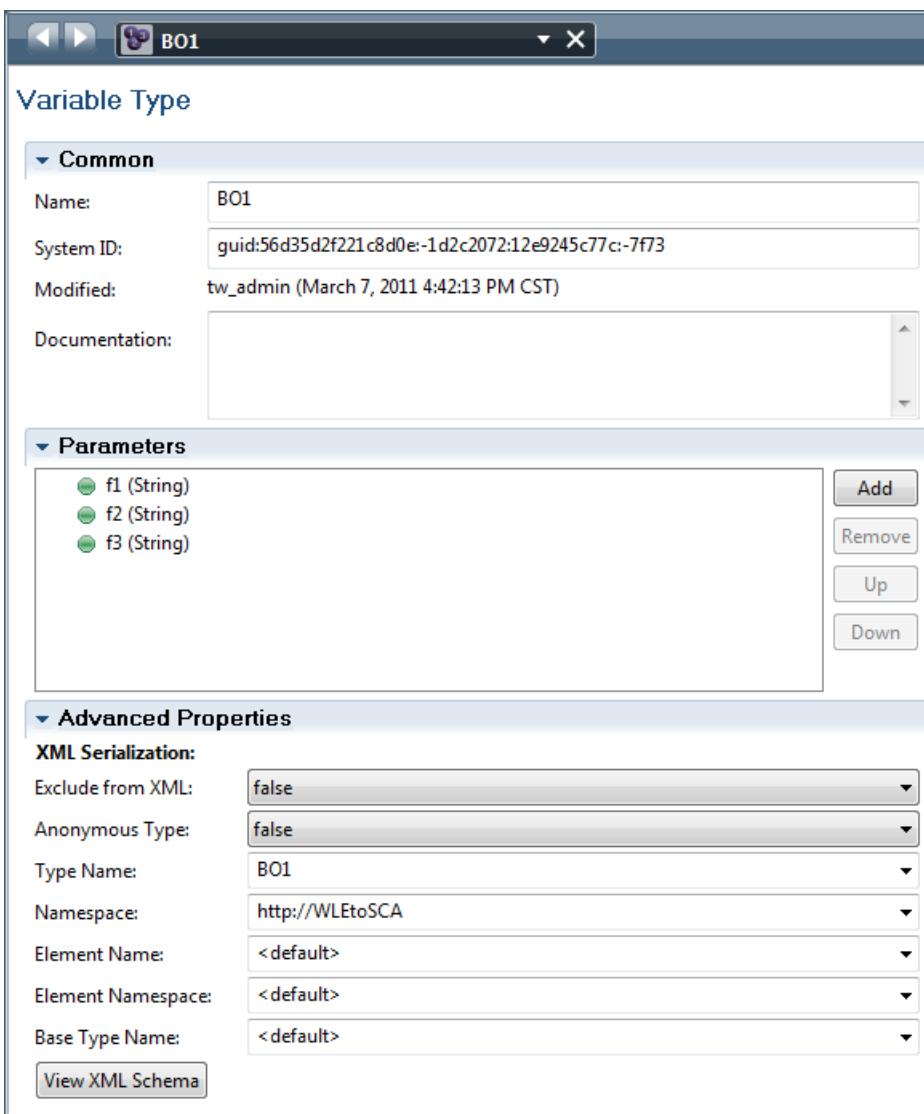
Description	Transport:	SOAP1.1/HTTP
Details	Address: http://localhost:9080/WLEtoSCAWeb/sca/WSExport1 	
Binding	Port:	WSExport1_SCA_11HttpPort <input type="button" value="Browse..."/>
Policy Sets	Service:	WSExport1_SCA_11HttpService
JAX-WS Handlers	Namespace: http://WLEtoSCA/SCA_11	
Propagation		
All Qualifiers		

becomes (when adding a "?wsdl" on the end)



The screenshot shows a Mozilla Firefox browser window. The address bar contains the URL http://localhost:9080/WLEtoSCAWeb/sca/WSExport1/WEB-INF/wsdl/WLEtoSCA_WSExport1.wsdl. A red arrow points to the end of the URL, specifically to the ".wsdl" extension.

After running through the IBPM Web Service integration steps and importing the data types, we end up with a new data type called BO1 which has the properties we desire:



The screenshot shows the configuration screen for the BO1 variable type. The window title is "BO1".

- Common** tab (selected):
 - Name: BO1
 - System ID: guid:56d35d2f221c8d0e:-1d2c2072:12e9245c77c:-7f73
 - Modified: tw_admin (March 7, 2011 4:42:13 PM CST)
 - Documentation: (empty text area)
- Parameters** tab:
 - Defined parameters: f1 (String), f2 (String), f3 (String).
 - Action buttons: Add, Remove, Up, Down.
- Advanced Properties** tab:
 - XML Serialization:
 - Exclude from XML: false
 - Anonymous Type: false
 - Type Name: BO1
 - Namespace: <http://WLEtoSCA>
 - Element Name: <default>
 - Element Namespace: <default>
 - Base Type Name: <default>
- View XML Schema** button (located at the bottom left of the Advanced Properties section).

and we also find that if we can see the settings/operation in the Implementation Web Service tab:

Properties Validation Errors Where Used

Step

Implementation

Header WSDL URI:

Security Protected WSDL:

Data Mapping Username:

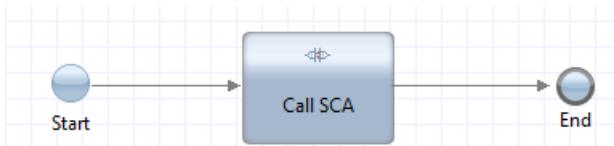
Pre & Post Password:

Operations:

Override Endpoint Address:

Endpoint Address URL:

If we now wire a test service as follows:



and invoke it, we find that the SCA module has indeed been called:



Where the implementation of LogCall looks like:

```

public void op1(DataObject in1) {
    System.out.println("Op1 called: " + in1);
}
    
```

We find the following in the output log:

```

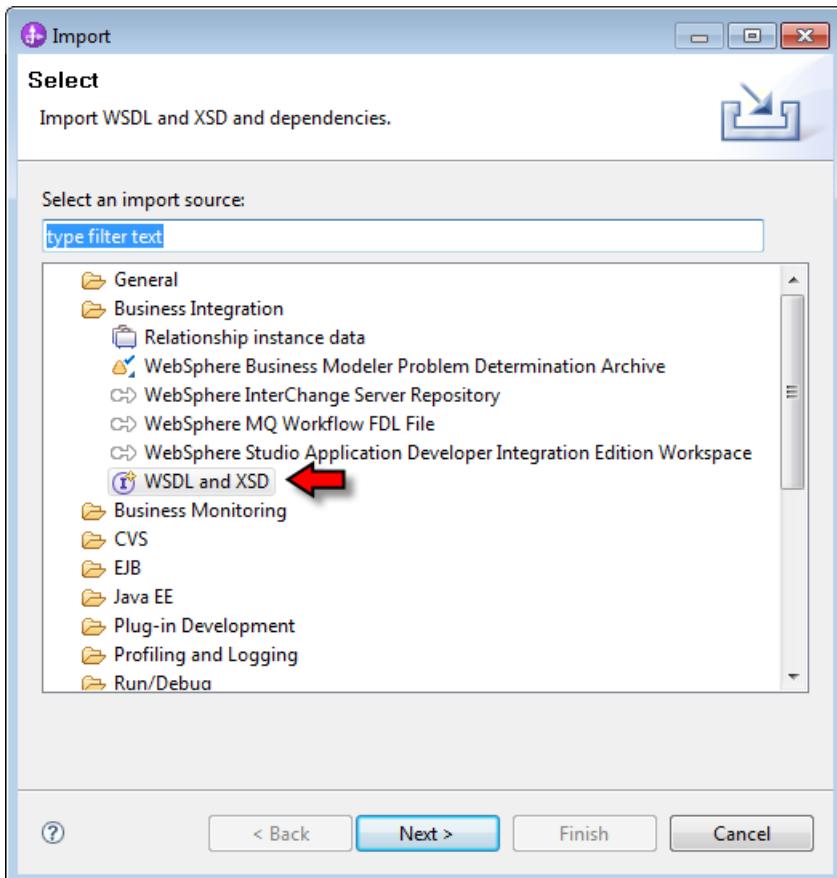
[3/7/11 16:57:13:326 CST] 0000007c SystemOut      O Op1 called: BusinessObject:
BO1@41c441c4 (f1=f1 val, f2=f2 val, f3=f3 val)
    
```

Hooray. This is half the challenge. We have now called SCA from IBPM. Now we need to wait for a response and have SCA call us back. Here we will assume that IBPM exposes a Web Service interface. This one will be invoked by the SCA module when the response is ready to be sent. Exposing a IBPM service is described elsewhere.

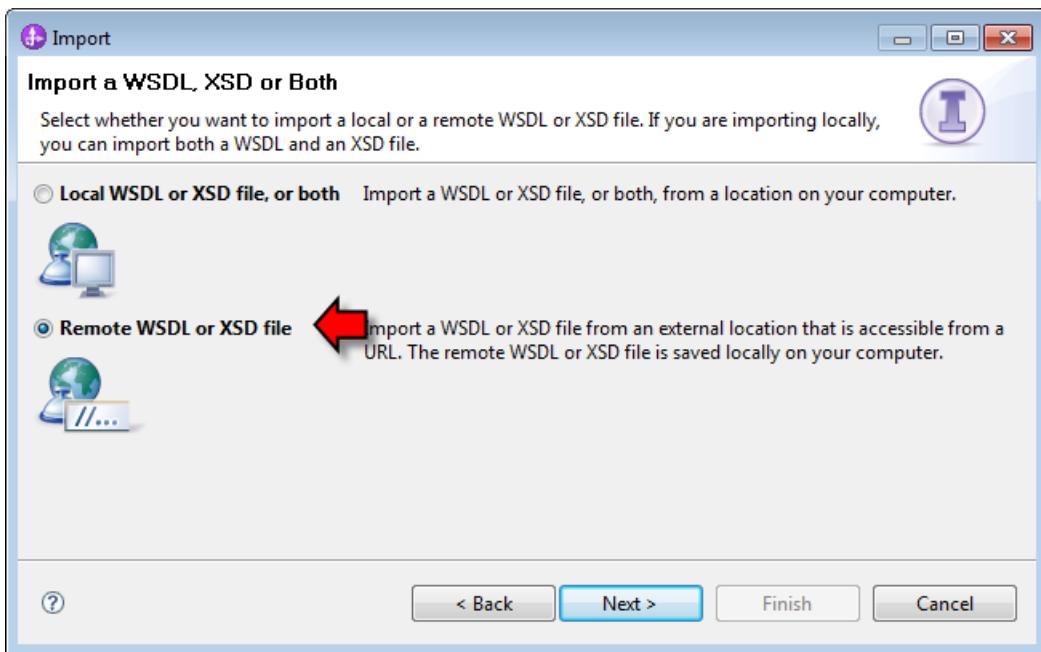
Once we have configured IBPM to expose a Web Service, we can import that Web Service definition into the SCA module. In the IBPM Web Service definition, we find the WSDL URI for the exposed service. We need to copy that URI into the copy buffer.

The screenshot shows the 'Web Service Definition Usage' dialog in WID. The 'Common' section contains fields for Name (SCAResponse), System ID (guid:56d35d2f221c8d0e:4cca349a:12e9246edb9:-7fc), Modified (tw_admin (March 7, 2011 5:09:01 PM CST)), and Documentation. The 'Behavior' section includes a 'Protected' checkbox and a 'WSDL URI' field containing the URL <http://win7-x64:19086/tea...EST1/SCAResponse.tws?WSDL>.

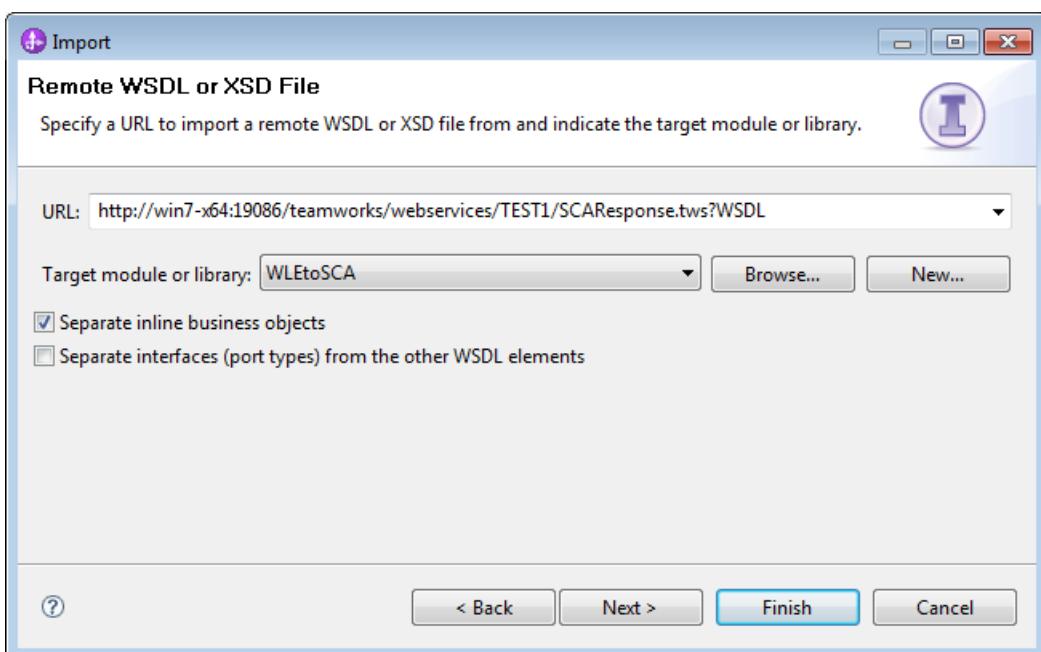
In WID, select the Imports and from there navigate to Business Integration > WSDL and XSD



Select the source of the WSDL to be imported as a remote artifact existing on the net.

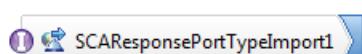


Paste the URL that was selected from the IBPM Web Services component. At this point you can select Finish and the artifacts will be created.



At the culmination, a new Web Services interface definition will be found in your project. Any data types associated with that interface will also be created.

An SCA Import component can then be used to call the IBPM web service.



Remember that correlation must be utilized in this solution. There **must** be some data in the returned response that allows IBPM to know *which* instance of the BPD being blocked by an Intermediate Message Event to wake up. A suitable BPD looks as follows:



The Intermediate Message Event is triggered by a UCA which is itself invoked by the SCA originated Web Service call. The Intermediate Message Event's configuration names a UCA supplied parameter that must match a correlation value in the originating BPD.

Properties Validation Errors Where Used

Step

Implementation

Pre & Post

Message Trigger

Attached UCA: [UCA SCA](#)

Condition:

Consume Message:

Durable Subscription:

UCA Output Correlation

- [ucaDataOut \(BO2\)](#)
- [correlationValue \(St...\)](#)

Process Scheduling with Job Scheduler

On occasion it may be required to start an IBPM process at configurable times of the day or week. To automate this task, a scheduler must be employed. Scheduling is the ability to set up recurring dates and times and have work kicked off at those times. An item of work that is to be executed is termed a job.

IBPM has an in-built ability to schedule work (see: Schedule initiated UCAs). In this section we will look at some alternatives that may add additional function.

Although WebSphere Application Server (WAS) has a built in scheduler, it is designed primarily for programmers to code against and does not easily lend itself to a generic business level job scheduling mechanism.

Examining the Open Source offerings, an Open Source product called Job Scheduler is available that provides a powerful and rich framework for job scheduling. This includes a rich user interface to perform administration of the majority of scheduling tasks at a very high level. Job Scheduler can be combined with IBM's IBPM solutions to provide a variety of techniques to allow IBPM based jobs to be scheduled and initiated.

Job Scheduler can be download from:

<http://jobscheduler.sourceforge.net/>

Job Scheduler is particularly flexible in the kinds of application types it can start. Using the product you can configure a Job and then when the start time is reached, that Job is initiated. The choices of application types that can be started include Java classes, native commands, scripts and much more.

The goal of this section is not to document Job Scheduler; the documentation available at the product web site seems more than sufficient for that task. Instead, this paper focuses on using Job Scheduler to initiate IBPM based activities.

The nature of a IBPM Job

When we look at IBPM and ask "what is it that we wish to periodically start?" the answer is a IBPM General Service. If this interface is exposed through a Web Service, then the nature of the Job that is to be started from the Job Scheduler will be the invocation of that interface through a SOAP/HTTP request call.

Other bindings between Job Scheduler and IBPM can also be employed including REST, JMS, MQ and others.

The Job Scheduler Java API

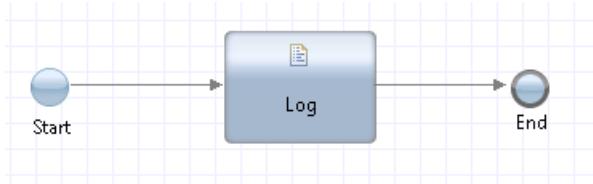
Job Scheduler exposes a set of Java classes that can be used to build a custom Java program that is tailored to call IBPM. Key amongst these classes is one called `sos.spooler.Job_impl`. This class can be extended by a custom user written class. In the custom class, one can then implement a method called `spooler_process()`. When Job Scheduler wishes to invoke a Java based Job, the result will be a call to this method. In the body of this method, one can then code arbitrary Java calls to achieve the function that the Job is to perform. In our case, this will result in the invocation of a IBPM process through a Web Service binding.

An illustrative solution

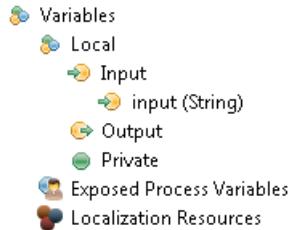
To demonstrate how these tasks can be achieved, we will work through a simple scenario. We will assume that you have downloaded and installed the latest version of Job Scheduler. This can be found at the Job Scheduler web site:

<http://jobscheduler.sourceforge.net/>

We start by creating a General Service in PD that exposes the input to the process that we may want. The General Service looks like:



with variables defined as:



This service does nothing more than log it was called but in reality could invoke a UCA to start a BPD instance.

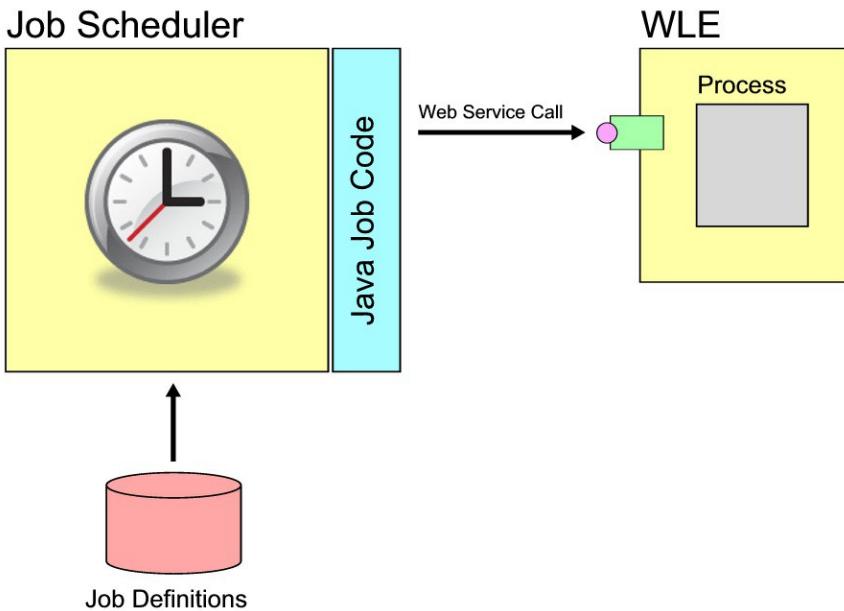
Next we define a Web Service resource to expose starting the General Service as a Web Service. This also defines the operation associated with the service. In our example, the operation is called "startProcess".

The screenshot shows the configuration of a Web Service named 'WS1'. The 'Common' tab displays the service name 'WS1' and the modified date 'tw_admin (October 5, 2010 12:11:41 PM CDT)'. The 'Behavior' tab shows the WSDL URI as <http://win7-x64:19086/t...ces/STESTS/WS1.bws?WSDL>. The 'Operations' tab lists the operation 'startProcess', which is selected. The 'Operation Detail' tab shows the operation name 'startProcess' and the attached service 'Start Process'.

So far, we have done nothing special relating to Job Scheduler. Everything described up to this point has been basic IBPM oriented activities but what we have achieved is to set the stage by

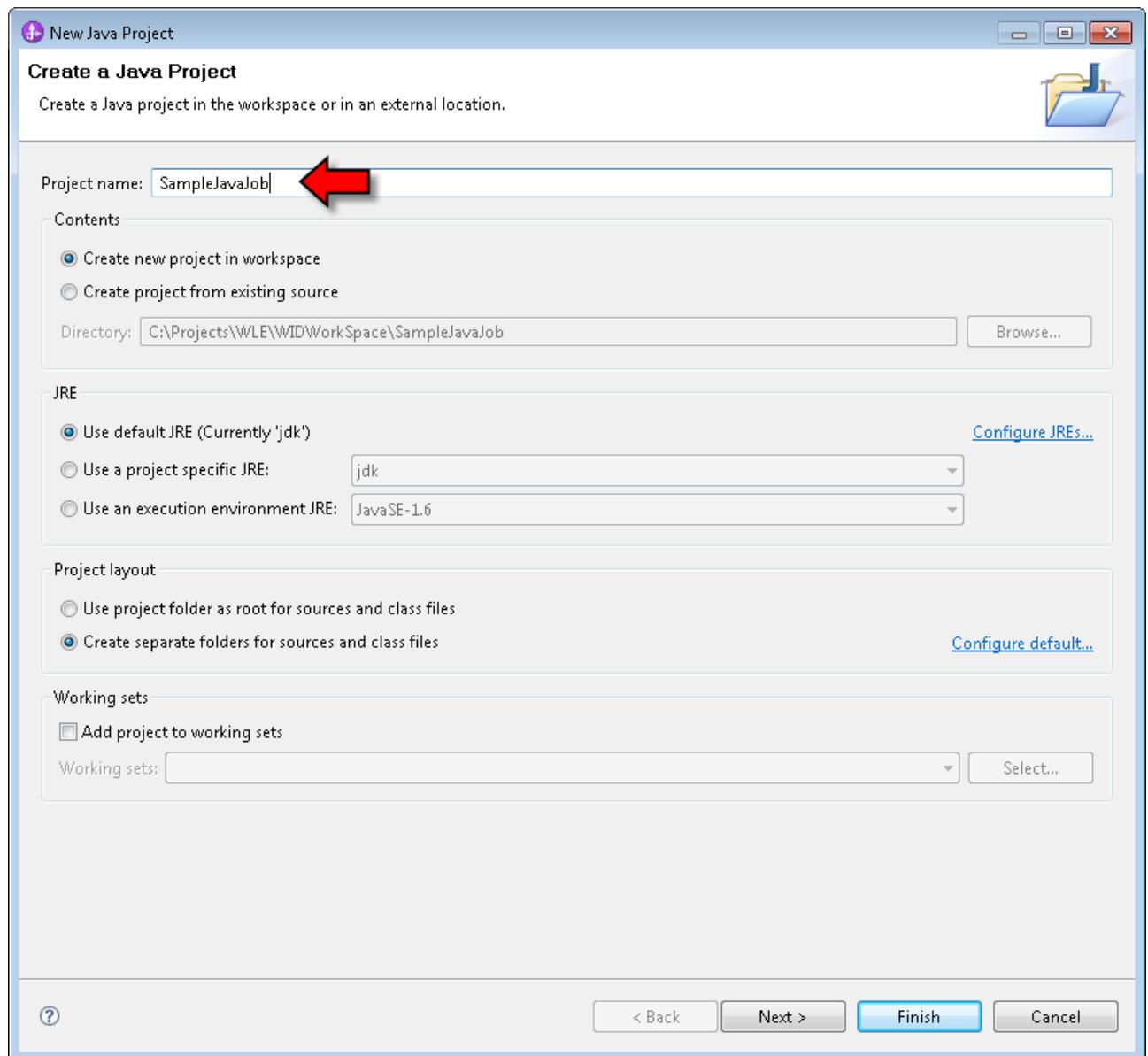
building and describing a simple test environment. Our goal now is to invoke the Generic Service called "Start Process" at scheduled intervals through the Job Scheduler environment.

The overall schematic is shown in the following image.



Job Scheduler will be configured with Job definitions that say what needs to happen and when. When a Job that is to be executed on IBPM is ready to be executed, custom Java code will be called in the context of Job Scheduler which will make a Web Services call to IBPM which will in turn execute the process when the request is received.

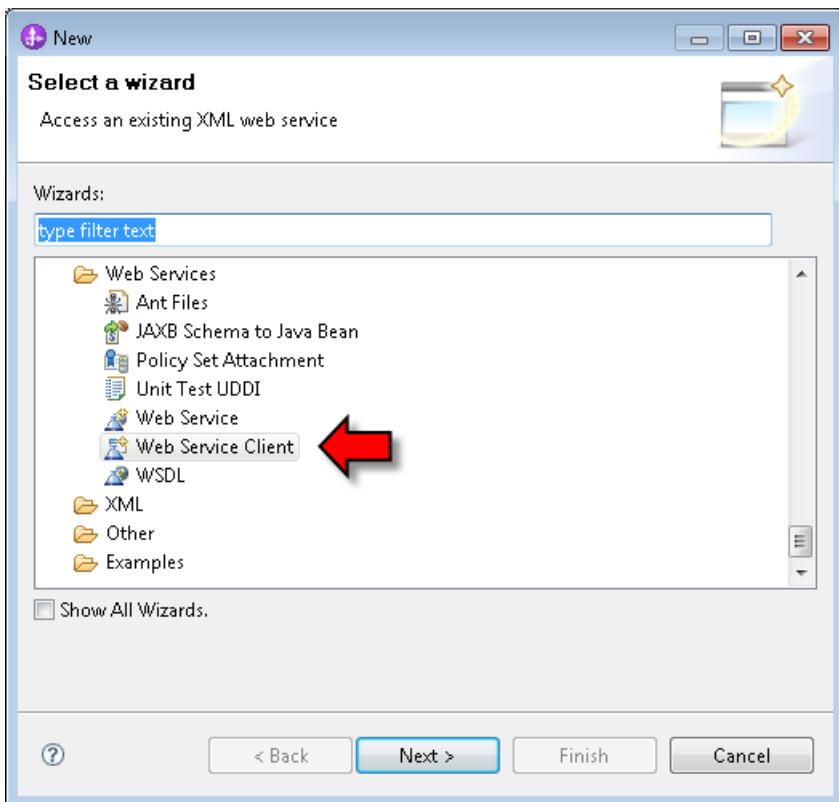
The next step in our solution is to build the Java Job code. We will start by building a simple Java Project in WID (we could also use RAD). We called our project SampleJavaJob.



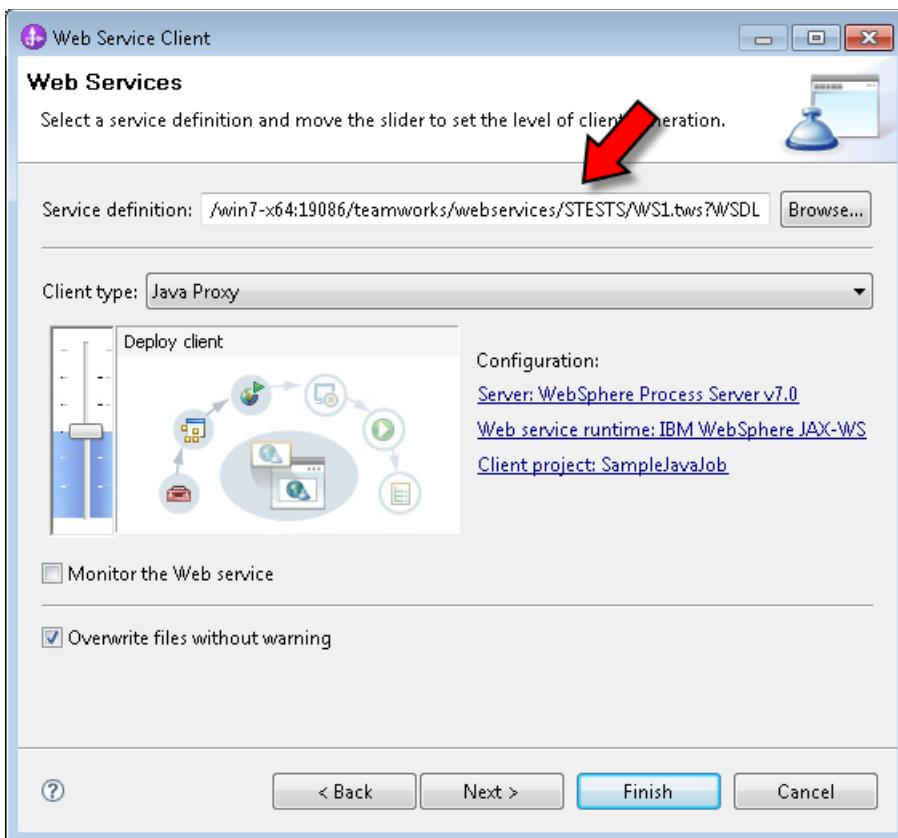
Since this code needs to make a Web Service call to IBPM, it is now time to build the Web Service client caller function. WID has the ability to take a WSDL file and generate all the Java code needed to invoke the service described by this WSDL. This makes calling a Web Service from Java a very easy proposition. The generated code will eventually be invoked by the Job Scheduler.

Right click the new project and select New > Other.

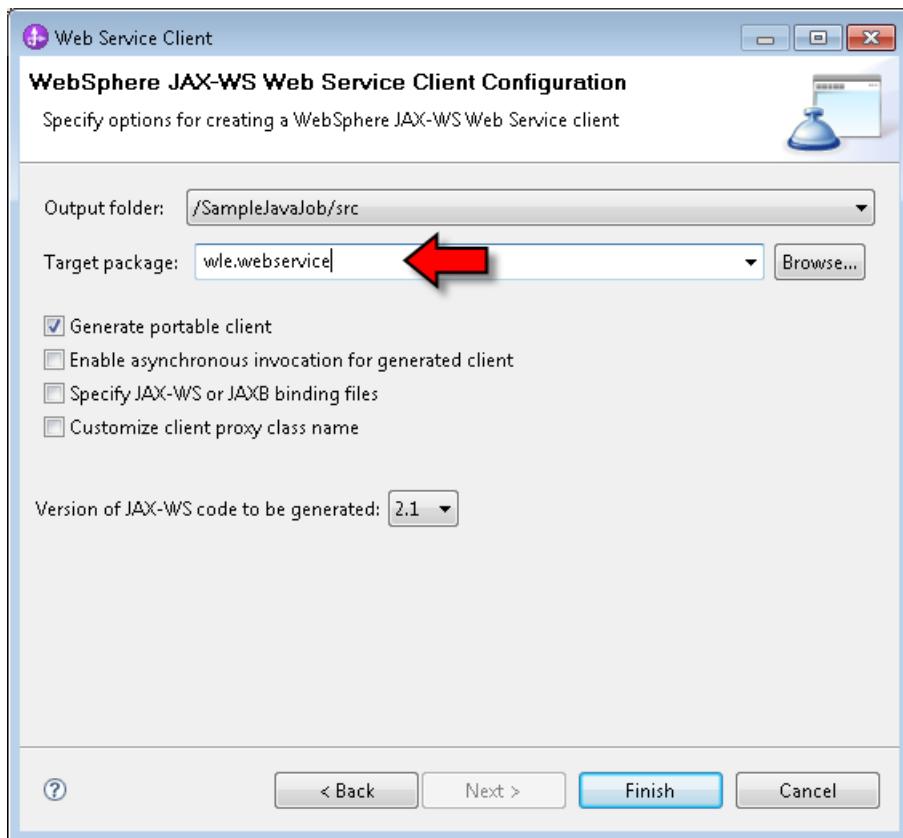
Within the Web Services folder, select Web Service Client:



In the next page of the Wizard, enter the URL show in the Web Services definition in IBPM PD.

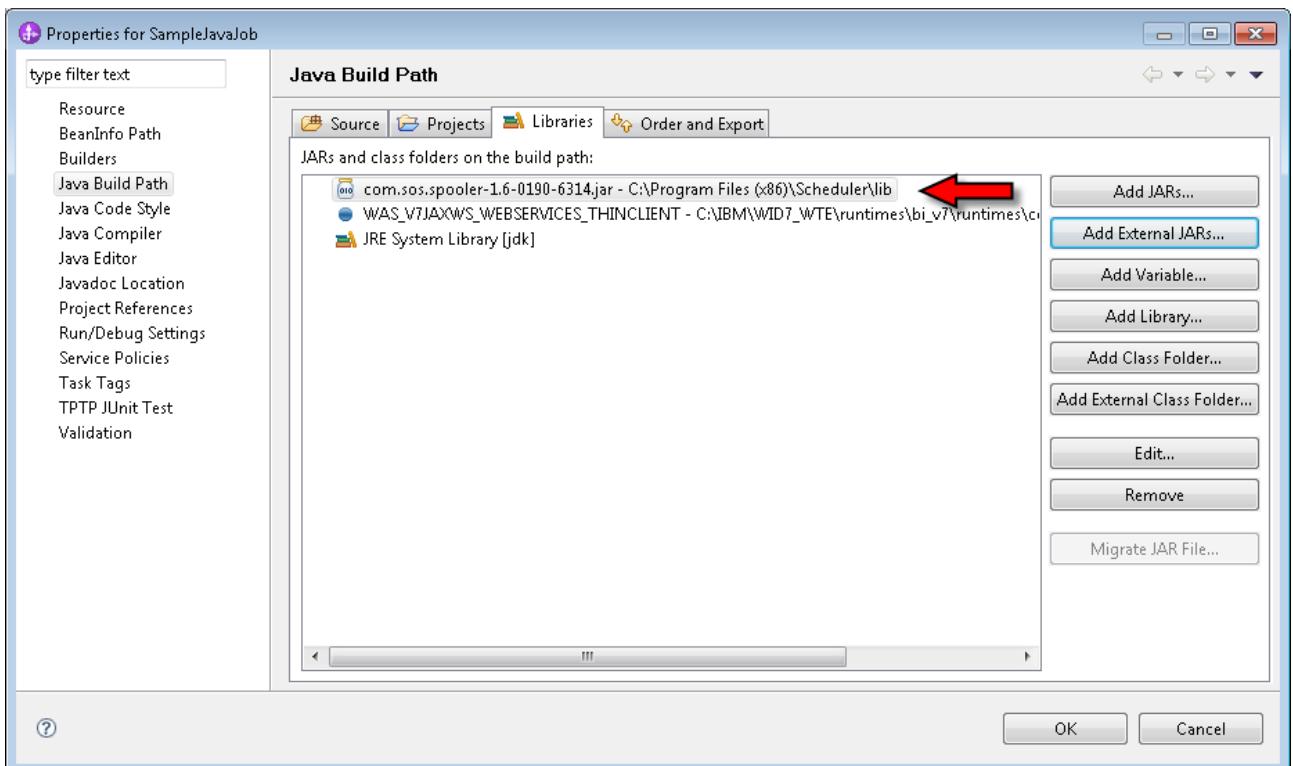


On the next page, set the Target package to be a meaningful name (for example, wle.webservice).

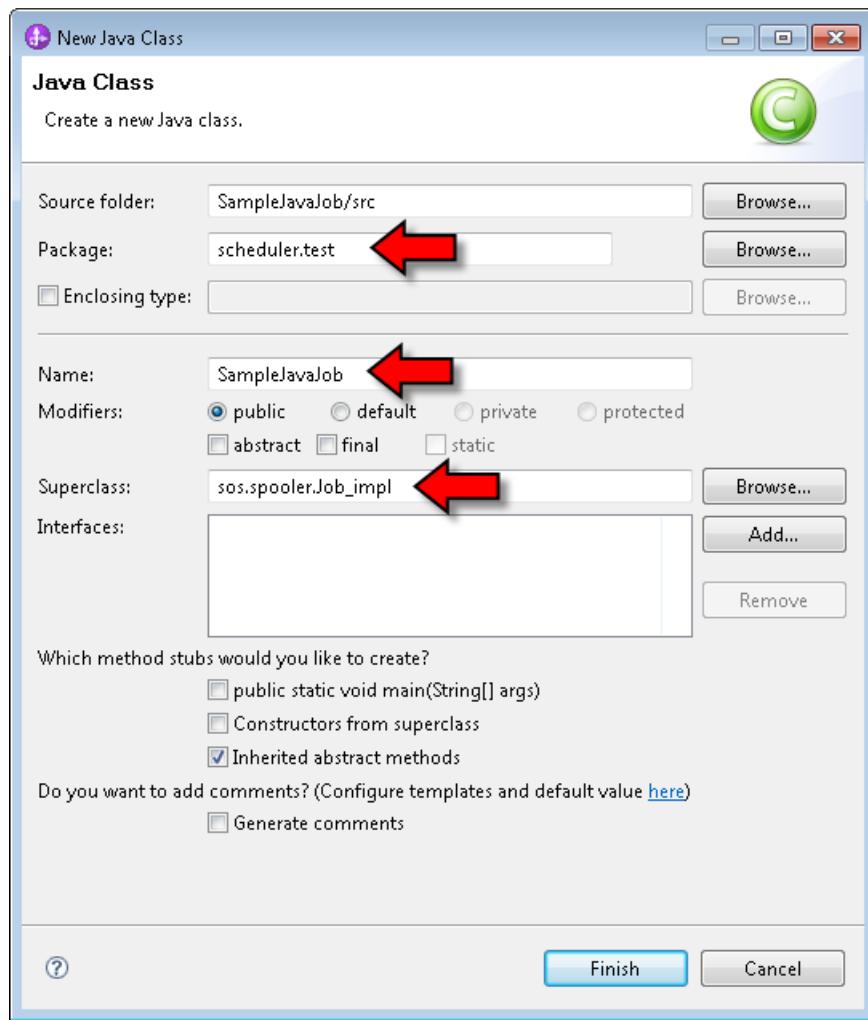


All the other settings can remain with their default settings. At the conclusion of this wizard activity, Java code will have been generated that will invoke the process as a Web Service. It is this code that we wish to invoke from within the Job Scheduler. Our next steps will be the creation of the Job Scheduler wrapper class.

Modify the build path of the Java profile within WID to include the `com.sos.spooler-<version>.jar` that is supplied with Job Scheduler and found in the `<Scheduler>/lib` folder. This Jar contains classes that we will need in our custom code. It provides the contract between Job Scheduler and the custom Job we are writing.



Create a new Java class that extends `sos.spooler.Job_impl`. We named this class `SampleJavaJob`. By extending the Job Scheduler supplied class, we inherit the necessary contracts and framework for Job Scheduler to invoke our own custom code.



The implementation of the class must implement/override a method called `spooler_process()`. This method is called to invoke an instance of the job. In our implementation we wish this to invoke a Web Service which in turn will invoke the BPD process. The following code shows our implementation.

```

package scheduler.test;
import sos.spooler.Job_impl;
import wle.webservice.WS1;
import wle.webservice.WS1PortType;

public class SampleJavaJob extends Job_impl {
    @Override
    public boolean spooler_process() throws Exception {
        spooler_log.info("Web Service Job Starting");

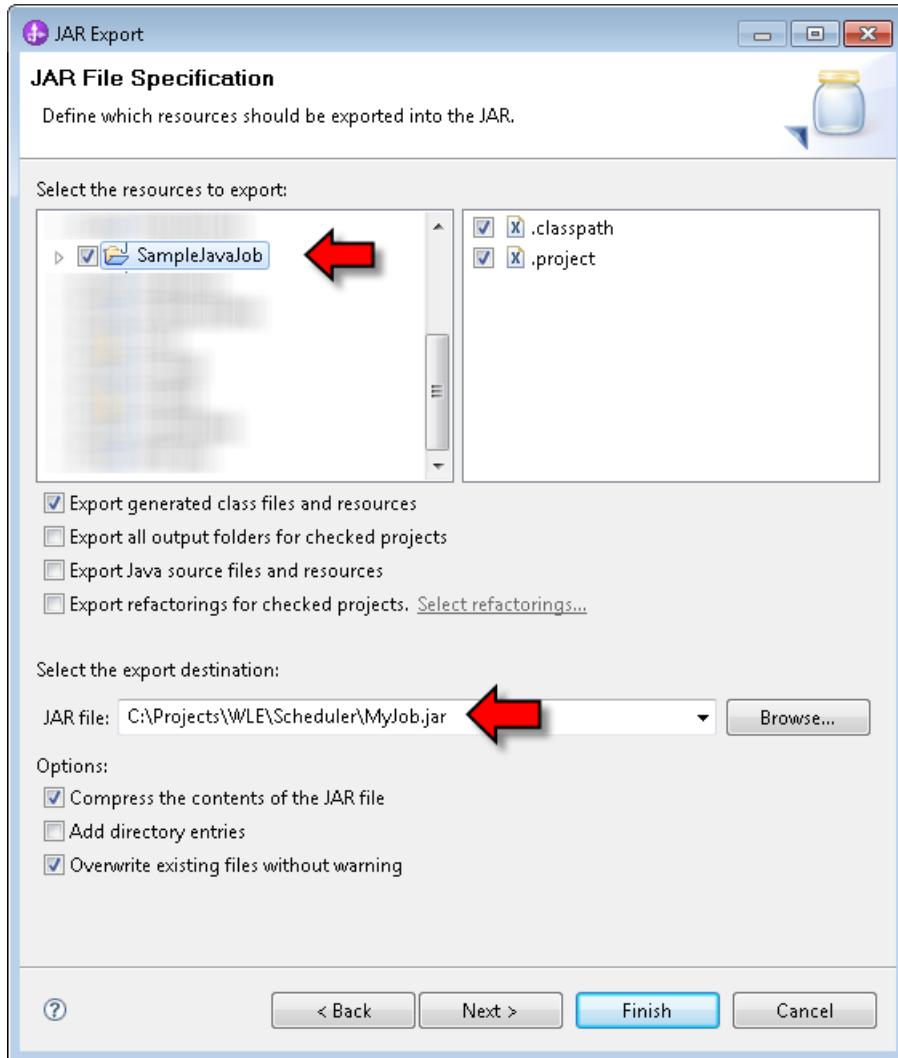
        // Called when the scheduler wishes this job to run
        WS1 ws1 = new WS1();
        WS1PortType ws1PT = ws1.getWS1Soap();
        ws1PT.startProcess("Hello World");

        spooler_log.info("Web Service Job Ending");
        return false;
    }
}

```

The code for the Java class is very simple. It creates an instance of the Service and then asks for the interface from that Service. Once in possession of the Interface, we are able to invoke the operation described within. We have hard-coded the parameter for the interface but this could also be dynamically supplied if required.

With the coding complete for the Job, what remains is to make Job Scheduler aware of the new class so that we can define a scheduled Job to invoke it. First we export the SampleJavaJob project as a JAR file. This will include the code we just created.



Job Scheduler has a configuration file called `factory.ini` that is located in `<scheduler>/config` directory. This file can be edited with notepad or a similar text editor. Within the file is a section called `[java]`. This contains an entry called `class_path`. We must now modify this entry.

First we must add the JAR file we just exported.

Next we must add a second jar file which is the IBM supplied set of libraries for making Web Services calls for JAX-WS. This jar is called `com.ibm.jaxws.thinclient_7.0.0.jar` and is provided with a WAS 6.1 server. Within WID, this can be found in the `<WLE>/AppServer/runtimes` directory.

An example entry for `factory.ini` may look like:

```
[java]
class_path =
C:\IBM\Lombardi7\AppServer\runtimes\com.ibm.jaxws.thinclient_7.0.0.jar;C:\Projects\WLE\Scheduler\MyJob.jar;C:\Program Files\scheduler\lib\*.jar
```

With these changes made, we are now ready to start Job Scheduler. If Job Scheduler is already started, shut it down first

From a DOS window and located in the <scheduler>/bin directory, we run the command:

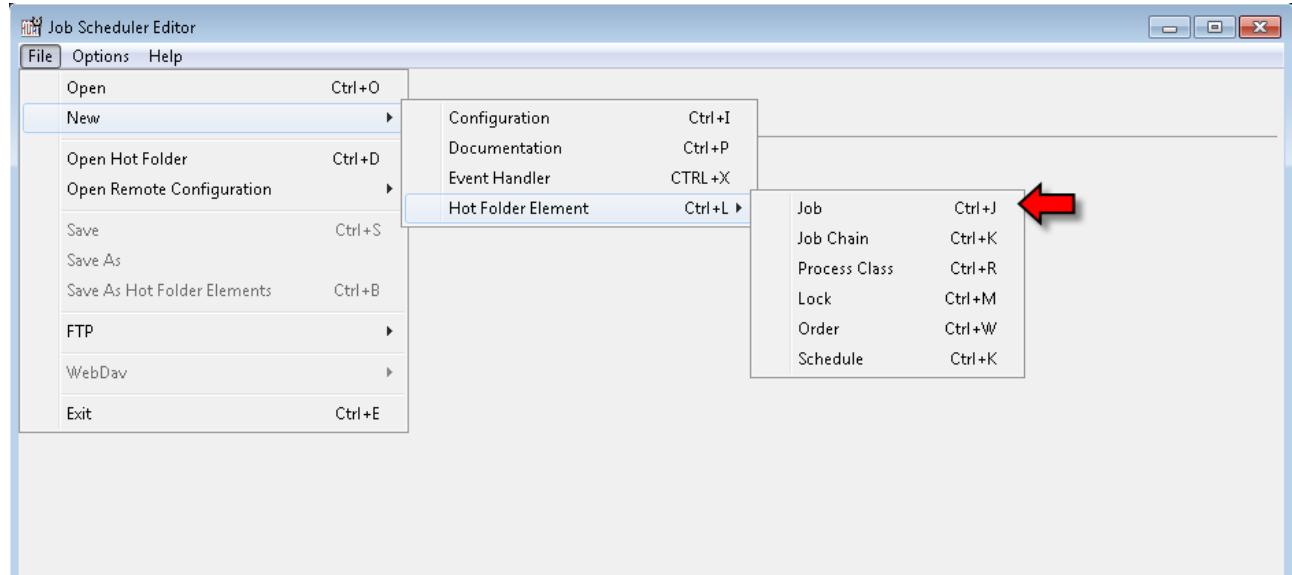
```
jobscheduler.cmd start
```

The program runs and holds the DOS Window open. In production, it would be started from the Windows Services.

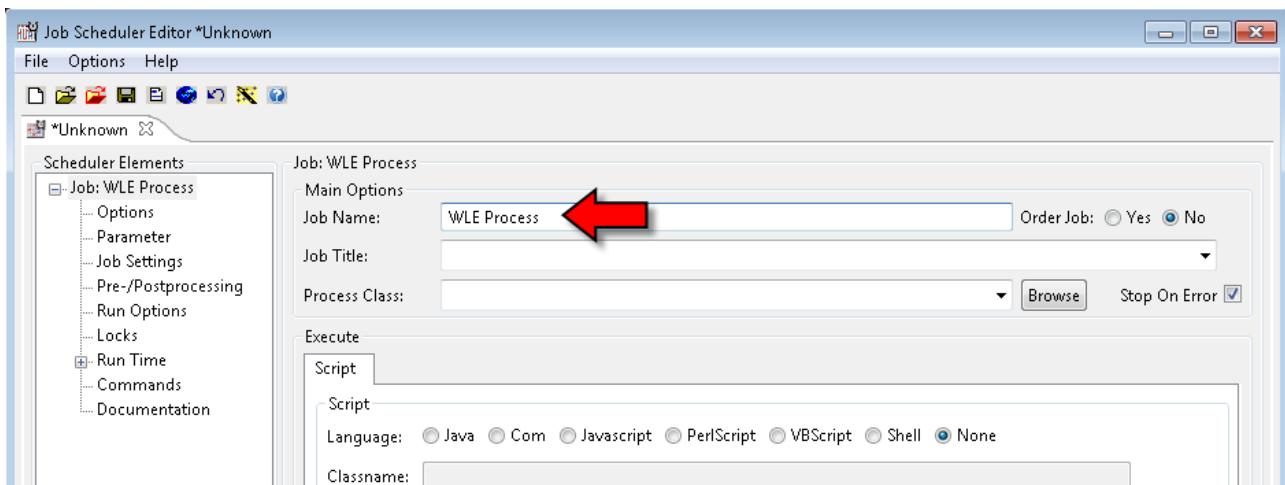
We are now ready to define the Job to the scheduler. Launch the Job Scheduler editor by running the command jobeditor.cmd.

Create a new job from the menu

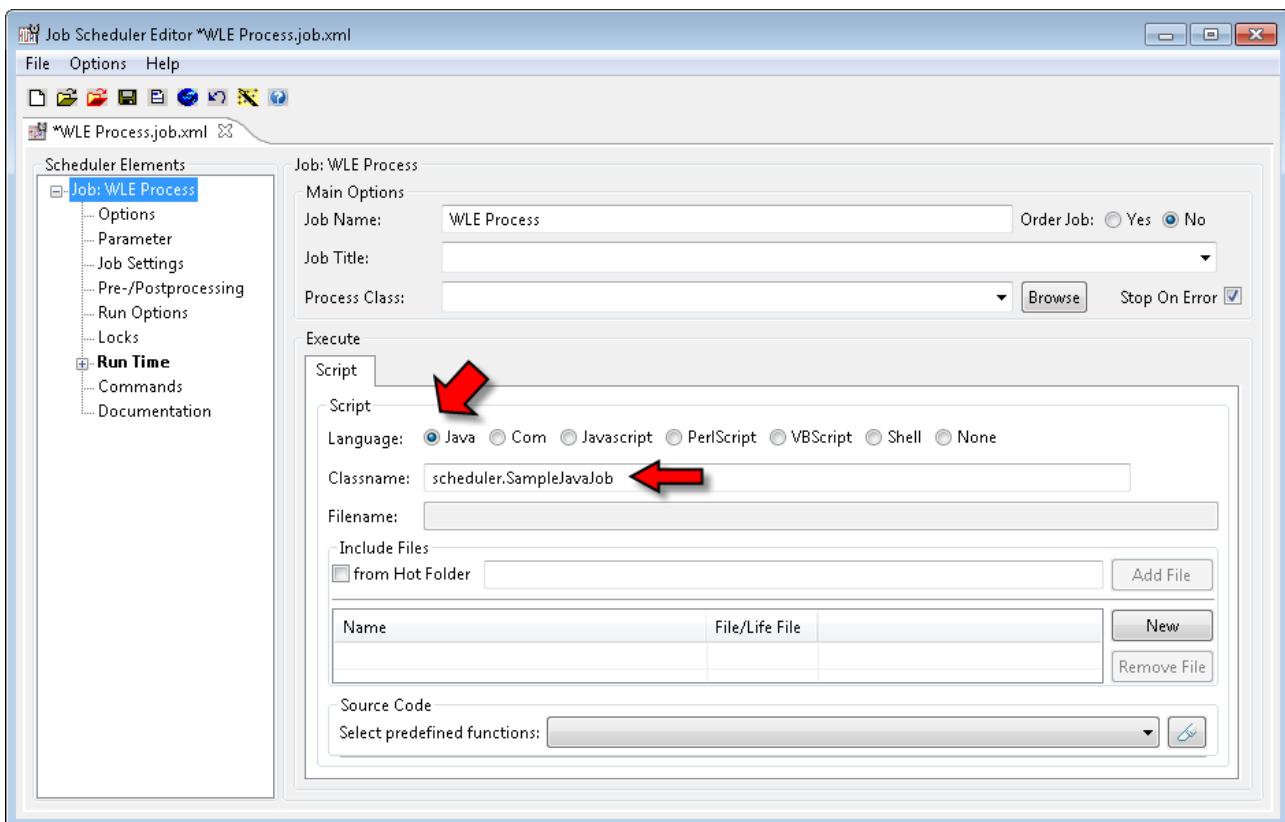
File ► New ► Hot Folder Element ► Job



Give the new Job a meaningful name, for example “IBPM Process”.



Next within the Execute section define how the Job will run. Define the job to be a script and written in Java. In the Classname field, enter the full name of the Java Job class including its package name. Remember that in a previous set of steps we exported the implementation of this class as a Jar file and then added the Jar into the class path of Job Scheduler.



Save the result in the <scheduler>/config/live folder.

Launch the Web based dashboard for Job Scheduler. This can be achieved by opening a browser to <http://localhost:4444/>.

In the Jobs page, we can see the Job we have just defined.

Documentation  

Update (every 5s) incl. Hot Folders Extras

Menu  **JOB SCHEDULER**
win7-x64:4444

ID: scheduler State: running Pid: 5284 0 need process 0 tasks 0 orders Time: 2010-10-05 13:52:05
0 jobs running 0 stopped Scheduler Start Time: 2010-10-05 13:51:10

Jobs Job Chains Orders Schedules Process Classes Last Activities

All jobs with state (all) and with process class (all) Show tasks

Sort jobs by name in ascending order

Job	Time	Steps	Next start / Orders	Job menu
scheduler_check_sanity		Check Sanity		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_check_updates		check availability of Job Scheduler updates		<input type="button" value="Job menu"/>
pending		0 2010-10-11 20:00:00		
scheduler_cleanup_files		Remove temporary files		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_cleanup_history		Compress log entries in Job Scheduler history		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_dequeue_events		Send buffered events to the supervisor		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_dequeue_mail		Send buffered mails from mail queue		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_restart		Restart Job Scheduler		<input type="button" value="Job menu"/>
pending		0 Without start time		
scheduler_rotate_log		Rotate and compress logfiles		<input type="button" value="Job menu"/>
pending		0 Without start time		
WLE Process		0 Without start time		<input type="button" value="Job menu"/>
pending				

Make sure that the IBPM Process Application we are about to call is installed and running in the IBPM server. In the next step we are about to execute it. Clicking on the Job definition shows us details of that Job. In the Job menu, we find and select an entry called “Start task immediately”. This will cause Job Scheduler to run an instance of the job immediately. This will prove to us that Job Scheduler is able to invoke the process.

Documentation  

Update (every 5s) incl. Hot Folders Update Extras

Menu **JOB SCHEDULER** win7-x64:4444 Time: 2010-10-05 13:55:23
ID: scheduler State: running Pid: 5284 0 need process 0 tasks 0 orders Scheduler Start Time: 2010-10-05 13:51:10

Jobs Job Chains Orders Schedules Process Classes Last Activities

All jobs with state (all) and with process class (all) Show tasks

Sort jobs by name in ascending order

Job	Time	Steps	Next start / Orders	Action
scheduler_check_sanity		Check Sanity		Job menu
pending		0 Without start time		
scheduler_check_updates		check availability of Job Scheduler updates		Job menu
pending		0 2010-10-11 20:00:00		
scheduler_cleanup_files		Remove temporary files		Job menu
pending		0 Without start time		
scheduler_cleanup_history		Compress log entries in Job Scheduler history		Job menu
pending		0 Without start time		
scheduler_dequeue_events		Send buffered events to the supervisor		Job menu
pending		0 Without start time		
scheduler_dequeue_mail		Send buffered mails from mail queue		Job menu
pending		0 Without start time		
scheduler_restart		Restart Job Scheduler		Job menu
pending		0 Without start time		
scheduler_rotate_log		Rotate and compress logfiles		Job menu
pending		0 Without start time		
WLE Process		1 Without start time		Job menu
pending				<ul style="list-style-type: none"> <input type="checkbox"/> Show log <input type="checkbox"/> Show description <input type="checkbox"/> Show dependency <input type="checkbox"/> Show start times <input type="checkbox"/> Start task immediately <input type="checkbox"/> Start task at <input type="checkbox"/> Start task parametrized <input type="checkbox"/> Set run time <input type="checkbox"/> Stop <input type="checkbox"/> Unstop <input type="checkbox"/> Reread <input type="checkbox"/> End tasks <input type="checkbox"/> Suspend tasks <input type="checkbox"/> Continue tasks <input type="checkbox"/> Delete job



After completion and clicking on the Last Activities button, the result will be:

Documentation Update (every 5s) incl. Hot Folders Update Extras

JOB SCHEDULER
win7-x64:4444

ID: scheduler State: running Pid: 5284 Time: 2010-10-05 14:00:15
0 jobs running 0 stopped 0 need process 0 tasks 0 orders Scheduler Start Time: 2010-10-05 13:51:10

Jobs Job Chains Orders Schedules Process Classes Last Activities

Show all Show only orders Show only tasks Show only faultily tasks and orders Show last tasks error

Order ID/ Job name	Started	Ended	Job chain/ Cause	Order state/ Exitcode
WLE 2010-10-05	2010-10-05	queue_at		Show task log
Process 13:54:37	13:54:41			

Job menu **JOB** Job menu

WLE Process

File timestamp: 2010-10-05 13:50:43
State: pending
State text:
Error:
Next start:
Steps: 1 Tasks: 1
Log level: info mail on error mail on warning From: scheduler@localhost
SMTP:localhost

Task Queue		Task History	
Id	Cause	Started	Steps
1	queue_at	2010-10-05 13:54:37	1 2010-10-05 13:54:41

If we examine the history in the console log of IBPM, we will have found that the process will have executed. What remains is to schedule the work to be executed at configured times in the future, for our testing purposes; we will have the process run every 15 seconds.

From the Job menu, select “Set run time”.

Documentation Update (every 5s) incl. Hot Folders Update Extras

JOB SCHEDULER
win7-x64:4444

ID: scheduler State: running Pid: 5284 Time: 2010-10-05 14:01:40
0 jobs running 0 stopped 0 need process 0 tasks 0 orders Scheduler Start Time: 2010-10-05 13:51:10

Jobs Job Chains Orders Schedules Process Classes Last Activities Hide

All jobs with state (all) and with process class (all) Show tasks

Sort jobs by name in ascending order

Job	Time	Steps	Next start / Orders
scheduler_check_sanity	Check Sanity	0 Without start time	Job menu
scheduler_check_updates	check availability of Job	0 2010-10-11 20:00:00	Job menu
scheduler_cleanup_files	Remove temporary files	0 Without start time	Job menu
scheduler_cleanup_history	Compress log entries in Job	0 Without start time	Job menu
scheduler_dequeue_events	Scheduler history	0 Without start time	Job menu
scheduler_dequeue_mail	Send buffered events to the supervisor	0 Without start time	Job menu
scheduler_restart	Send buffered mails from mail queue	0 Without start time	Job menu
scheduler_rotate_log	Restart Job Scheduler	0 Without start time	Job menu
WLE Process	Rotate and compress logfiles	0 Without start time	Job menu
pending			

Job menu **B** Job menu

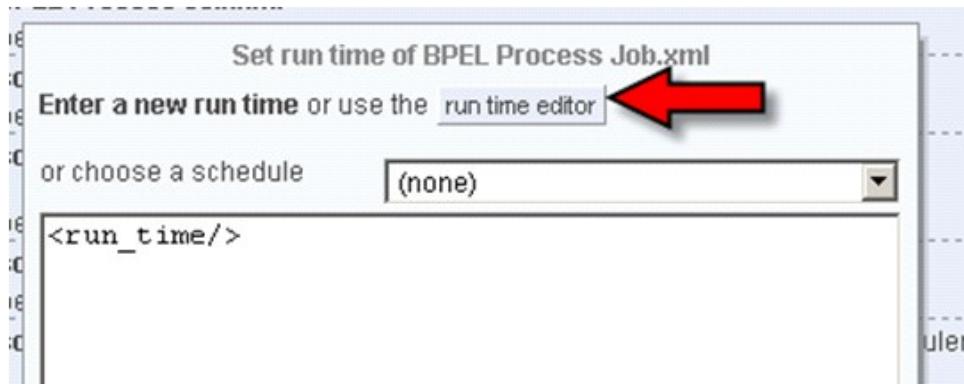
WLE Process

File timestamp: 2010-10-05 13:50:43
State: pending
State text:
Error:
Next start:
Steps: 1 Tasks: 1
Log level: info mail on error mail on warning From: scheduler@localhost
SMTP:localhost

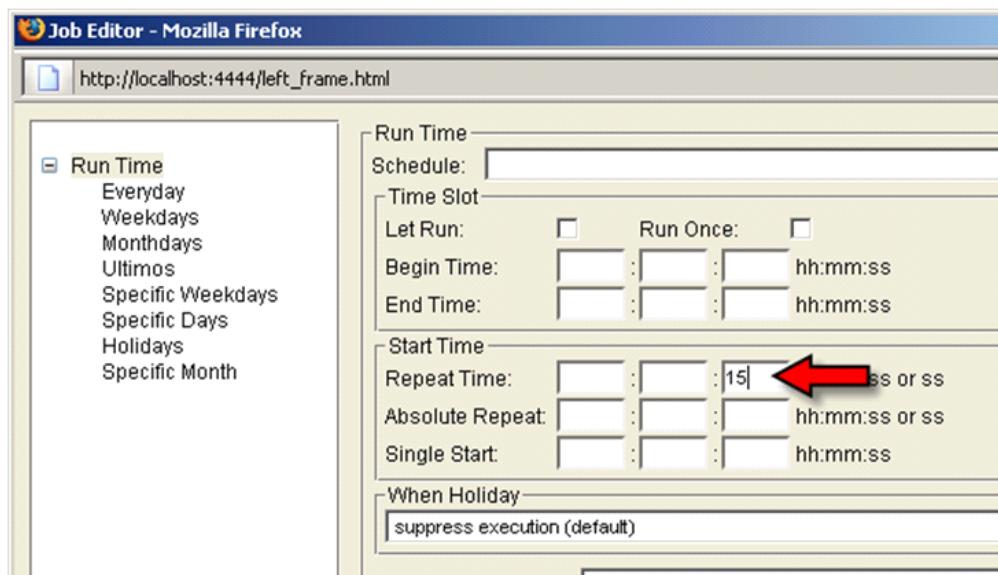
Start task immediately
Start task at
Start task parametrized
Set run time
Stop
Unstop
Reread
End tasks
Suspend tasks
Continue tasks
Delete job

Id	Cause	Started	Steps
1	queue_at	2010-10-05 13:54:37	1 2010-10-05 13:54:41

Click on the “run time editor” button to launch the assistance window for creating run time specifications.



In the assistance window, enter a value of 15 seconds in the “Repeat Time” field.



Complete the changes and then watch the results. After a short period of time we will find that the General Service has executed every 15 seconds.

Documentation Update (every 5s) incl. Hot Folders Update Extras

JOB SCHEDULER
win7-x64:4444

ID: scheduler State: running Pid: 5284 0 need process 0 tasks 0 orders Time: 2010-10-05 14:04:01 Scheduler Start Time: 2010-10-05 13:51:10

Jobs Job Chains Orders Schedules Process Classes Last Activities Hide

All jobs with state (all) and with process class (all) Show tasks

Sort jobs by name in ascending order

Job	Time	Steps	Next start / Orders	Job menu
scheduler_check_sanity		Check Sanity		Job menu
pending		0 Without start time		
scheduler_check_updates		check availability of Job		Job menu
pending	2010-10-11 20:00:00	Scheduler updates		
scheduler_cleanup_files		Remove temporary files		Job menu
pending		0 Without start time		
scheduler_cleanup_history		Compress log entries in Job		Job menu
pending		Scheduler history		
scheduler_dequeue_events		0 Without start time		Job menu
pending		Send buffered events to the supervisor		
scheduler_dequeue_mail		Send buffered mails from mail queue		Job menu
pending		0 Without start time		
scheduler_restart		Restart Job Scheduler		Job menu
pending		0 Without start time		
scheduler_rotate_log		Rotate and compress logfiles		Job menu
pending		0 Without start time		
WLE Process				Job menu
pending	2010-10-05 14:04:15	3		

WLE Process

File timestamp: 2010-10-05 14:03:24
 State: pending
 State text:
 Error:
 Next start: 2010-10-05 14:04:15 (-13s)
 Steps: 3 Tasks: 3
 Log level: info mail on error mail on warning From: scheduler@localhost
 SMTP: localhost

Task Queue **Task History**

Id	Cause	Started	Steps	Ended	Show log
4	period_repeat	2010-10-05 14:03:59	1	2010-10-05 14:04:00	Show log
3	period_repeat	2010-10-05 14:03:43	1	2010-10-05 14:03:44	Show log
2	period_repeat	2010-10-05 14:03:27	1	2010-10-05 14:03:28	Show log

Obviously Job Scheduler has a lot of features and we have barely scratched the surface on how to use it but what this section has shown is how to expose a Business Process as a Web Service, how to write a Java Job wrapper to call that Web Service, how to define the Java Job to Job Scheduler and finally how to run a simple test to ensure that Job Scheduler can correctly call the process.

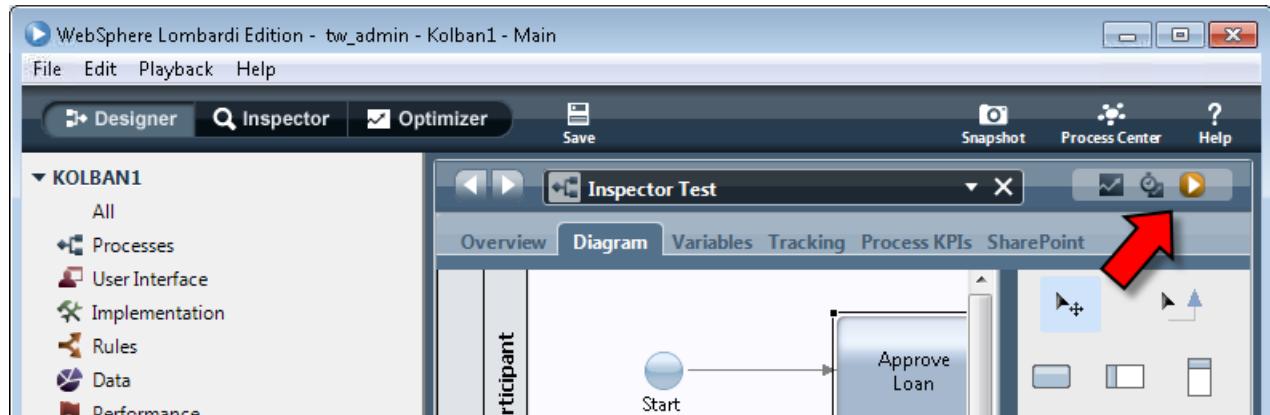
Debugging

When building solutions, unfortunately they don't always work first time.

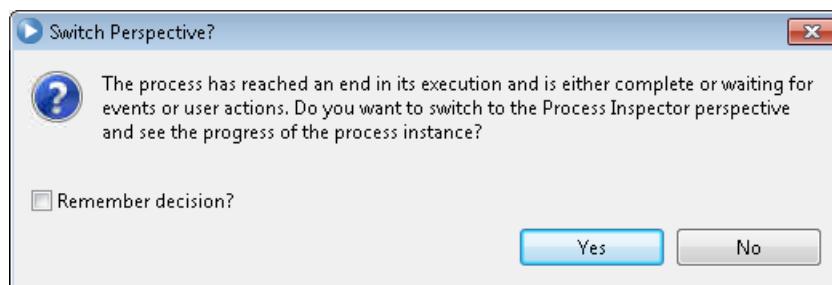
Debugging with Inspector

One of the core components of IBPM PD is the Inspector. By using the Inspector you can demonstrate playbacks of the solution in real-time. It is Inspector that provides a lot of the iterative nature of IBPM value. Inspector allows the process developer to run instance of the process on the Process Center Server.

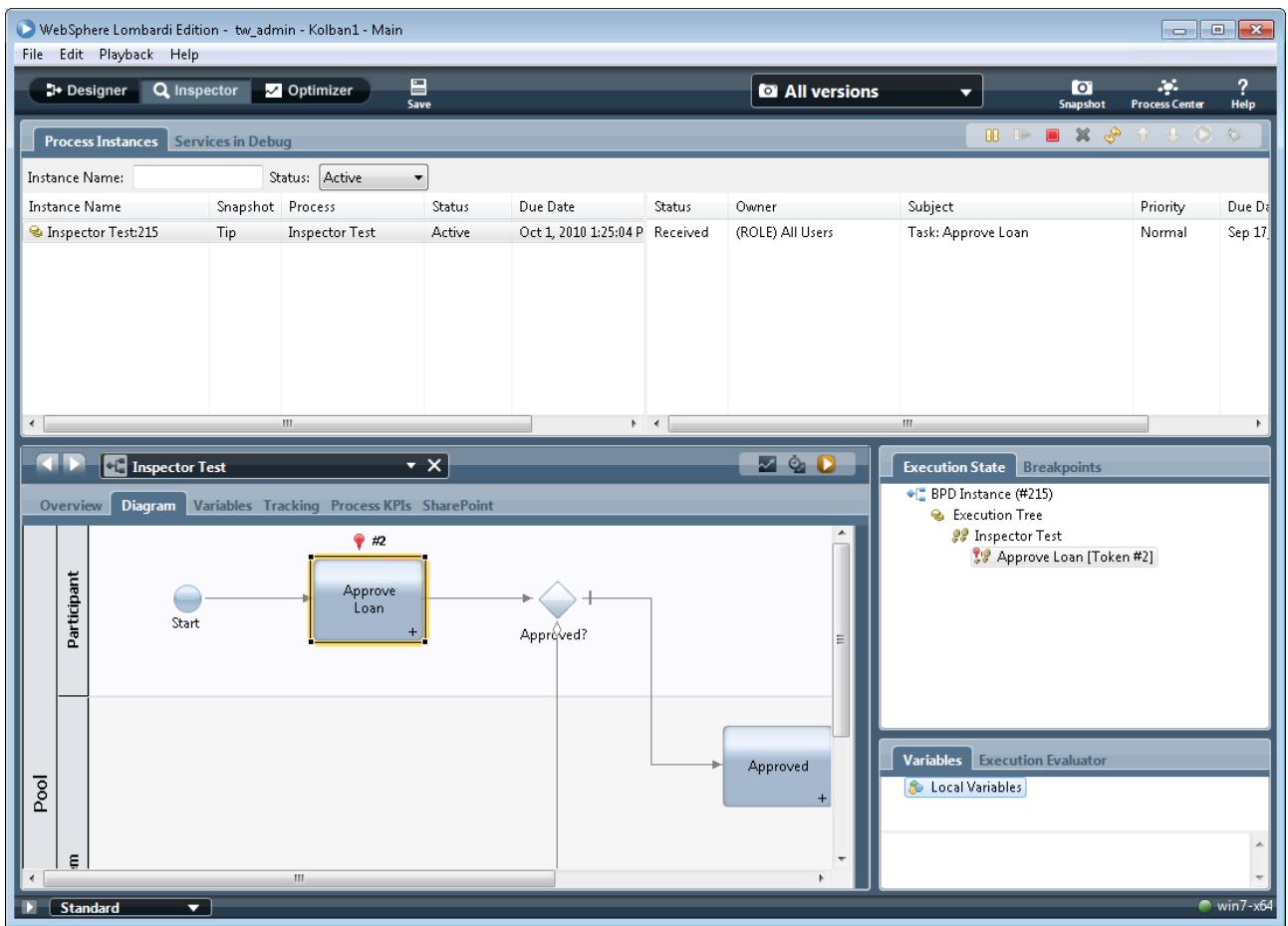
There are a number of ways to start a process for testing. On the BPD diagram, there is a run button at the top right:



This will start a run of the process and prompt you to switch perspectives to that of the Inspector tool:



Within the Inspector perspective of IBPM PD, there are many parts.



In the top left we have the list of process instances. These can be filtered by instance name or by process status. The instance name is a dynamic filter and can be extremely useful if there are many entries in the list. Entering a prefix or part of the name can dramatically reduce the number of items to be seen.

Instance Name:	Status:	Active			
Instance Name	Snapshot	Process	Status	Due Date	Instance Id
Inspector Test:215	Tip	Inspector Test	Active	Oct 1, 2010 1:25:04 PM	215

The columns in this area are:

Column Name	Description
Instance Name	The human readable name of the instance of the process
Snapshot	The snap shot associated with the process
Process	The name of the BPD
Status	The status of the process. Options are: <ul style="list-style-type: none"> Active Completed Failed Terminated Did not start Suspended
Due Date	The date/time at which the process is due
Instance Id	The IBPM unique instance ID for this instance of the process

The area to the right of the process instances changes depending on which process instance is selected.

Status	Owner	Subject	Priority	Due Date	Task Id
Received	(ROLE) All Users	Task: Approve Loan	Normal	Sep 17, 2010 2:...	211

It describes the tasks/steps within the process. The columns in this area are:

Column Name	Description
Status	The current status of the task. States are: <ul style="list-style-type: none">Received
Owner	The owner of the task.
Subject	The name of the task within the process.
Priority	The priority of the task. Choices are: <ul style="list-style-type: none">Normal
Due Date	When this task is due.
Task Id	The unique ID of this task.

Not all BPD activities list/show up in this area. The ones that don't include:

- Activities implemented as inline JavaScript

At the top right of the Inspector are a series of control buttons:

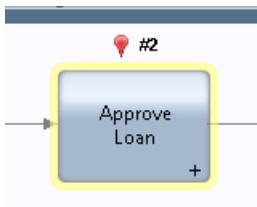


The meaning of the buttons change depending on what is selected.

- Suspends the process. The status of the process changes to suspended.
- Resumes the process. If the process has previously been suspended, this button resumes the process.
- Terminates the process. The process is terminated. The status of the processes changes to Terminated.
- Deletes the data associated with the process.
- Refreshes inspector with the updated knowledge of the environment from the runtime.
- Runs the selected task.
- Debugs the selected task.

If the Task includes a Coach, a browser window will appear in which the details of the coach will be shown and the user will be allowed to interact with the process. When the coach is completed, the process will continue to the next task.

If there is a task waiting for attention, it will show with a yellowed background:

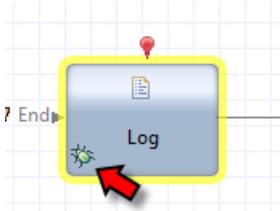


We can set breakpoints in activities in the Inspector view. By right-clicking on an Activity, we bring up the context menu and in that menu, there is a Breakpoint entry. Switching on the Breakpoint results in a breakpoint icon (a bug icon) appearing in the lower left of the activity.

Here we see an inspector view and a breakpoint about to be added.

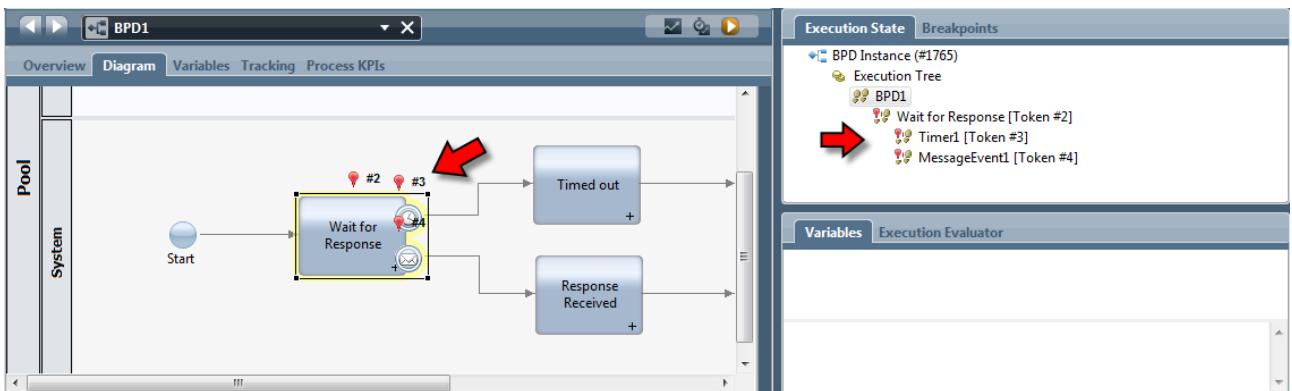
Service	Owner	BPD Instance ...	BPD I...
My Data Source 3	tw_admin		null

After having added the Breakpoint, the Activity shows the debug icon:



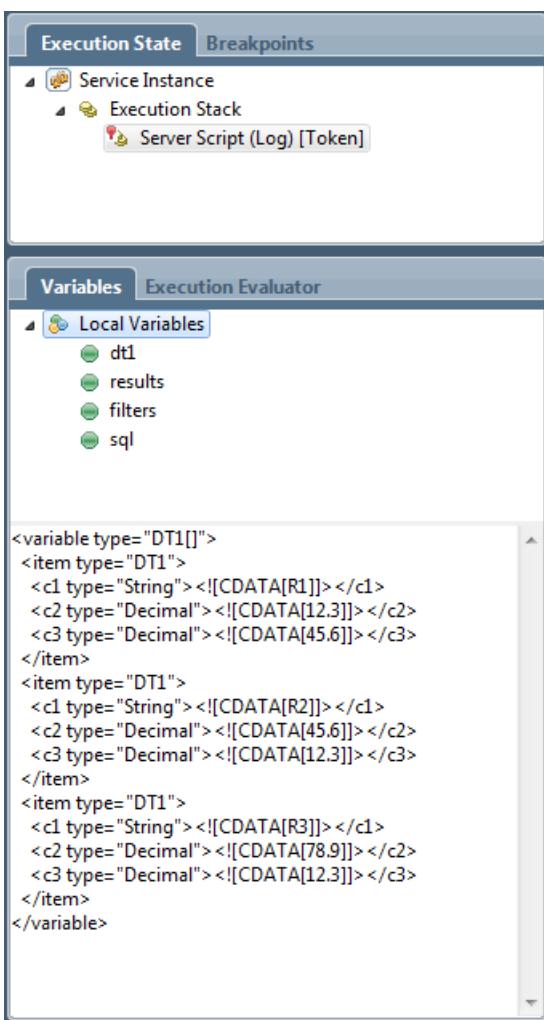
When a process is executed under the inspector and allowed to run freely, execution will stop at the next breakpoint encountered. We also have the ability to dynamically control whether or not a breakpoint is honored (causes the interruption of the process). Again from the context menu, we can select Breakpoint Condition after a Breakpoint has been enabled. Selecting this entry produces a dialog window into which a JavaScript expression can be entered. The expression must result in a true or false outcome. When the breakpoint is encountered during execution, the expression is evaluated. The breakpoint is ignored if the expression evaluates to false.

The Inspector can also show us where the tokens are within our process.



In the diagram view, the tokens are shown as red colored markers with a number beside them. In the Execution State area, we again see these markers along with the name of the step/activity against which the token is blocked or waiting.

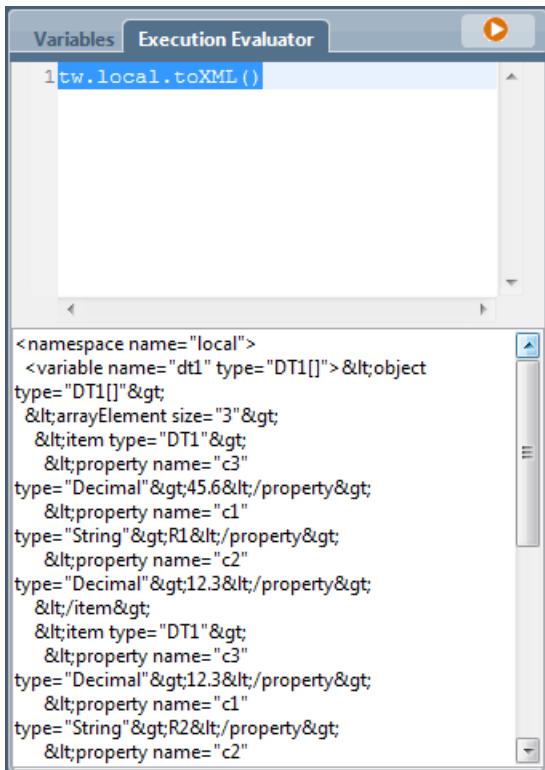
In addition, we can see the values of any currently set variables.



By clicking on a variable name, its current string representation/value is shown in the lower portion of the variables window.

A JavaScript execution evaluator is also provided that allows the user to enter a JavaScript

expression that is executed within the Process Server runtime. This can be used to print values of variables or execute small snippets of JavaScript during testing. After entering (or copy/pasting) the JavaScript to execute, the run button can be pressed and the JavaScript will execute.

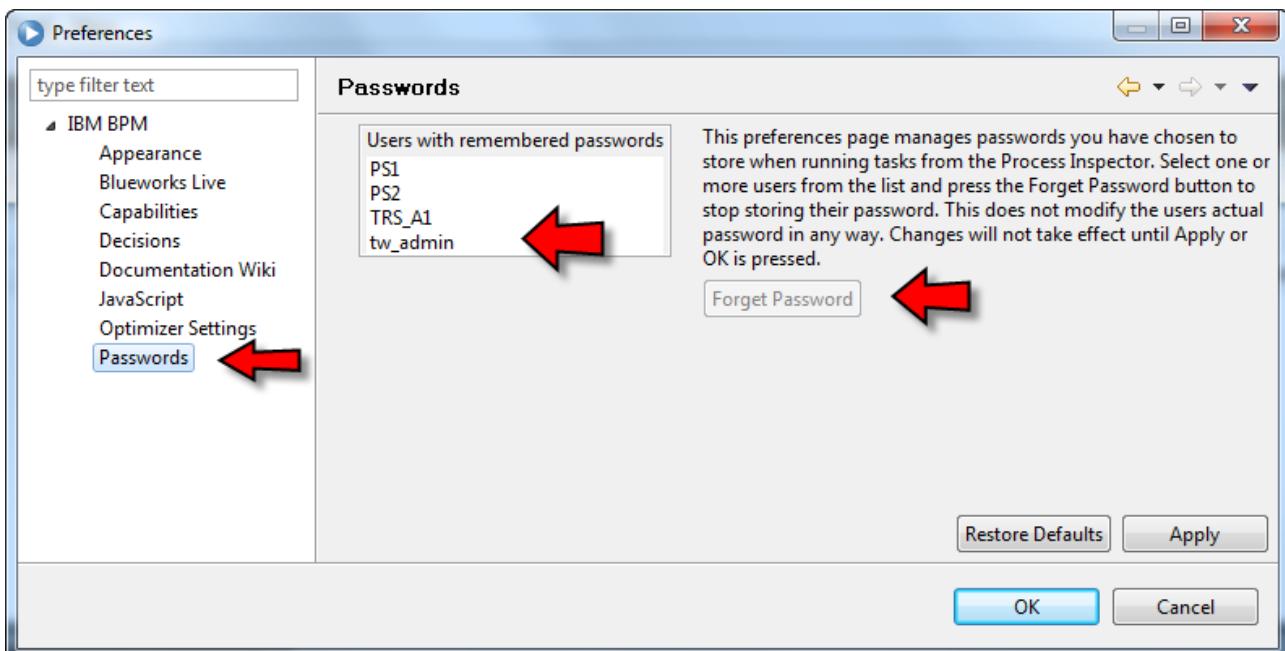


The screenshot shows the 'Execution Evaluator' tab of a software interface. The input field contains the JavaScript code `tw.local.toXML()`. The output pane displays the resulting XML code:

```
<namespace name="local">
<variable name="dt1" type="DT1[]">&lt;object
type="DT1[]">&gt;
&lt;arrayElement size="3"&gt;
&lt;item type="DT1">&gt;
&lt;property name="c3"
type="Decimal">&gt;45.6&lt;/property&gt;
&lt;property name="c1"
type="String">&gt;R1&lt;/property&gt;
&lt;property name="c2"
type="Decimal">&gt;12.3&lt;/property&gt;
&lt;/item&gt;
&lt;item type="DT1">&gt;
&lt;property name="c3"
type="Decimal">&gt;12.3&lt;/property&gt;
&lt;property name="c1"
type="String">&gt;R2&lt;/property&gt;
&lt;property name="c2"
type="Decimal">&gt;12.3&lt;/property&gt;

```

When launching a task from the lists of tasks, we are prompted with a selection of users eligible to work upon that task. PD has the ability to remember the password for users to save us from having to enter it. If we miss-type the password or the password changes, we must tell PD to forget about its remembrance of the password. This can be achieved through the Preferences window:



Debugging the environment

The environment in which IBPM executes causes logs to be written. Knowing where these logs exist and what their contents are is vital if lower level debugging is required. The logs for the server are written to the standard WAS consoles. These will exist in <profile>/logs. Specifically, the following relative files are useful:

- server1/SystemOut
- server1/SystemErr
- ffdc/*

Browser tabs and Process Inspector

When using Process Inspector to examine coaches, it has been found that if the browser is FireFox, then a new tab is opened for each coach displayed. This is not necessarily a good thing as the result is a set of old browser tabs that simply remain around after use that add nothing but clutter to the overall view of the browser.

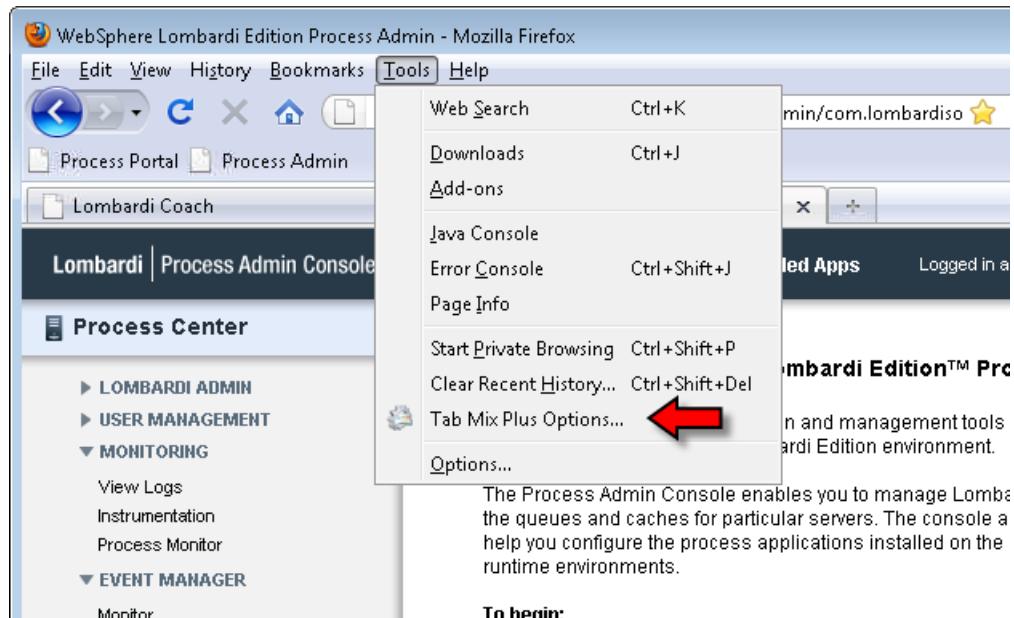
Fortunately, there is an elegant solution. The FireFox browser supports "plugins" and one such plugin is called "Tab Mix Plus". The web site for this plugin is:

<http://tmp.garyr.net/help/>

while the FireFox plugin itself can be found at:

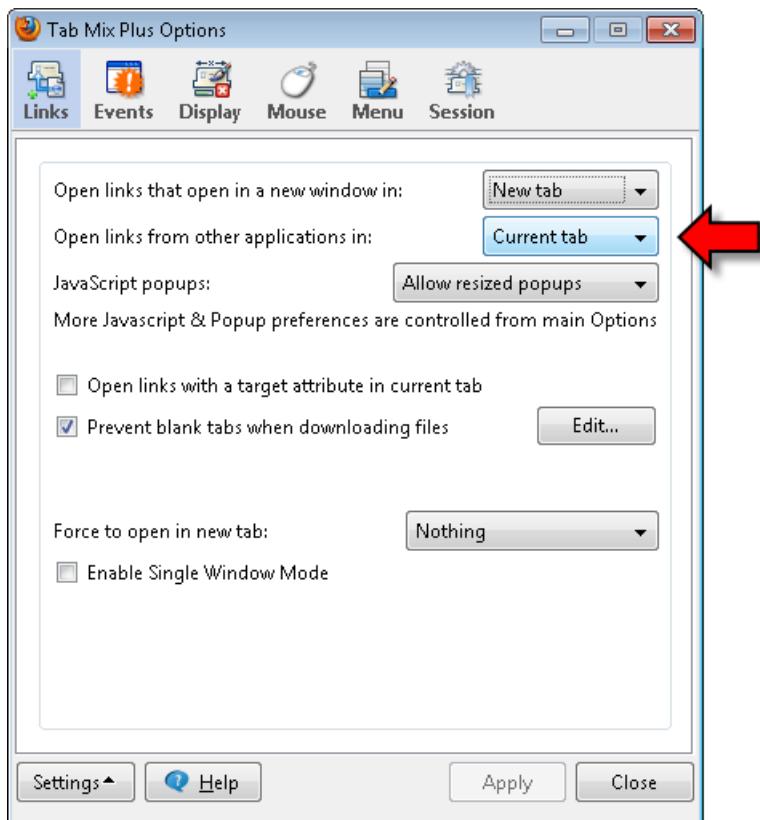
<https://addons.mozilla.org/en-US/firefox/addon/1122/>

Once installed, a new menu is available in the browser Tools pull-down. This menu is called "Tab Mix Plus Options..."



Refresh

From the tab window that appears, in the Links tab, an option called "Open links from other applications in:" can be set. Setting this to "Current tab" results in the browser tab being reused by Process Inspector for each coach shown which is more what we want to happen.



Logging

IBPM makes heavy use of JavaScript to describe logic that should be executed at runtime. Unfortunately, there is currently no source level debugger for this language available in IBPM. Because of this, it can be difficult to diagnose certain kinds of problems. One solution to this is to insert instrumentation code into the JavaScript. When reached, this code can log information externally to IBPM to allow the developer to see what is going on.

In the IBPM environment, a JavaScript variable called `log` is always in scope. This variable is an instance of a class which includes methods called:

- `info`
- `warn`
- `error`
- `debug` (note that `log.debug` does not appear to work in 7.5 and above)

Each of these functions take a string parameter. When reached, the string is appended into a log file called `SystemOut` in the directory `<profile>/server1/logs`.

For example, coding the following in JavaScript

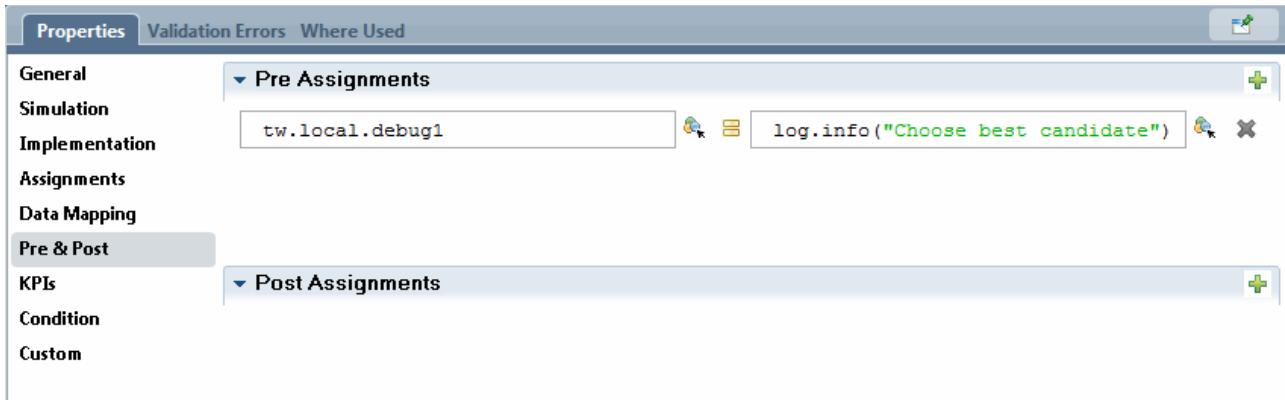
```
log.error("Hello World");
```

results in the following being appended to the log file:

```
[6/18/11 20:14:36:398 CDT] 00000079 wle_javascript E Hello World
```

A tail tool such as the popular [BareTail](#) or [LogExpert](#) can be used to follow log files.

One interesting technique for debugging processes is to place log statements in the pre or post assignments. This can be achieved by defining a junk variable (I called mine debug1) and then placing a pre or post assignment as follows:



What we are saying here is that we are assigning the junk variable the outcome of the `log()` function. The log function logs information and we ignore any result it may have returned.

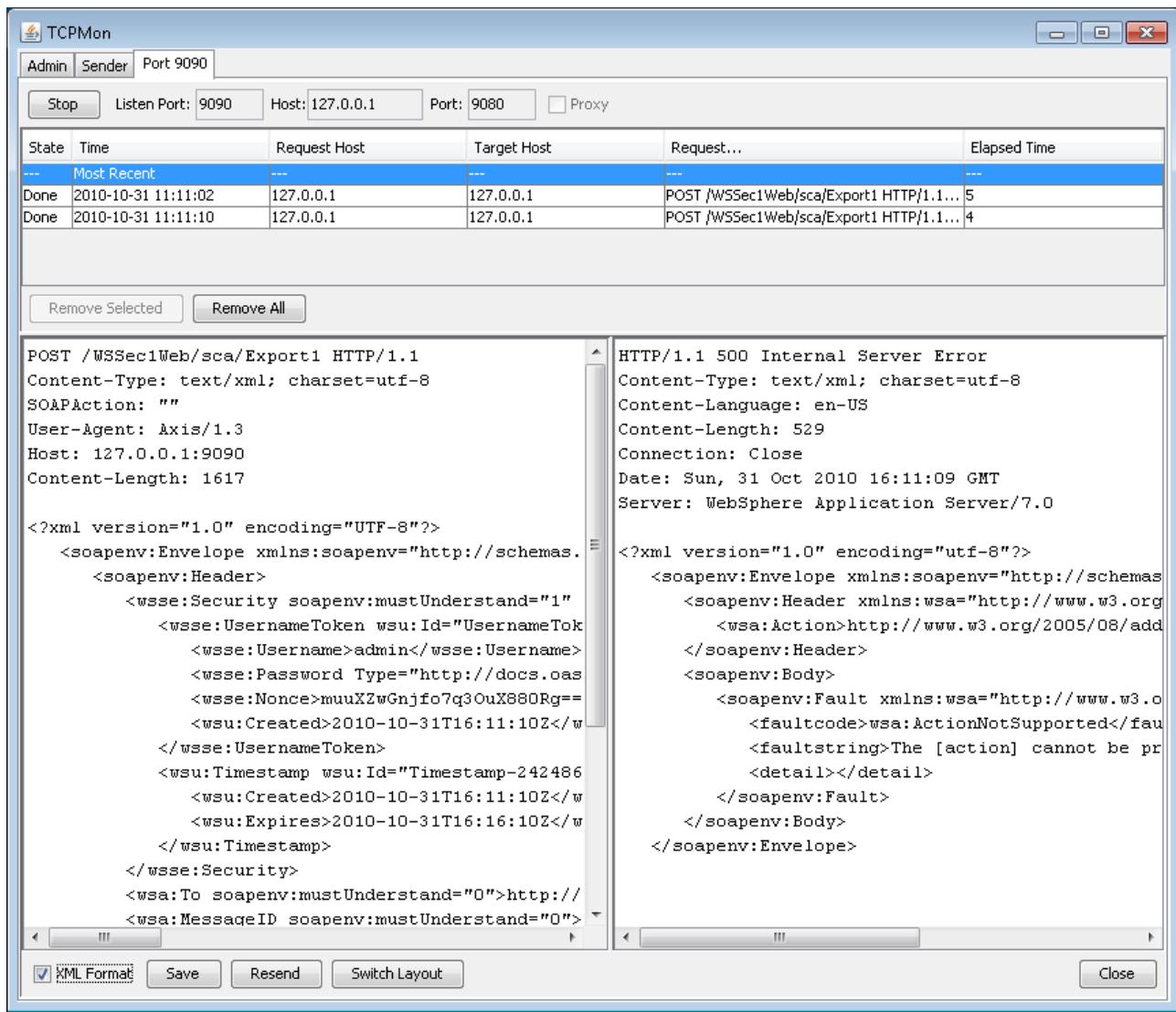
Log files can grow pretty quickly. If a problem is re-creatable, it is suggested that the old logs files be deleted or copied away before re-running a test. The log files that can be cleaned include:

- <profile>/logs/server1/*
- <profiles>logs/ffdc/*

Tracing Web Service SOAP traffic

Web Services requests are SOAP over HTTP. On occasion, one may want to examine the SOAP data traveling over the network. A useful tool to achieve this is TCPMon from Apache:

<http://ws.apache.org/commons/tcpmon/index.html>



A possible alternative to TCPMon is the tool called Fiddler. (see: <http://www.fiddler2.com>).

Working with IBM Defect Support

If a suspected defect is found within the product, IBM wants to hear about this so it can fix it. The way this is done is to raise a PMR (a defect report). When raising a defect, it is best to provide as much detailed information as possible. Some suggested practices for capturing the information needed to resolve the issue include:

- Providing a clean startup log

Shutdown the servers and delete all the log files in the <profile>/logs/server1 and <profile>/logs/ffdc directories. Recreate the problem and then ZIP up all the data contained within these two folders.

Raising defects with IBM

If a problem is believed to be a defect with IBPM itself, a PMR (and IBM problem report) should be raised. IBM defect support has a "must gather" set of logs that they require when submitting a

report. The details and instructions for these can be found here:

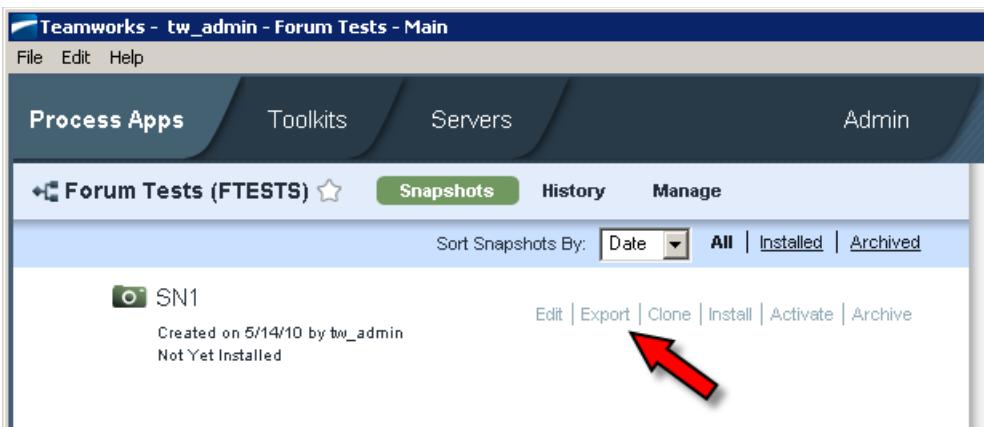
<http://www.ibm.com/support/docview.wss?uid=swg21440086>

Development

The actual tasks of building out a IBPM process includes constructing BPDs, services, reports, Coaches and other related artifacts. Generically, this is called development. This section touches on some of the areas associated with the act of developing a IBPM based solution.

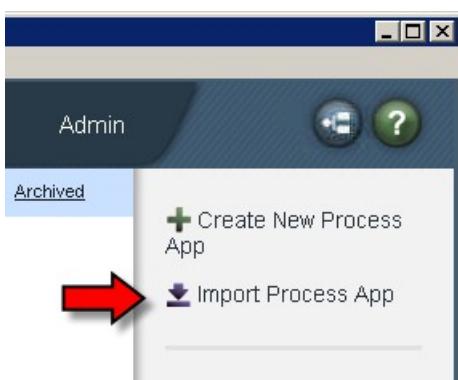
Sharing projects with others

It is not uncommon to want to share projects with others. In the IBPM environment, this is normally achieved without any work as all the artifacts are saved in the Process Center so that all users attached to the same Process Center can see the same views of the data. If multiple Process Centers are being used then it may, on rare occasion, be desirable to export a project from one Process Center and import it into another. This can be easily achieved with the export capability. First a snapshot must be created. On the Process Center page, selecting an application will show its snapshots. From here, an export operation can be performed.



This will prompt for a destination for a file that will contain the exported resources. The file suffix for exported files is ".twx".

To import a process, on the main page for the Process Apps in the Process Center, an Import Process App link can be found.



Selecting this produces a wizard for importing the project.

Naming conventions and recommendations

IBPM is pleasantly un-restrictive on names that may be given to artifacts. In fact, we can give multiple different types of artifacts the same name if desired. The type of artifact acts as the discriminator.

When an artifact is exposed for starting such as a BPD or a service, the name of that artifact is what shows up in the menu. As such, it is a good idea to give the artifact a name that you wish to publicly expose. For example, in the following diagram we have exposed three BPDs. The first two have poor names but the third gives an indication of what it does.

The screenshot shows the Lombardi Process Portal interface. On the left, there's a sidebar with 'My Tasks' (Inbox, History, Help Requests, Alerts, Suspended) and 'My Scoreboards' (My Performance, My Team Performance, Process Performance, SLA Overview). The main area is titled 'SearchResults' and contains a list of BPDs: BPD1, BPD4, and New Employee On Board. A large red arrow points to the 'New Employee On Board' item. Below this list is a table titled 'Instances' with one row: Instance Name (On Board:156) and Process Definition (On Board).

Instance Name	Process Definition
On Board:156	On Board

The instance name of a process is also exposed. The value of the Process Instance is set in the BPD overview and shows up on Process Portal tables.

Process App name recommendations

When a new Process App is created, an acronym value is supplied. This can be up to seven characters in length. My recommendation for a naming convention on these is:

[A-Z] YYMMDD

where A-Z is a single character and YYMMDD is the current year/month/day. For example, if I needed a new Process App, I might give it the name:

A110807

By using this technique, we have a good chance that we won't have a collision.

Documentation

When a BPD or a service is written it is still fresh in the writers mind and is believed to be able to be remembered in the future. That is never the case and the original writer may no longer be around in the future. Documenting the solution as it is built is vital. It may seem like wasted effort and, in many cases, the documentation is indeed never read again ... however ... if it is read and understood just once in the future, it can have paid for itself over and over again. Most of the artifacts associated with IBPM have a documentation section associated with them. This is a text input area where information can be entered that does not affect the operation of the solution. When a IBPM PD user subsequently visits the artifacts, the documentation previously entered can be re-read.

Recruiter

Overview Diagram Variables Tracking Process KPIs

Common

Name: Recruiter
Modified: Bill (Aug 31, 2013 9:59:51 PM)

This process plays the part of the recruiter. It is initiated by a request to find suitable candidates. Candidates are "pumped out" until a subsequent request to stop recruiting is received. If a candidate is entered, a "New Candidate" event is published.

Documentation: [\(Edit\)](#)

Events - Input

- Start Recruiter - A request to start recruiting. This is a process initiator event.
- Stop Recruiter - A request to stop

Advanced

Instance Name:
Due in:
Enable Tracking
Allow Projected Management:

Work Schedule

Time Schedule: 
Timezone:
Holiday Schedule

Exposing

Expose to Start:

At this time there is no known way to produce a report from the documentation but that normally acceptable. I'm not sure what it would mean to generate a report of parallel branches of a process flow and have them read serially in a PDF document.

Operations

After installing IBPM and building solutions to be deployed upon it, there will come a time when the environment must be managed for operations.

Configuration Files

IBPM uses some configuration files that are read during the Process Server and Process Center startup. These files contain XML content and are designed to be readable and editable in a text or XML editor. Only an educated runtime administrator should consider editing these files.

The files are located in the folder called:

```
<Install>/profiles/<profileName>/config/cells/<cellName>/nodes/<nodeName>/server  
/<serverName>/ [process-center|process-server]/config/system
```

The names of the files are:

00Static.xml	
50AppServer.xml	
60Database.xml	
80EventManager.xml	
98Database.xml	
99Local.xml	
99Sharepoint.xml	

The idea behind these files is that they contain settings for the operation of the server. All the files are read in numeric order based on the numerics at the start of the file name. It is allowable for one file to override the settings made in a previous file. Think of it as all the files being merged together in the numeric order and the result being a single XML document with replacements for previous entries.

These files are *not* designed to be edited by the deployment administrator. They can be examined to see what values are to be in effect. If changes are desired, it is strongly recommended to place those changes in the file called:

```
<Install>/<...path...>/100Custom.xml
```

This is assured to be the last file read. Changes in this file will override any of the settings in the other files.

Any changes made to files are not reflected until the servers are restarted.

When changes are made, it is extremely good practice to watch the logs of a server startup. If any problems are found in the parameter settings, that is where they will show up and should be corrected before continuing.

Unfortunately, despite the wealth of over-ridable entries in these files, very little published documentation exists on them so unless explicitly directed to make changes by IBM staff, these are probably best left alone. If changes are made, make sure that logs of what was changed, when, why and most importantly, any source emails or documentation from IBM on those changes should be recorded.

See also:

- TechNote - [Teamworks Config File Overview and Explanation](#) - 2010-08-09

Networking

The Process Center and Process Server products run on top of WAS. When running the products on a laptop, it is not uncommon to change networking characteristics. It has been found that if the network IP addresses change from under the server, bad things can happen. A solution to this is to create a localhost entry in the hosts file. On windows, this can be found in C:\Windows\System32\drivers\etc. It is also a good idea to determine your own machine's hostname through the hostname command and create an alias for that to be 127.0.0.1.

WAS Server

When IBPM is installed, a WAS profile is created that installs at <Install>/AppServer. This AppServer can be registered as a Windows service. This shows up as an entry called:

IBM WebSphere Application Server v8.0 - ProcessCenter01

Stopping the server

A WAS server can be stopped by running the command stopServer <serverName>. When run, a dialog appears prompting for the userid/password to be used to allow shut down. For convenience during development, this pair of parameters can be scripted with the "-user <userName> -password <password>" options. When a server is defined on Windows, a stop entry is added to the menu of available commands.

From 8.5 onwards with the arrival of the mandatory network deployment model, we can stop and start the servers through the nodeagent.

Windows Services

WAS can be configured as a Windows Service to allow it to start at machine boot time. The service can be registered by the WAS supplied command called "WASService".

Port Numbers

TCP/IP port numbers are the endpoints for TCP/IP communications. When combined with hostname or IP address, the port number provides the unique endpoint for a browser or service request. The port numbers for the servers:

- SOAP – Look in the console for a line which reads:

JMXSoapAdapte A ADMC0013I: The SOAP connector is available at port nnnn

The port number for SOAP is listed on that line.

See also:

- TechNote - [WebSphere Lombardi Edition 7.1 Port information](#) - 2010-09-29

WAS Admin Console

WAS has an administrative console known as the "Integrated Solutions Console" or ISC. It can be

accessed from:

<https://localhost:9044/ibm/console/logon.jsp>

The default administrative userid is admin/admin.

JDBC Resources

A set of predefined JDBC resources are created when IBPM is installed.

JNDI name	Description
jdbc/TeamWorksDB	Used by the Process Center to access the repository of artifacts for the common shared model.
jdbc/PerformanceDB	Used by the Performance Data Warehouse to hold performance information.
jdbc/ProcessDB	Used by Process Servers to store runtime information used during the execution of processes.
jdbc/TwqlDB	<TBD>

WAS Security

Here is an un-tested recipe for programmatically creating WAS based users:

```
com.ibm.websphere.wim.Service service =
new com.ibm.websphere.wim.client.LocalServiceProvider(null);
commonj.sdo.DataObject root =
com.ibm.websphere.wim.util.SDOHelper.createRootDataObject();
commonj.sdo.DataObject entity =
com.ibm.websphere.wim.util.SDOHelper.createEntityDataObject(root,
null,DO_PERSON_ACCOUNT);
entity.set("uid", "kolban");
entity.set("password", "mypassword");
commonj.sdo.DataObject result = service.create(root);
```

The wsadmin command

WAS provides a command called "wsadmin" that is the primary tool for changing WAS configuration settings. It can be invoked from the command line interactively or by being passed a file of command to execute. It supports a variety of scripting languages including JACL and Jython but Jython is the one that is recommended.

See also:

- Redbook - [WebSphere Application Server v8.5 Administration and Configuration Guide for the Full Profile](#) – SG24-8056-01
- developerWorks - [IBM WebSphere Developer Technical Journal: Administrative Command Framework Programming Guide for WebSphere Application Server](#) - 2006-10-25

Java programming for admin commands

There are times when we might wish to perform WAS admin commands from within a Java environment. WAS provides a framework to achieve this.

First there is the object known as the CommandMgr. From the CommandMgr, we create an instance of an AdminCommand.

If we want to know what commands are available, the cmdMgr.listAllCommands() will return a collection.

Once we know which command we want to execute, we can create an AdminCommand instance with:

```
AdminCommand adminCommand = cmdMgr.createCommand("CommandName");
```

From an adminCommand, we can ask what the parameters are for it. We can get these through:

```
List parameters = adminCommand.listParameterName();
```

We can then set a parameter with:

```
adminCommand.setParameter("Name", value);
```

Next we can execute the command with:

```
adminCommand.execute();
```

And finally retrieve the results with:

```
Collection results = adminCommand.getResult();
```

Changing Passwords

- [Changing the database password for a WebSphere Lombardi Edition Version 7.1 or 7.2 full installation](#) – 2011-07-14
- [How to change the database password on WebSphere Lombardi Edition \(WLE\) 7.1 Simple Install](#) – 2011-02-04
- [How do you change the tw_admin password in Teamworks 7.0.x?](#) - 2010-06-01

File Structures

By default, IBPM installs itself at C:\IBM\Lombardi7.

On a Windows 7 or Vista environment, IBPM requires to be started with Administrative permissions. Starting the Server can be achieved through the script provided at:

```
C:\IBM\Lombardi7\process-center\bin\startProcessCenter.cmd
```

The console log file for the WAS server can be found in the directory called:

```
C:\IBM\Lombardi7\process-center\logs\twprocsvr
```

Operational Databases

IBPM makes use of a number of databases that host data for its own operation. Because IBPM is an IBM product, it is recommended to use DB2 as the database management system. More IBMers are going to be familiar with DB2 than alternative database systems. An instance of DB2 is provided with IBPM to provide the necessary Database environment. These databases used for IBPM operations should **not** be used for hosting application data. It is strongly recommended to separate operational data from application oriented data.

If we attach a table browser to the database we will see a lot of tables. Here is a list of some of the more interesting ones:

Table name	Description
LSW_BPD	Information about a BPD. Seems to have a new row written per save.
LSW_USR	Users defined to the system.
LSW_BPD_INSTANCE_VARIABLES	This table contains a row per variable per BPD instance that is exposed as searchable. A column in the table called BPD_INSTANCE_ID can be used to retrieve the BPD ID after finding a variable that has a value of interest.
LSW_TASK	Information about each task in the system.

Process Server Database

The Process Server Database is used to manage the state of in-flight processes and tasks. Although there is no documentation on the tables contained within this database nor should any assumptions be made about its content, it can be instructive to examine the content. It is **strongly** advised never to change the content of tables in this database unless explicitly instructed to do so by IBM.

Performance Data Warehouse Database

The Performance Data Warehouse database tracks information relating to the performance of the processes run by the Process Server. This includes historical data kept over time about the outcomes of completed processes.

Defining custom databases

Within an IBPM solution, there are times when we will want to access data contained in application databases. These can be accessed through SQL statements within the solution. Before we can use these databases, they must first be defined to the IBPM environment as WAS JDBC definitions. The instructions for achieving this are well documented in the WAS documentation and there are additional TechNotes and DeveloperWorks articles available for reference. A summary is reproduced here.

JDBC resources are defined in the WAS runtime. The WAS Admin Console can be used to manually define such definitions. In addition, scripting can be used to create the same definitions if desired.

To launch the WAS Admin Console, open a browser to:

<https://localhost:9044/ibm/console/logon.jsp>

Login as admin/admin or another WAS administrative userid.

First we must create a JDBC provider entry that points to our new Database server.

Navigate to the JDBC providers section found at Resources → JDBC → JDBC providers



Click the New button to begin a new definition.

Select the definitions appropriate to the database type being defined. The following shows those for DB2.

Create a new JDBC Provider

Step 1: Create new JDBC provider

Step 2: Enter database class path information

Step 3: Summary

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.

Scope
cells:LombardiCell01:nodes:ProcessCenter01:servers:twprocsvr

* **Database type**
DB2

* **Provider type**
DB2 Using IBM JCC Driver

* **Implementation type**
XA data source

* **Name**
DB2 Using IBM JCC Driver (XA)

Description

Two-phase commit DB2 JCC provider that supports JDBC 4.0 using the IBM Data Server Driver for JDBC and SQLJ. IBM Data Server Driver is the next generation of the DB2 Universal JCC driver. Data sources created under this provider support the use of XA to perform 2-phase commit processing. Use of JDBC driver type 2 on WebSphere Application Server for Z/OS is not supported for data sources created

Next **Cancel**

Paths to the local driver files for the database server may need to be entered:

Create a new JDBC Provider

Step 1: Create new JDBC provider

Step 2: Enter database class path information

Step 3: Summary

Enter database class path information

Set the environment variables that represent the JDBC driver class files, which WebSphere(R) Application Server uses to define your JDBC provider. This wizard page displays the file names; you supply only the directory locations of the files. Use complete directory paths when you type the JDBC driver file locations. For example: C:\SQLLIB\java on Windows(R) or /home/db2inst1/sqllib/java on Linux(TM).

If a value is specified for you, you may click Next to accept the value.

Class path:

`\${DB2_JCC_DRIVER_PATH}/db2jcc4.jar
`\${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
`\${DB2_JCC_DRIVER_PATH}/db2jcc_license_cisuz.jar

Directory location for "db2jcc4.jar, db2jcc_license_cisuz.jar" which is saved as WebSphere variable \${DB2_JCC_DRIVER_PATH}
C:\Program Files (x86)\IBM\SQLLIB\java

Native library path

Directory location which is saved as WebSphere variable \${DB2_JCC_DRIVER_NATIVEPATH}
C:\Program Files (x86)\IBM\SQLLIB\BIN

Previous **Next** **Cancel**

Now that a JDBC provider has been defined, we can now create a JDBC Data source definition which names a Database hosted by that kind of server.

Navigate to the JDBC Data sources section found at Resources → JDBC → Data sources

The screenshot shows the 'Integrated Solutions Console' interface. The left sidebar has a tree view under 'Resources'. The 'JDBC' node is expanded, and its 'Data sources' child node is selected, highlighted with a red arrow. Other nodes like 'Schedulers', 'Object pool managers', and 'Data sources (WebSphere Application Server V4)' are also visible.

Click the New button to begin a new resource definition.

This screenshot shows the 'Create a data source' wizard at Step 1: Enter basic data source information. The left sidebar lists steps from 1 to 5. The main panel is titled 'Enter basic data source information' and contains fields for 'Scope' (set to 'cells:LombardiCell01:nodes:ProcessCenter01:servers:twprocsvr'), 'Data source name' (set to 'TEST DB'), and 'JNDI name' (set to 'jdbc/TEST').

This screenshot shows the 'Create a data source' wizard at Step 2: Select JDBC provider. The left sidebar lists steps from 1 to 5. The main panel is titled 'Select JDBC provider' and contains a note about selecting a provider. It shows two radio button options: 'Create new JDBC provider' (unchecked) and 'Select an existing JDBC provider' (checked), with 'DB2 Using IBM JCC Driver (XA)' selected in a dropdown menu.

Create a data source

Create a data source

Step 1: Enter basic data source information Step 2: Select JDBC provider → Step 3: Enter database specific properties for the data source Step 4: Setup security aliases Step 5: Summary	Enter database specific properties for the data source Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through the datasource.										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Name</th> <th style="text-align: left;">Value</th> </tr> </thead> <tbody> <tr> <td>* Driver type</td> <td style="background-color: #ffffcc;">4</td> </tr> <tr> <td>* Database name</td> <td style="background-color: #ffffcc;">TEST</td> </tr> <tr> <td>* Server name</td> <td style="background-color: #ffffcc;">localhost</td> </tr> <tr> <td>* Port number</td> <td style="background-color: #ffffcc;">50000</td> </tr> </tbody> </table> <p><input type="checkbox"/> Use this data source in container managed persistence (CMP)</p>	Name	Value	* Driver type	4	* Database name	TEST	* Server name	localhost	* Port number	50000
Name	Value										
* Driver type	4										
* Database name	TEST										
* Server name	localhost										
* Port number	50000										

[Previous](#) [Next](#) [Cancel](#)

Create a data source

Create a data source

Step 1: Enter basic data source information Step 2: Select JDBC provider Step 3: Enter database specific properties for the data source → Step 4: Setup security aliases Step 5: Summary	Setup security aliases Select the authentication values for this resource.												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Authentication alias for XA recovery</td> <td style="width: 33%;"><input type="text" value="ProcessCenter01/db2admin"/></td> <td style="width: 33%;"></td> </tr> <tr> <td>Component-managed authentication alias</td> <td><input type="text" value="ProcessCenter01/db2admin"/></td> <td></td> </tr> <tr> <td>Mapping-configuration alias</td> <td><input type="text" value="(none)"/></td> <td></td> </tr> <tr> <td>Container-managed authentication alias</td> <td colspan="2"><input type="text" value="ProcessCenter01/db2admin"/></td> </tr> </table> <p>Note: You can create a new J2C authentication alias by accessing one of the following links. Clicking on a link will cancel the wizard and your current wizard selections will be lost.</p> <p style="color: blue;">Global J2C authentication alias Security domains</p>	Authentication alias for XA recovery	<input type="text" value="ProcessCenter01/db2admin"/>		Component-managed authentication alias	<input type="text" value="ProcessCenter01/db2admin"/>		Mapping-configuration alias	<input type="text" value="(none)"/>		Container-managed authentication alias	<input type="text" value="ProcessCenter01/db2admin"/>	
Authentication alias for XA recovery	<input type="text" value="ProcessCenter01/db2admin"/>												
Component-managed authentication alias	<input type="text" value="ProcessCenter01/db2admin"/>												
Mapping-configuration alias	<input type="text" value="(none)"/>												
Container-managed authentication alias	<input type="text" value="ProcessCenter01/db2admin"/>												

[Previous](#) [Next](#) [Cancel](#)

Create a data source

Step 1: Enter basic data source information Step 2: Select JDBC provider Step 3: Enter database specific properties for the data source Step 4: Setup security aliases → Step 5: Summary	<h3>Summary</h3> <p>Summary of actions:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #f2f2f2;">Options</th> <th style="background-color: #f2f2f2;">Values</th> </tr> </thead> <tbody> <tr> <td>Scope</td> <td>cells:LombardiCell01:nodes:ProcessCenter01:servers:twprocsrv</td> </tr> <tr> <td>Data source name</td> <td>TEST DB</td> </tr> <tr> <td>JNDI name</td> <td>jdbc/TEST</td> </tr> <tr> <td>Select an existing JDBC provider</td> <td>DB2 Using IBM JCC Driver (XA)</td> </tr> <tr> <td>Implementation class name</td> <td>com.ibm.db2.jcc.DB2XADataSource</td> </tr> <tr> <td>Driver type</td> <td>4</td> </tr> <tr> <td>Database name</td> <td>TEST</td> </tr> <tr> <td>Server name</td> <td>localhost</td> </tr> <tr> <td>Port number</td> <td>50000</td> </tr> <tr> <td>Use this data source in container managed persistence (CMP)</td> <td>false</td> </tr> <tr> <td>Authentication alias for XA recovery</td> <td>ProcessCenter01/db2admin</td> </tr> <tr> <td>Component-managed authentication alias</td> <td>ProcessCenter01/db2admin</td> </tr> <tr> <td>Mapping-configuration alias</td> <td>(none)</td> </tr> <tr> <td>Container-managed authentication alias</td> <td>ProcessCenter01/db2admin</td> </tr> </tbody> </table>	Options	Values	Scope	cells:LombardiCell01:nodes:ProcessCenter01:servers:twprocsrv	Data source name	TEST DB	JNDI name	jdbc/TEST	Select an existing JDBC provider	DB2 Using IBM JCC Driver (XA)	Implementation class name	com.ibm.db2.jcc.DB2XADataSource	Driver type	4	Database name	TEST	Server name	localhost	Port number	50000	Use this data source in container managed persistence (CMP)	false	Authentication alias for XA recovery	ProcessCenter01/db2admin	Component-managed authentication alias	ProcessCenter01/db2admin	Mapping-configuration alias	(none)	Container-managed authentication alias	ProcessCenter01/db2admin
Options	Values																														
Scope	cells:LombardiCell01:nodes:ProcessCenter01:servers:twprocsrv																														
Data source name	TEST DB																														
JNDI name	jdbc/TEST																														
Select an existing JDBC provider	DB2 Using IBM JCC Driver (XA)																														
Implementation class name	com.ibm.db2.jcc.DB2XADataSource																														
Driver type	4																														
Database name	TEST																														
Server name	localhost																														
Port number	50000																														
Use this data source in container managed persistence (CMP)	false																														
Authentication alias for XA recovery	ProcessCenter01/db2admin																														
Component-managed authentication alias	ProcessCenter01/db2admin																														
Mapping-configuration alias	(none)																														
Container-managed authentication alias	ProcessCenter01/db2admin																														
Previous Finish Cancel																															

See also:

- Database Integration
- TechNote - [1444016 - WebSphere Lombardi Edition v7.1 – Configuring a Data Source](#)
- DeveloperWorks - [Configuring a data source in WebSphere Lombardi Edition V7.1](#) - 2010-10-13

SI Bus Resources

With WAS being an implementation of Java EE, it must provide an implementation of Java Messaging Service (JMS). WAS implements JMS through a native messaging and queuing provider called System Integration Bus (SI Bus). IBPM creates two instances of SI Buses. These are:

- PROCSVR.<Cell Name>.Bus

Name	Type
EventMgrControlTopicDestination	Topic
TWClientTopicDestination	Topic
cacheTopicDestination	Topic
eventerrorqueueDestination	Queue
eventmgrOpTopicDestination	Topic
eventqueueDestination	Queue

- PERFDW.<Cell Name>.Bus

JMS Resources are also created

jms/eventqueue	
jms/eventerrorqueue	
jms/ViewManagerQueue	
jms/RepresentationManagerQueue	
jms/PostLoadCalculationManagerQueue	
jms/DataDefLoaderQueue	
jms/DataDefLoaderErrorQueue	

See also:

- [RedBook - WebSphere Application Server V7 Messaging Administration Guide – 2009-07-07](#)

Cleaning/removing completed processes

When a BPD instance has completed, the record of that BPD instance is not deleted from the databases. As such, over time, the databases will become full of historical information.

A wsadmin command called `BPMProcessInstanceCleanup` is provided that will clean/remove data associated with previously ended process instances. The syntax for this command is:

- `-containerAcronym <acronym>` - The acronym for the Process App containing the process instances to be removed.
- `-containerSnapshotAcronym <acronym>` - The acronym of the snapshot containing the process instance details to be removed.
- `-instanceStatus <Status>` - A filter describing the state of process instances to be deleted. This can be one of:
 - COMPLETED
 - FAILED
 - CANCELLED
 - ALL
- `-instanceID <instanceIDs>` - A list of one or more instanceIds that are to be deleted.
- `-endAfterLocal ...` - A date/time that says which instances to delete after a date
- `-endBeforeLocal ...` - A date/time that say which instances to delete before a date
- `-outputFile <outputFileName>` - The name of a file which will be used to record the outcome of the deletions including which instances were deleted.

The wsadmin command called `BPMShowProcessApplication` can be used to list the details of a Process Application as known to IBM BPM. This command takes a parameter which is the acronym id of the Process Application to be shown.

eg.

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym MyApp]')
```

An example output of this command is:

```

Name: 801Tests
Acronym: T801
Description:
Toolkit: false
Tracks:

Track Name: Main
Track Acronym: Main
Default: true

Tip:
    Created On: 2012-11-22 12:12:53.8
    Created By: User.9
    State: State[Inactive]
    Capability: Capability[Standard]
    No of running instances: 0

List of Snapshots:
    Name: SN1
    Acronym: SN1
    Created On: 2012-11-22 12:12:53.8
    Created By: User.9
    Is Default: false
    State: State[Inactive]
    Capability: Capability[Standard]
    No of running instances: 0

```

In addition to the previous recipe, IBPM provides a stored procedure called `LSW_BPD_INSTANCE_DELETE` that clears historic data.

The signature of this procedure looks as follows:

```
CREATE PROCEDURE LSW_BPD_INSTANCE_DELETE(IN bpdInstanceId DECIMAL(12,0))
```

Here is a Java application that will delete such instances:

```

package com.kolban.wle;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class CleanBPD {
    public static void main(String[] args) {
        CleanBPD cleanBPD = new CleanBPD();
        cleanBPD.run();
    }

    private void run() {
        System.out.println("Running");
        try {
            Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
            String url = "jdbc:db2://localhost:50000/PROCDB";
            String userid = "db2admin";
            String password = "db2admin";
            Connection con = DriverManager.getConnection(url, userid, password);
            PreparedStatement queryStmt = con.prepareStatement("SELECT BPD_INSTANCE_ID FROM
LSW_BPD_INSTANCE WHERE EXECUTION_STATUS = 2");
            CallableStatement deleteStmt = con.prepareCall("{CALL LSW_BPD_INSTANCE_DELETE(?)}");
            ResultSet rs = queryStmt.executeQuery();
            while(rs.next())
            {
                int bpdId = rs.getInt("BPD_INSTANCE_ID");
                System.out.println("Deleting BPD: ID = " + bpdId);
                deleteStmt.setInt(1, bpdId);
                deleteStmt.execute();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

See also:

- Calling stored procedures

Adding and removing Process Servers from Process Center

Process Center is the hub of knowledge about process applications while Process Servers act as the run-time environments for executing work. When a new Process Server is created, it initially knows nothing about any available Process Servers. In order for a Process Center to know about the Process Server, the Process Server's configuration needs to be changed to refer to the target Process Center. The configuration file called `99Local.xml` contains this information. An example of the relevant section is as follows:

```
<repository-server-url>/ProcessCenter</repository-server-url>
<repository-server-user-auth-alias></repository-server-user-auth-alias>
<repository-server-designated-user-auth-alias>BPMAuthor_Auth_Alias</repository-server-designated-
user-auth-alias>
<repository-server-interval>-1</repository-server-interval>
```

The highlighted lines are the ones that need changed.

The `repository-server-url` should be changed to the URL of the ProcessCenter while the `repository-server-interval` should have a value of 10 (just not -1).

For example:

```
<repository-server-url>http://localhost:9080/ProcessCenter</repository-server-url>
<repository-server-user-auth-alias>myAlias</repository-server-user-auth-alias>
<repository-server-designated-user-auth-alias>BPMAuthor_Auth_Alias</repository-server-designated-
user-auth-alias>
<repository-server-interval>10</repository-server-interval>
```

The name of the server as shown in Process Center can also be found in the `99Local.xml` file under the `<server-name>` section:

```
<server-name>envName1</server-name>
<server-description>A running process server</server-description>
<server-host>localhost</server-host>
<server-port>9081</server-port>
<environment-type>Test</environment-type>
```

Deployment of applications to servers

Process Applications can be deployed to multiple Process Server instances. In order to deploy an application, a snapshot of the application must first be taken. It is the content of the snapshot that is deployed. Changes made to the application after the snapshot will not be reflected in the deployed application until after a new snapshot is taken and that instance of the snapshot deployed.

See also:

- Migration of in-flight BPMN process instances

Offline Application Deployment

During development, the applications are built and tested in the Process Center environment with its local/unit test run-time. However, when it comes time to deploy to production, many customers wish to perform manual install of such solutions directly into their Process Servers as opposed to a dynamic push out from a Process Center. There are many reasons for this but a primary one is that an exact copy of the application is desired so that the system can be recreated as needed from artifacts stored on the file system. Another common reason for manual deployment is that it can be scripted and supplied to operations staff for production installation and management. The script

and associated artifacts can then be logged as the system of record for the production system.

In order to perform a manual deployment of a solution, we must first export the package that represents the solution. This can be done from the Process Center console. Before we can do this, we need to define our target environment as an "offline server".

In Process Center, switch to the Servers tab and click the "Add a New Offline Server" button:

The screenshot shows the 'Servers' tab selected in the top navigation bar. On the left, there's a list of servers: 'PS1 (win7-x64)' (TEST - Status: Connected?) and 'PS1-offline' (PRODUCTION - Status: Offline). To the right of the servers is a sidebar with a 'Servers Page' section containing a brief description of connected servers and a link to 'Managing Process Servers'. At the top right of the main content area, there's a blue button labeled '+ Add a New Offline Server' with a red arrow pointing to it. The bottom right corner of the page displays the 'IBM | Process Center' logo.

A dialog will appear prompting for details of the new (offline) server:

The dialog box has a title bar 'Create New Server' with a close button. It contains three input fields: 'Server Name' with 'PS1-offline' entered, 'Environment Type' set to 'Production', and a 'Description' field with the text 'My tests offline server'. Below the description field is a rich text editor toolbar with buttons for bold, italic, underline, font size (12pt), and alignment. At the bottom right of the dialog is a 'Create' button.

Once completed, it will appear in the list:

The screenshot shows the 'Servers' tab in the IBM Process Center interface. There are two entries listed:

- PS1 (win7-x64)**: TEST - Status: Connected? (This server may not be available right now.)
- PS1-offline**: PRODUCTION - Status: Offline

On the right side, there is a sidebar with the following content:

- Add a New Offline Server**
- Servers Page**: A brief description of connected Process Servers.
- [Managing Process Servers](#)

At the bottom right of the main content area, it says **IBM | Process Center**.

In the Process Apps tab for the snapshot of the application that is to be deployed, click the Install button:

The screenshot shows the 'Process Apps' tab in the IBM Process Center interface. It displays a list of snapshots for the application **App01 (APP01)**:

- Current**: Last changed on 11/12/12 by admin
- SN1** (New): Created on 11/12/12 by admin, Not Yet Installed to Process Server

For the **SN1** snapshot, there are two buttons at the end of its row: **Export** and **Install**. A red arrow points to the **Install** button. To the right of the snapshot list, there is a sidebar with the following content:

- Create New Snapshot**
- Snapshots**: A brief description of what snapshots are used for.
- [Managing snapshots](#)

At the bottom right of the main content area, it says **IBM | Process Center**.

Next select the offline server and then the "Create installation package" button:



After a few moments, the details of the Snapshots for the Process App will change to show that a deployment package is available:

Snapshots	History	Manage
Sort Snapshots By: Date All <u>Installed</u> Deployed Archived		
<input checked="" type="radio"/> Current Last changed on 11/12/12 by admin		
<input checked="" type="radio"/> SN1 (New) Created on 11/12/12 by admin Deployment Package Available: PS1-offline	Export Install	<p>Create New Snapshot</p> <p>Snapshots</p> <p>A snapshot captures the state of the library items within a process application or toolkit at a specific point in time. Snapshots usually represent a milestone or are used for playbacks or for installation.</p> <p>Managing snapshots</p>

At this point, Process Center has *built* the package for deployment. Now we have to retrieve the package from the Process Center. This is achieved from the Windows DOS command prompt. Open a DOS window and change directory to:

The Wsadmin command called `BPMExtractOfflinePackage` can be used to create an installation package. It has the following options:

- `containerAcronym` – The acronym of the application to be extracted
- `containerSnapshotAcronym` – The name of the snapshot to be extracted

- `containerTrackAcronym` – The name of the track within the snapshot to be extracted. This parameter is also mandatory. The default value for the default track is "Main".
- `serverName` – The name of the offline Process Server instance
- `outputFile` – The name of the file to be used to hold the extracted installation package

Note: Another useful command is called `BPMShowProcessApplication` which will list the details of a named Process App.

```
print AdminTask.BPMShowProcessApplication('[-containerAcronym APP01]')
Name: App01
Acronym: APP01
Description:
Toolkit: false
Tracks:

    Track Name: Main
    Track Acronym: Main
    Default: true

    Tip:
        Created On: 2012-11-12 11:49:49.159
        Created By: User.9
        State: State[Inactive]

    List of Snapshots:
        Name: SN1
        Acronym: SN1
        Created On: 2012-11-12 11:49:49.159
        Created By: User.9
        State: State[Inactive]
```

For example, the following command will extract an installation package:

```
AdminTask.BPMExtractOfflinePackage('[-containerAcronym APP01 -containerSnapshotAcronym SN1
-containertTrackAcronym Main -serverName PS1-offline -outputFile C:/temp/APP01.zip]')
```

The result of running this command will be a new file that contains the package for the application. This should be copied to the machine hosting the Process Server on which it is to be installed. To deploy the application to the server, use another command. This command is called `BPMInstallOfflinePackage`. This takes as input the package file name and will deploy it for execution.

For example:

```
AdminTask.BPMInstallOfflinePackage('[-inputFile C:/temp/APP01.zip]')
```

See also:

- DeveloperWorks - [Deploying IBM Business Process Manager packages to offline process servers](#) - 2013-03-27

Un-Deployment of applications from servers

There may come a time when a previously deployed application is not desired to be run again. At this point you may wish to remove the application from the Process Server. To achieve that task, first de-activate the Process App from the Process Admin Console:

The screenshot shows the IBM Process Admin Console interface. At the top, there are tabs for 'Process Admin Console', 'Server Admin', 'Process Inspector', and 'Installed Apps'. Below these are sub-tabs: 'DeployTests (DEPLOY1) - SN1 (Active)', 'Exposing', 'Role Bindings', 'Servers' (which is highlighted in green), 'Environment Vars', and 'Event Subscriptions'. On the right side, there is a user status bar with 'Logged in as admin | Preferences | Log out' and a help icon. A context menu is open over the 'Servers' tab, listing options: 'Deactivate Application' (with a red arrow pointing to it), 'Migrate Inflight Data', 'Sync Settings', and 'Update Tracking Definitions'.

Once the snapshot has been de-activated, it can be deleted using the wsadmin command:

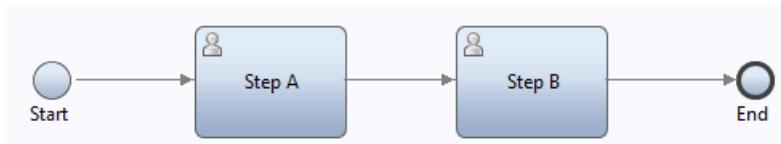
```
AdminTask.BPMDeleteSnapshot('[-containerAcronym DEPLOY1  
-containerSnapshotAcronyms [SN1]]')
```

This will remove the Snapshot from the server.

Modification of Process Instance data

Consider the notion that when a process instance exists, it maintains its own state over the lifetime of that process. That state information may be initially supplied data or it may be data gathered during the further operation of the process. In either case, it is the process instance that owns that data.

From an operational perspective, there may be times where we would want to come in and change the state of the data of currently in-flight processes. Another way of saying this is that if we look at the following sample process:



Imagine that it has a private variable called "myData". How would we go about changing the value of "myData" within the process?

Our first attempt may be to look for a system API that given the process instance ID (PIID) we could get and set process variables. Sadly, such an API does not exist at the process level. So, what options do we have?

The first option is to use an "event sub-process" which is triggered by an incoming message. Imagine we created a UCA called "Process Command" that had input data that looked like:

- piid (String)
- responseId (String)
- command (String)
- name (String)
- value (String)

We could use the "piid" as the correlation value using the Process Instance ID as the key. The command could be "set" (for example) and the "name" and "value" parameters could define the name of the variable to change and the new value it should have. Example code for this might be:

```
if (tw.local.processCommand.command == "set") {
    tw.local[tw.local.processCommand.name] = tw.local.processCommand.value;
}
```

See also:

- Modeling Event Sub-processes

Performance

As the product operates, we can ask ourselves about its performance. Performance has a number of qualities including throughput and latency.

See also:

- RedBook - [IBM Business Process Manager \(BPM\) 7.5 Performance Tuning and Best Practices](#) – REDP-4784-00 - 2012-01-27
- TechNote: [How to tune your WebSphere Lombardi Edition Servers for performance](#) - 1446648 - 2011-04-08
- TechNote: [Slow performance might result with WebSphere Lombardi Edition and DB2](#) - 1472705 - 2011-03-21
- TechNote: [TeamWorks Performance Tuning Tips](#) - 1439610 – 2011-02-04
- Webcast replay: [WebSphere Lombardi Edition - Developing for Performance and Scalability](#) – 2011-05-24
- TechNote: [Understanding and Tuning the Event Manager](#) - 2011-01-29

The Event Manager - Tuning attributes

A key component inside the internals of IBM BPM is the Event Manager. The Event Manager is used to schedule the execution of parts of a Process Application. This includes:

- Handling UCA invocations
- Handling BPD notifications
- Executing BPD system lane activities
- Executing BPD timer events
- maybe more ...

The Event Manager manages the queues of work to be performed. Logically, there are two types of queues. These are the Synchronous queues (can be multiple) or the Asynchronous queue (a single queue). Work placed on the Asynchronous queue is ready for immediate execution while work placed on a Synchronous queue can only be executed once any previous work from that same queue has completed.

The XML configuration files have a number of properties in them related to performance. These can be found in the 80EventManager.xml file. These include:

- enabled – This is a boolean valued property and is hence either true or false. It controls whether or not the Event Manager is "active" in this Process Server instance. The default is true. If set to false, then no event processing is performed. It is not clear when/why this would ever be used.
- heartbeat-period
- heartbeat-expiration
- loader-long-period
- loader-short-period
- loader-advance-window
- sync-queue-capacity
- async-queue-capacity
- bpd-queue-capacity

- system-queue-capacity
- min-thread-pool-size
- max-thread-pool-size
- re-execute-limit
- kick-on-schedule

See also:

- Process Admin - Event Manager
- TechNote - [Understanding and Tuning the Event Manager](#) – 2011-01-29
- RedBook - [IBM Business Process Manager V7.5 Performance Tuning and Best Practices](#) – 2012-04-11
- TechNote - [Webcast replay: WebSphere Lombardi Edition - Developing for Performance and Scalability](#) - 2011-05-24

Monitoring using the Process Admin Console

The Process Admin Console has a section in it titled Monitoring:



As Process Server executes, it is constantly collecting operational statistics about the execution of the server as a whole. This information can then be examined to determine where time and resources are being spent during execution. There are two subsections contained within Monitoring and these will be discussed individually.

Process Admin Console – Monitoring > Instrumentation

The **Monitoring > Instrumentation** area shows information about the internals of the IBM BPM product's operation. The Refresh button will update the screen with new information captured since the last refresh. The Reset button will allow us to reset the statistics to zero. The Start Logging button will start recording the logged information to a file on the file system. Unfortunately, it is not known how to use/analyze this file data.

Monitoring > Instrumentation

[Start Logging](#)

[Refresh](#)

[Reset](#)

Automatically refresh every [30 seconds](#)

Name	Count/Value	In Process	Average Duration (ms)	Moving Average Duration (ms)	Total (ms)
(+) BPD					
(+) Instances					
BPD Instances Completed	0				
BPD Instances Failed	0				
BPD Instances Resumed	0				
BPD Instances Started	0				
BPD Instances Terminated	0				
Cache					
(+) Connectors					
(+) Webservices					
(+) Inbound					
WebService Call	0	0	0.00		0.00
(+) Outbound					
WebService Call	0	0	0.00		0.00
(+) Event Manager					
EventManager Lag (seconds)	0		0.00		0.00
Sync Queues Owned	0		0.00		0.00
Tasks Added	0		0.00		0.00
Tasks Cancelled	0		0.00		0.00
Tasks Executed	0	0	0.00		0.00
Tasks Rescheduled	0		0.00		0.00
(+) Task Loader					
Kicked into major tick	0				
(+) Load on major tick	6	0	1.00		6.00
Load on minor tick	0	0	0.00		0.00
Major Tick	1				
Minor Tick	0				

Process Admin Console – Monitoring > Process Monitor

The **Monitoring > Process Monitor** section allows us to examine statistics on processes and services. It is composed of three major sections. The first is a Summary page where we can see the top users of the system in terms of resources.

Active Processes Currently Executing	0
Active Services Currently Executing	0

Most Expensive Services

Process App	Service Name	Total Time	Total Steps
Java Integration (tip)	MyTest	0:00:29.620	2
Java Integration (tip)	MyTest	0:00:28.494	2
Java Integration (tip)	MyTest	0:00:15.272	2
Java Integration (tip)	MyTest	0:00:11.504	2
Java Integration (tip)	MyTest	0:00:10.502	2
Java Integration (tip)	MyTest	0:00:09.651	2

Most Expensive Processes

Process App	Process Name	Total Time	Total Steps
No data available			

Most Expensive Service Steps

Process App	Service Name	Sub-Service Name	Step Name	Total Time	Total Instances
Java Integration (tip)	MyTest		Call MyTest	0:00:29.614	1
Java Integration (tip)	MyTest		Untitled1	0:00:28.487	1
Java Integration (tip)	MyTest		Untitled1	0:00:15.267	1
Java Integration (tip)	MyTest		Untitled1	0:00:11.491	1
Java Integration (tip)	MyTest		Untitled1	0:00:10.428	1
Java Integration (tip)	MyTest		Call MyTest	0:00:09.645	1
System Data (7.5.0.1)	Send Mail GS	Send E-mail via SMTP	Send Mail	0:00:00.217	1
Java Integration (tip)	MyTest		Untitled1	0:00:00.156	1
Mail Tests (tip)	Send Mail GS		Send Mail	0:00:00.101	2

Most Expensive Process Steps

Process App	Process Name	Sub-Process Name	Step Name	Total Time	Total Instances
No data available					

In this view, the Service Name entry is click-able which will open up a services view against that service.

Clicking on the services button at the top will show us a list of services executed.

Active Services Currently Executing

Process App	Service Name	Enter Time	Duration	Total Steps
There are no active services that are currently executing				

Active Services Not Currently Executing/Completed Services

Process App	Service Name	Last Enter Time	Last Duration	Total Duration	Total Steps
Horizon Tests (tip)	GS1	Oct 20, 2011 9:18:05 PM	0:00:00.028	0:00:00.028	8
Horizon Tests (tip)	GS1	Oct 20, 2011 9:17:21 PM	0:00:00.010	0:00:00.010	8
Horizon Tests (tip)	GS1	Oct 20, 2011 8:56:04 PM	0:00:00.016	0:00:00.016	8
Mail Tests (tip)	Send Mail GS	Oct 20, 2011 3:31:50 PM	0:00:00.035	0:00:00.035	5
Mail Tests (tip)	Send Mail GS	Oct 20, 2011 3:31:00 PM	0:00:00.129	0:00:00.129	5
Mail Tests (tip)	Send Mail GS	Oct 20, 2011 3:30:35 PM	0:00:00.352	0:00:00.352	5
Java Integration (tip)	MyTest	Oct 20, 2011 2:39:32 PM	0:00:09.651	0:00:09.651	2

Clicking on a service name in the Service Name column will show us details of the Service executed:

The screenshot shows a user interface for monitoring service execution. At the top, there are tabs for 'Summary', 'Processes', and 'Services'. The 'Services' tab is selected, indicated by a blue border. To the right of the tabs are 'Refresh' and 'Back' buttons. Below the tabs, the title 'Service "GS1" Details' is displayed. A table provides detailed information about the service, including its name (GS1), last enter time (Oct 20, 2011 9:18:05 PM), and total steps completed (8). Below this table, a section titled 'Completed Steps' lists the steps taken by the service, showing their names, last execution times, total durations, and total instances. The steps listed are End, Untitled2, S2, and S1.

Process App	Sub-Service Name	Step Name	Last Enter Time	Total Duration	Total Instances
Horizon Tests (tip)		End	Oct 20, 2011 9:18:05 PM	0:00:00.000	1
Horizon Tests (tip)		Untitled2	Oct 20, 2011 9:18:05 PM	0:00:00.000	3
Horizon Tests (tip)		S2	Oct 20, 2011 9:18:05 PM	0:00:00.003	1
Horizon Tests (tip)		S1	Oct 20, 2011 9:18:05 PM	0:00:00.003	3

From here we can see the name of the step, the date/time it was last executed and the total duration that the step took to execute. The Total Instances column shows how many times that step was executed in this instance of the service.

DB2 Database Performance

If DB2 is the database provider used to operate IBPM, there are various ways in which the performance of the DB can be monitored and tuned.

Monitoring DB2

As DB2 operates it can generate events when interesting things happen. These events can then be recorded and an examination of what is happening over time within the database can be found.

DB2 monitoring has the concept of "switches" for various aspects of event recording. If a switch is off, then that aspect of recording does not generate events. If switched on, then event recording occurs.

The switches available are:

Switch name	Description
BUFFERPOOL	
LOCK	
SORT	
STATEMENT	Information about SQL statements that have executed
TABLE	
TIMESTAMP	
UOW	

The DB2 command to control the switches has the following syntax:

```
update monitor switches using switchName {on|off} [switchName {on|off}]
```

For example:

```
update monitor switches using statement on
```

The setting of the switches affects the generation of events and now we need to look at the location to which this event information is written. This is controlled by the creation of event monitors.

```
create event monitor monitorName
  for event eventToMonitor
    write to { table | pipe | file } details
    [ { manualstart | autostart } ]
    [ additional event conditions ]
    [ addition table options ]
    [ additional file options ]
    [ additional workload manager options ]
```

If an event monitor is not started, it can be started with:

```
set event monitor monitorName state=1
```

and switched off with

```
set event monitor monitorName state=0
```

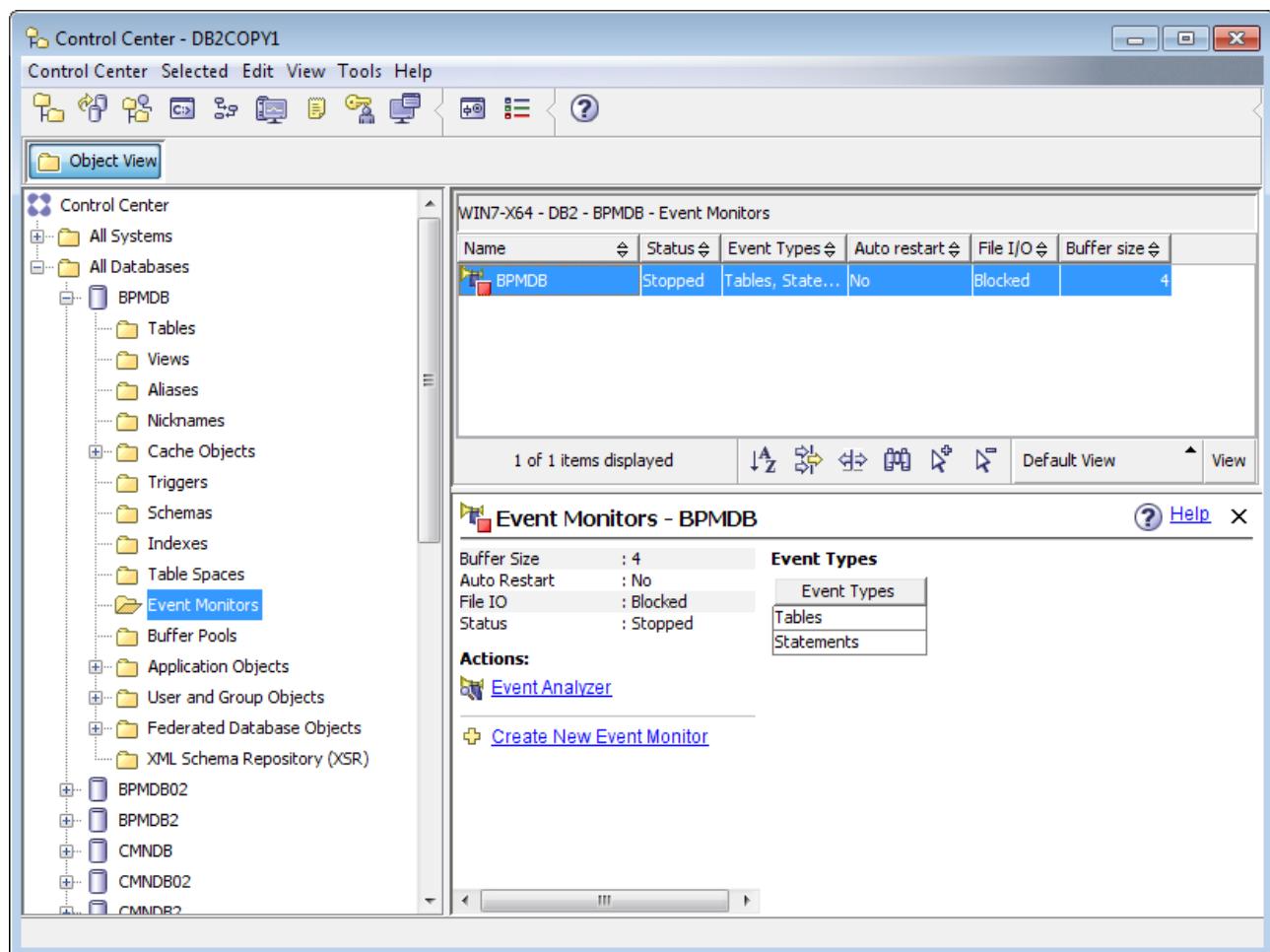
For example, to record data to a file for statements

```
create event monitor mymonitor for statements write to file 'C:\DB2\PerfData'
update monitor switches using statement on
set event monitor mymonitor state=1
```

The monitor data that is collected can then be examined by running the db2evmon command.

```
db2evmon -db PDWDB -evm mymonitor > file.txt
```

These commands are good for executing the event monitoring from the command line, we also have the ability to execute these commands from the Control Center through a nice GUI interface.



From here we can create new monitors and then view the results. We can also enable/disable the monitors here.

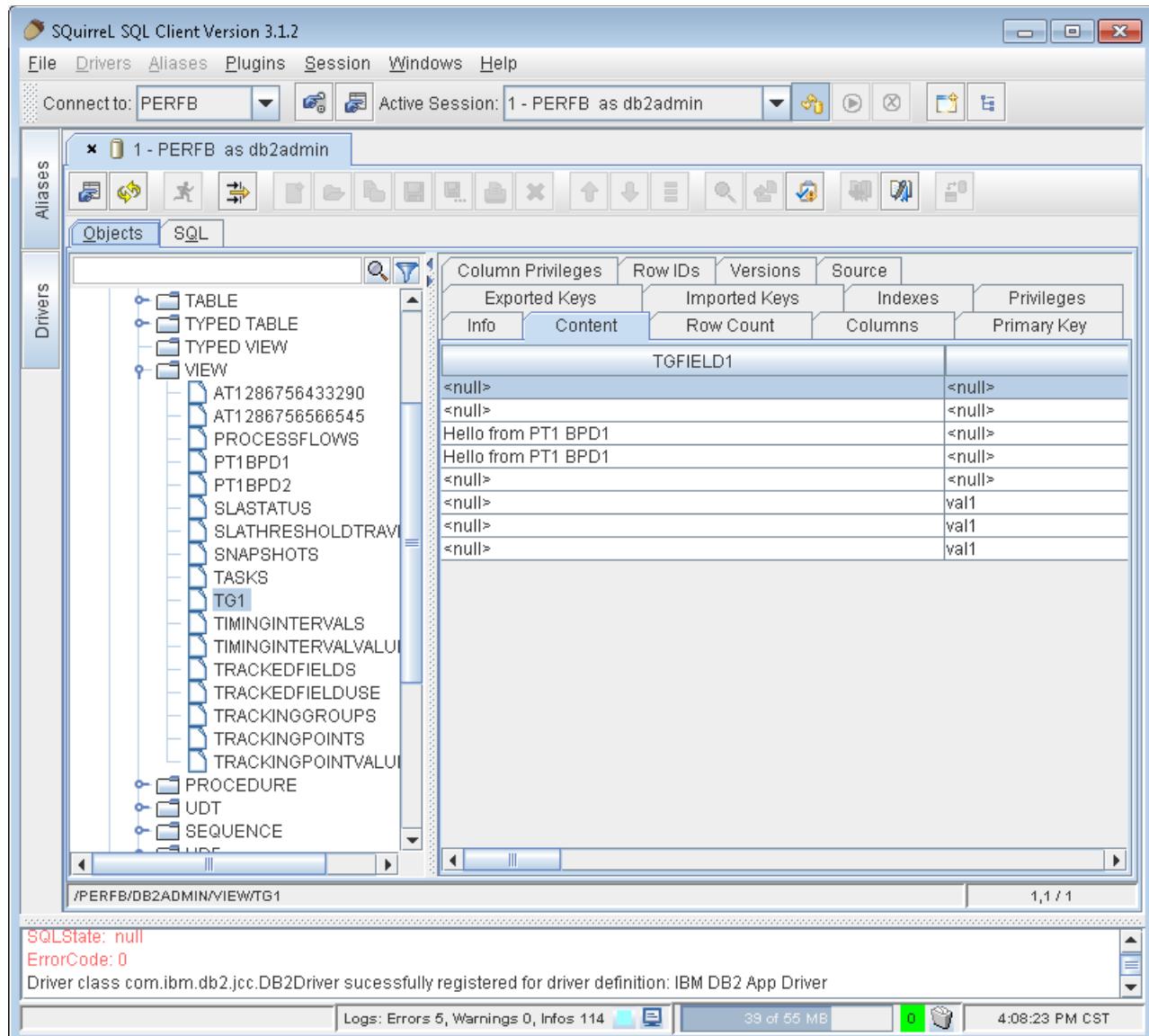
Useful scripts and tools

When working with IBPM, there are times when it is useful to have tools that can be used in administration. This section is a list of such tools and scripts.

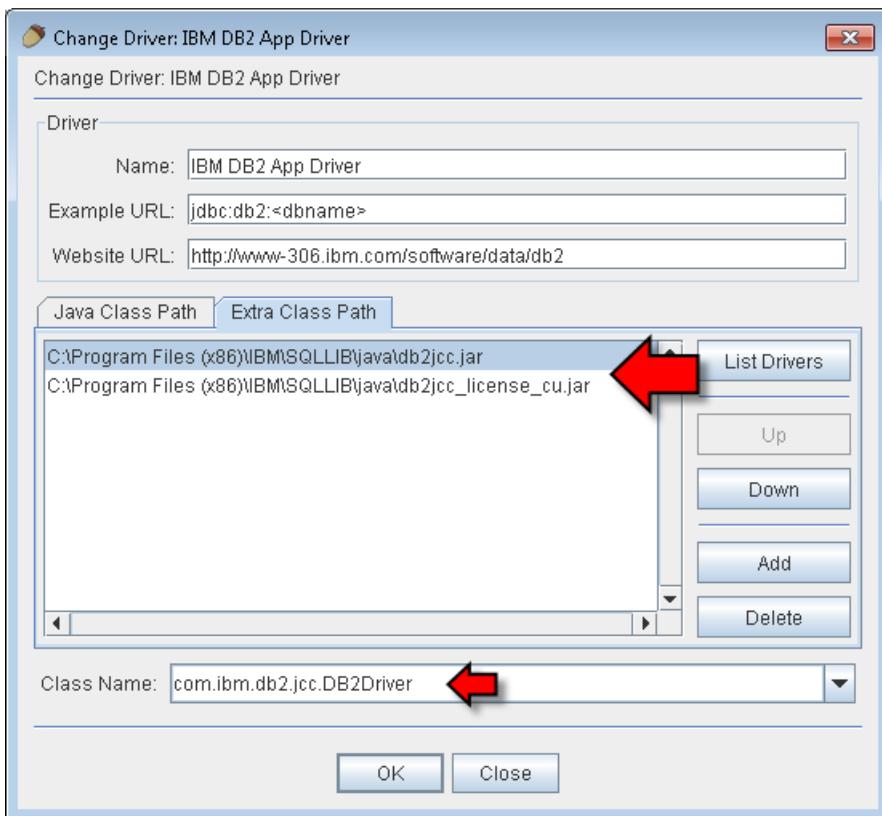
Viewing database table contents

IBPM comes with DB2 Express as the database but it appears that this is not a full DB2 installation. Specifically, the DB2 control center seems to be missing in Express. As such, there are times when one wants to see the content of tables but we are missing a tool to achieve this task. One solution to this problem is to use a free database utility downloaded from the Internet.

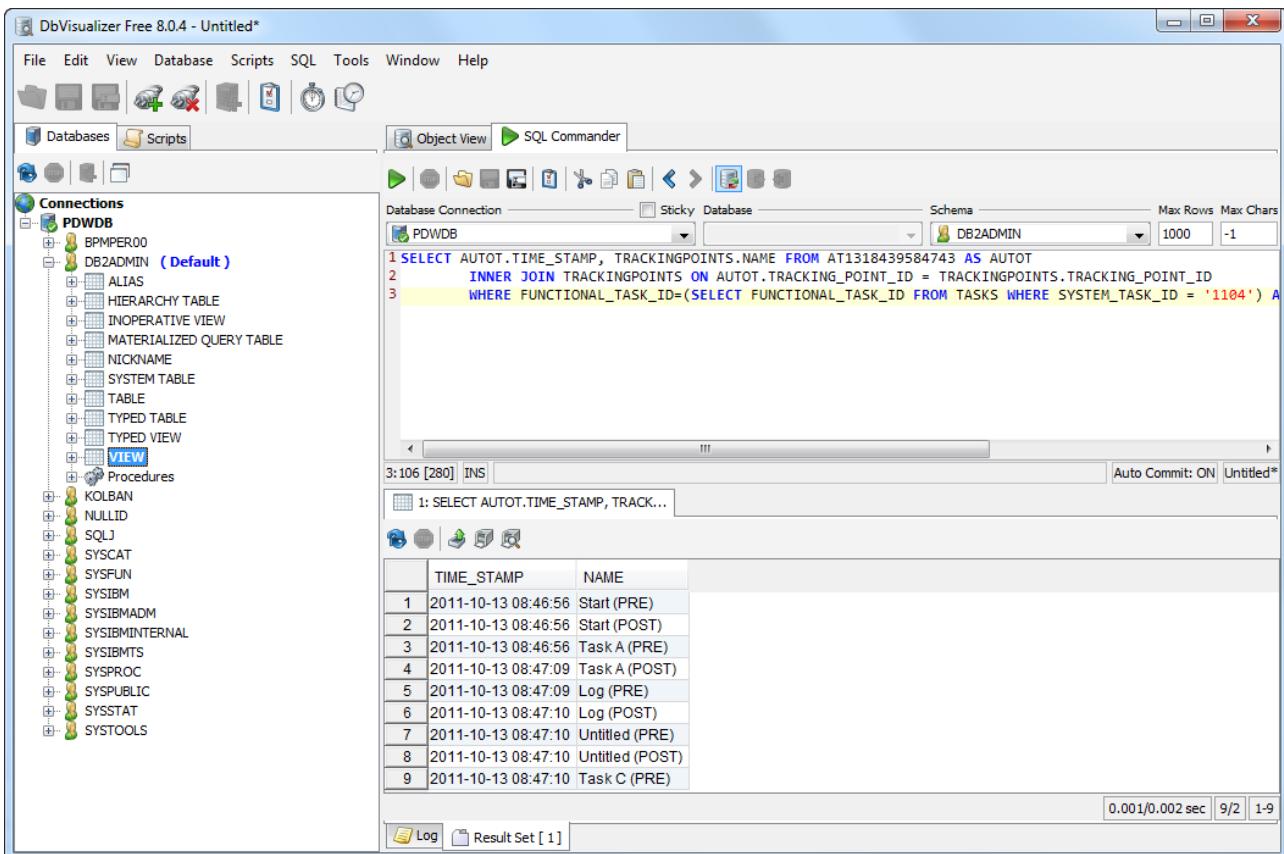
One such tool is called [SQuirreL SQL](#).



To set this up for DB2, in the Driver Configuration for this tool, add the DB2 Jars and change the name of the driver class.



Another tool available on the Internet is called [DB Visualizer](#). While SQuirreL SQL appears to be an open source project, DB Visualizer appears to be a for-fee product with a free personal use variation. Excluding the nature that it is unlikely that a IBPM user will be utilizing related software for personal use, the free version of DB Visualizer appears to be more limited in function than SQuirreL SQL. Conversely, the for-fee version appears more polished and more robust than the open-source product.



Recovery options

In the event of system failures, the environment will need to recover. There are a number of papers written on this subject which are referenced in the "See Also" section. From a high level, it appears that we do not wish to attempt to create an "Active/Active" configuration across data centers.

See also:

- DeveloperWorks - [Comment lines: Tom Alcott: Everything you always wanted to know about WebSphere Application Server but were afraid to ask -- Part 3](#) - 2012-11-21
- [Disaster Recovery in an IBM BPM v7.5 Production Environment](#) – 2011-05-31
- DeveloperWorks - [The WebSphere Contrarian: High availability \(again\) versus continuous availability](#) – 2010-04-14
- DeveloperWorks - [The WebSphere Contrarian: Run time management high availability options, redux](#) – 2010-01-27
- DeveloperWorks - [Asynchronous replication of WebSphere Process Server and WebSphere Enterprise Service Bus for disaster recovery environments](#) - 2008-11-24
- DeveloperWorks - [Comment lines: Tom Alcott: Everything you always wanted to know about WebSphere Application Server but were afraid to ask, Part 5](#) - 2007-07-18
- DeveloperWorks - [Comment lines: Tom Alcott: Everything you always wanted to know about WebSphere Application Server but were afraid to ask -- Part 2](#) - 2005-12-07
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Maintain continuous availability while updating WebSphere Application Server enterprise applications](#) – 2004-12-01
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Maintain continuous availability while updating WebSphere Application Server enterprise applications](#) - 2004-12-01

JavaScript in IBPM

The JavaScript programming language is used extensively in IBPM solutions. JavaScript can be seen to be used in service definitions and also in expression evaluations. Understand that the JavaScript being executed here is executed at the server side and **not** within a browser. A common mistake I have seen is working in JavaScript on the server side and then working in JavaScript in the browser and mistakenly thinking that in both cases, code runs in the same environment.

In addition, JavaScript can be defined as the direct implementation type for a BPD activity. This might be useful for testing or quick hacks but is unlikely to be a good long term practice. There is no reuse available for code implemented in this fashion.

It is believed that a knowledge of JavaScript will be essential to long term construction of IBPM solutions. Although simple assignments can be performed without detailed JavaScript skills, more subtle activities such as defining variables, loops and other language constructs will require skills. Fortunately, JavaScript is not a complex language to learn and can be picked up quickly especially if you already know the Java programming language.

See:

- w3schools.com - [JavaScript and HTML DOM Reference](#)
- JavaScriptKit.com - [JavaScript Reference](#)
- DojoToolkit – [Documentation](#)

Editing JavaScript

The editing of the JavaScript code should be done within IBPM PD. Each of the JavaScript editor areas provided in PD has content assist and syntax checking. Unfortunately, the content assist doesn't address all the possible content and the syntax checking assistance sometimes indicates perfectly valid and correct JavaScript as containing warnings or errors (on occasion) so although both these functions are useful, don't rely on them 100% of the time. The fact that the assistance features are present is probably better than them not being present but don't panic if a variable you think should be in scope doesn't show up in the content assist or a statement is flagged as in error when no amount of review shows a problem ... simply ignore and move on as though the assistance tools were not present.

When the JavaScript entry assistance does let us down, we can get into quite a mess. JavaScript is a loosely typed language. As such, declaration of variables with types can't be easily checked at editing time (by PD). Making a typo in JavaScript code is very common as is calling a function by the wrong name. Debugging and correcting such errors can be frustrating. For example, to find a Process Instance by its ID, we call the following function:

```
tw.system.findProcessInstanceByID()
```

There are many opportunities here to make an error that won't be caught. I had mistakenly coded:

```
tw.system.findProcessInstanceById()
```

and it took me far too long to find that mistake.

IBPM JavaScript name spaces

IBPM exposes a number of JavaScript name spaces which contain various data items.

`tw.local.*` Variables within the current process/service.

The JavaScript language doesn't have the notion of namespaces so this concept is provided by properties of objects. For example, the notion of "tw" is actually a JavaScript object which has a property called "local". This property is itself an object.

Reusing JavaScript

Commonly used JavaScript functions can be added to a Java Script source file (a file ending in ".js"). This file can then be added as a managed file in a Process Application. The Managed File has to be of type "Server". Once added, any JavaScript found within becomes part of the environment when running other JavaScript routines. This becomes especially useful if the JavaScript source file contains function declarations. These can then be used in other Java Script code fragments within the solution without any further work being needed. Although not investigated yet, it is believed that the re-use of the JavaScript code by adding it as server managed files needs to be carefully managed to prevent name collisions since the scope of Server Managed files appears quite large. It is currently seen that the Java Script editor flags the use of such functions as a warning as they are said to be unknown functions. However, they can be run just fine.

BPD Data Types

The IBPM environment utilizes and provides some pre-existing data types. Understanding these data types is useful in advanced situations. Many of these data types have a method on them called `gettypeName()` which when called will return a String representation of the data type being examined.

ConditionalActivity	
Map	
NameValuePair	
TWAdhocStartingPoint	
TWChart	
TWProcessInstance	
TWDocument	
TWManagedFile	
Date	
TWUser	
Record	
TWChart	
TWDate	
XMLElement	
TWOObject	
TWReport	
TWSearchColumn	
TWSearchColumnMetaData	
TWSearchCondition	
TWSearchResults	

TWSearchResultsRow	
TWTTask	
TWStep	
SQLResult	

Data Type – ConditionalActivity

A `ConditionalActivity` object represents an Activity in the BPD that is defined as conditional.

● <code>id</code> - String
● <code>name</code> - String
● <code>isConditional</code> - Boolean
● <code>children</code> - <code>TWObject*ConditionalActivity</code>

- `id` – An internal GUID tied to the Activity on the BPD
- `name` – The name of the Activity in the BPD
- `isConditional` – Whether or not the activity is flagged as conditional

An array of these objects for the currently executing process can be found in the variable:

```
tw.system.currentProcess.conditionalActivities
```

One can use the `toXMLString()` method on variables of this type to dump an XML representation.

See also:

- Conditional Activities

Data Type – Map

The Map data type holds an unordered collection of key/value pairs. You can think of this very much like the Java `HashMap` class. A value is added to the map using the `put()` method with the key as the first parameter and the value as the second. A value can subsequently be retrieved from the map using the `get()` method.

The array of all keys can be retrieved with the `keyArray()` method. Similarly, an array of all values can be obtained with `valueArray()` method. For both these methods, the order of returned entries is undefined as there is no ordering in maps.

The number of items contained within the map can be obtained with the `size()` method.

An entry can be removed from the map using the `remove()` method.

If we simply want to know if a map contains a value for a specific key, we can use the `containsKey()` method.

	resultArray - TWObject
	size() - Integer
	keyArray() - *ANY
	valueArray() - *ANY
	get(ANY key)
	put(ANY key, ANY value)
	remove(ANY key) - VOID
	containsKey(ANY key) - Boolean

An instance of this object can be created through:

```
tw.local.myMap = new tw.object.Map();
```

If passed to a Java class, the data type passed is `java.util.HashMap`.

Data Type – NameValuePair

A Business Object data structure that is pre-defined by IBPM that contains two fields:

- name – String
- value – String

Data Type – ProcessSummary

(As of 8.5)

This structure contains:

- processAppId
- processAppName
- processId
- name
- description
- countActive
- countOverdue
- countAtRisk
- countOnTrack

This data type can be returned from `tw.system.retrieveProcessSummaries()`,
`tw.system.retrieve.ProcessSummary()`, `TWProcessPerformanceMetric.retrieveProcessSummary()`

See also:

- Data Type – TWProcessPerformanceMetric
- Process Summary Control

Data Type – Record

The Record data type appears to be a dynamic set of properties associated with the an instance of the object. It looks like one can assign to and retrieve values from this object as though they were properties. Quite why it shows that it has the properties of a list object is unknown and may be an error in the PD tool.

```

● propertyNames - *String
● listLength - Integer
● listSelectedIndex - Integer
● listAllSelectedIndices - *Integer
● listSelected - *#
● listAllSelected - Array
● toXMLString() - String
● toXML() - XMLElement
● describe() - XMLElement
● remove(String propertyName) - VOID
● removeIndex(Integer listIndex) - VOID
● insertIntoList(Integer position, ANY object) - VOID
● listAddSelected(Integer index) - VOID
● listRemoveSelected(Integer index) - VOID
● listClearAllSelected() - VOID
● listIsSelected(Integer index) - Boolean
● listToNativeArray() - Array

```

For example, if we define a variable called "myRecord" of type Record then in one activity I can assign a value to a property within it such as:

```
tw.local.myRecord = new tw.object.Record();
tw.local.myRecord.myProp = "Hello";
```

and then in a different activity we can code:

```
log.info(tw.local.myRecord.myProp);
```

The Record object appears to behave as a cross between a standard JavaScript object and a BPM Business Object. We can assign values to its properties and they are persisted across activities however they do not need to be predefined.

The `propertyNames` attribute returns a string which is a comma separated list of the explicitly added properties to the record. This allows us to enumerate the properties contained within it.

See also:

- Integration with supplied DB connectors

Data Type – SQLResult

This data type is defined in the System Data toolkit. It appears to have been designed to hold results from executing DB queries through the SQL query services supplied in the System Data Toolkit.

```

● type - String
● columns - TWObject*SQLResultSetColumn
● columnIndexes - Map
● rows - TWObject*IndexedMap
● updateCount - Integer
● outValues - TWObject*ANY

```

- `type` – The type of data. Usually "RESULT_SET"
- `columns` – Unknown
- `columnIndexes` – Unknown
- `rows` – List of IndexedMap
- `updateCount` – Unknown

- outValues – Unknown

Imagine a SQL Query executed using the System Data Toolkit function called `SQLExecuteStatement` that returns a `SQLResult []` array. It is strange that this returns an array as `SQLResult [0]` seems to hold everything we need. Specifically `SQLResult [0]` holds an array object called `rows`. Each element of the `rows` array has a property named after the retrieved data column.

For example, if the data returned from a query in a `SQLResult` were:

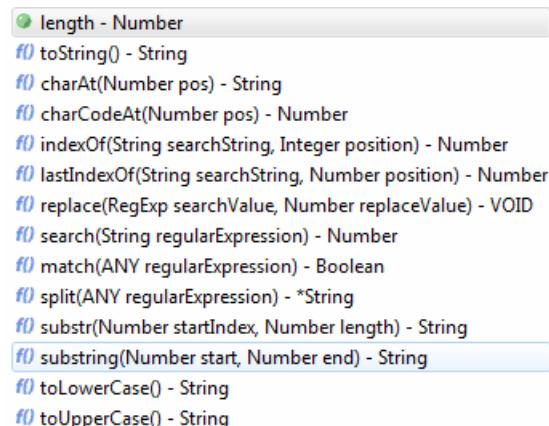
A	B
"A1"	"B1"
"A2"	"B2"
"A3"	"B3"

Then `myVar [0].rows [1].B` would be "B2".

To see how many rows are returned, use `SQLResult [0].rows.listLength`.

Data Type – String

The String data type represents a piece of text within the process.

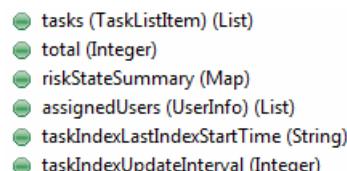


The `search()` method will return 0 if found and -1 if not found.

The `replace()` method seems to work with formats such as /<pattern>/g.

Data Type – TaskListData

This data type appears to be used as the return type from `tw.system.retrieveTaskList()`. It contains details of the tasks that are eligible to be performed by a user. It is primarily used to populate the Task List Coach View.



- `tasks` – a list of `TaskListItem` objects containing the specific details of each task
- `total` – The total number of tasks available. This looks like it will be useful to allow us to

page through the tasks. Since the `retrieveTaskList()` method has both a start point and length pair of parameters, we can determine where in the list to start retrieval and how many to retrieve.

- `riskStateSummary` – This is a JavaScript Map which is keyed by "risk state" and has values of how many tasks there are in that particular state.
- `assignedUsers` – A list of `UserInfo` objects, one for each user that is associated with a task in the returned list.
- `taskIndexLastIndexStartTime` – The time when the list of tasks was internally refreshed
- `taskIndexUpdateInterval` – How often the list of tasks is internally updated

See also:

- Data Type – `TaskListItem`
- Method: `retrieveTaskList()`

Data Type – `TaskListItem`

This data type describes the details of a single task. A list of such items can be found in the `TaskListData` object.

- `id (String)`
- `subject (String)`
- `priority (Integer)`
- `isAtRisk (Boolean)`
- `riskState (String)`
- `dueDate (String)`
- `closedDate (String)`
- `assignedToUserId (String)`
- `assignedToUserFullName (String)`
- `assignedToTeamId (String)`
- `assignedToTeamName (String)`
- `processInstanceId (String)`
- `processInstanceName (String)`

- `id` – The unique identifier of this task
- `subject` – The subject property of the task
- `priority` – The priority of the task (integer value)
- `isAtRisk` – A boolean which flags this task as at risk (or not)
- `riskState` – One of
 - Overdue
 - ...
- `dueDate` – When the task is due
- `closedDate` – When the task was closed
- `assignedToUserId` – The userid that owns the task. Off the format "2048.xxxx".
- `assignedToUserFullName` – The text full name of the user that owns the task.
- `assignedToTeamId` – The id of the team to which this task is assigned.

- assignedToTeamName – The human readable name of the team to which this task is assigned.
- processInstanceId – The unique identifier of the process which created this task
- processInstanceName – The instance description of the process instance

See also:

- Data Type – TaskListData

Data Type – TaskListInteractionFilter

A string defined as one of the following:

Value	Display Text
ASSESS_AND_WORK_ON	Assess and work on
ASSESS_AVAILABLE	Assess available
WORK_ON	Work on
CHECK_COMPLETED	Check completed

- ASSESS_AND_WORK_ON – Either tasks which are not claimed or you have claimed but not yet completed
- ASSESS_AVAILABLE – Tasks which are not claimed
- WORK_ON – Tasks which you have claimed but not yet completed
- CHECK_COMPLETED – Tasks which have been completed

See also:

- Data Type – TaskListProperties
- Method: retrieveTaskList()

Data Type – TaskListProperties

This data type is passed as a parameter to tw.system.retrieveTaskList(). It controls which tasks are returned.

- [teamId \(String\)](#)
- [userId \(String\)](#)
- [interactionFilter \(TaskListInteractionFilter\)](#)
- [includeAssignedUsers \(Boolean\)](#)
- [includeRiskStateSummary \(Boolean\)](#)
- [dueSlice \(DateRangeString\)](#)
- [searchFilter \(String\)](#)
- [collapsedRiskStates \(TaskListRiskState\) \(List\)](#)

- teamId
- userId
- interactionFilter – One of
 - ASSESS_AND_WORK_ON
 - ASSESS_AVAILABLE
 - WORK_ON

- CHECK_COMPLETED – Tasks which have been completed
- includeAssignedUsers
- includeRiskStateSummary
- dueSlice
- searchFilter
- collapsedRiskStates

See also:

- Data Type – TaskListInteractionFilter
- Method: retrieveTaskList()

Data Type – Team

This data type defines a dynamic Team.

- name (String)
- members (String) (List)
- managerTeam (String)

The name property is the name of the Team. Apparently this is ignored. The members property is a list of members of the team. The managerTeam property is the name of another Team that holds the managers of this Team.

See also:

- Teams
- Data Type – TWTeam

Data Type – TeamDashboardSupport

(as of 8.5)

f1 retrieveTeamSummary(String searchFilter, String timeZoneAsString, Boolean checkAuthorization) - TeamSummary
f1 retrieveTeamTaskTrend(String units, Integer numPeriods, String endPeriod, String timezone, String searchFilter, Boolean checkAuthorization) - ChartData
f1 retrieveTeamMemberList(String timezone) - TWObject*TeamRosterEntry

See also:

- Data Type – TWTeam

Data Type – TWAdhocStartingPoint

This JavaScript data structure represents an "ad-hoc" starting point within an already running IBPM process instance.

Properties:

Name/Type	Description
id:String	The id value of this ad-hoc starting point
name:String	The name value of this ad-hoc starting point
processInstance:TWProcessInstance	The process instance associated with this ad-hoc starting point

Methods:

Name	Description
startNew	Starts an ad-hoc activity

A field in the `TWProcessInstance` data type called `adhocStartingPoints` contains a list of `TWAdhocStartingPoints` associated with that instance of the process. The `TWProcessInstance` also has convenience methods called `findAdhocStartingPointByID()` and `findAdhocStartingPointByName()` to walk this list looking for specific entries.

Data Type – TWDate

The `TWDate` object represents a date/time. An instance of this object can be created with:

```
new tw.object.Date()
```

The format fields in some of the methods allow for custom date/time parsing or construction. An example of such a function is `format()`. A date can also be constructed using the `parse()` function. The format is a String that is encoded as follows:

Code	Description	Example
G	Era designator	AD ("GG")
y	Year	2011 ("yyyy"); 11 ("yy")
M	Month in year	July; Jul; 07
w	Week in year	26
W	Week in month	3
D	Day in year	203
d	Day in month	11
F	Day of week in month	3
E	Day in week	Tuesday ("EEEE"); Tue ("EEE")
a	AM/PM indicator	PM ("GG")
H	Hour in the day (0-23)	0
k	Hour in the day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	45
s	Second in minute	33
S	Millisecond	934
z	Time zone	Central Standard Time; CST; GMT-06:00
Z	Time zone	-800

Examples:

Pattern	Example
MM/dd/yyyy	11/08/2012
yyyy-MM-dd HH:mm:ss	
yyyy-MM-dd HH:mm:ss.SSS zzz	

(As of 8.5)

```
fl) getDate() - int
fl) getDay() - int
fl) getFullYear() - int
fl) getMonth() - int
fl) getHours() - int
fl) getMinutes() - int
fl) getSeconds() - int
fl) getMilliseconds() - int
fl) setDate(int dayOfMonth) - void
fl) setDay(int dayOfWeek) - void
fl) setFullYear(int year) - void
fl) setMonth(int month) - void
fl) setHours(int hours) - void
fl) setMinutes(int minutes) - void
fl) setSeconds(int seconds) - void
fl) setMilliseconds(int millis) - void
fl) getUTCDate() - int
fl) getUTCDay() - int
fl) getUTCFullYear() - int
fl) getUTCMonth() - int
fl) getUTCHours() - int
fl) getUTCMinutes() - int
fl) getUTCSeconds() - int
fl) getUTCMilliseconds() - int
fl) setUTCDate(int dayOfMonth) - void
fl) setUTCDay(int dayOfWeek) - void
fl) setUTCFullYear(int year) - void
fl) setUTCMonth(int month) - void
fl) setUTCHours(int hours) - void
fl) setUTCMinutes(int minutes) - void
fl) setUTCSeconds(int seconds) - void
fl) setUTCMilliseconds(int millis) - void
fl) toNativeDate() - Date
fl) getTimezoneOffset() - void
fl) format() - String
fl) format(String formatString, String timeZoneString) - String
fl) formatDate(String dateStyle) - String
fl) formatTime(String timeStyle) - String
fl) formatDateTime(String dateStyle, String timeStyle) - String
fl) parse(String dateAsString, String formatString, String timeZoneString, String localeString) - void
fl) getTime() - int
fl) setTime(int time) - void
fl) getDayOfWeek() - int
fl) getUTCTime() - int
fl) setUTCTime(int time) - void
```

Notice the `getTime()` and `setTime()` methods. These return and set times as seconds from an epoch.

Unfortunately this object has no explicit date/time arithmetic available to it so adding or subtracting time can be a challenge. However, if we look to Java, we find a powerful class called `java.util.Calendar`. The `Calendar` class has all kinds of mechanisms to perform arithmetic and other functions. The task then is to create an instance of a Java `Calendar` from the `TWDate` object, perform the arithmetic on the `Calendar` and then create a new `TWDate` from the updated `Calendar` object. The following will subtract one day from the original date:

```

tw.local.dueDate1.setFullYear(2010);
tw.local.dueDate1.setMonth(2);
tw.local.dueDate1.setDate(1);
tw.local.dueDate1.setHours(15);
tw.local.dueDate1.setMinutes(18);
tw.local.dueDate1.setSeconds(00);
tw.local.dueDate1.setMilliseconds(0);

log.info("Original Date is: " + tw.local.dueDate1.format("yyyy-MM-dd HH:mm:ss.SSS zzz"));

var call1 = new java.util.GregorianCalendar(
    tw.local.dueDate1.getFullYear(),
    tw.local.dueDate1.getMonth(),
    tw.local.dueDate1.getDate(),
    tw.local.dueDate1.getHours(),
    tw.local.dueDate1.getMinutes(),
    tw.local.dueDate1.getSeconds());

call1.add(java.util.Calendar.DATE, -1);

tw.local.secondDate = new tw.object.Date();
tw.local.secondDate.setMilliseconds(0);
tw.local.secondDate.setFullYear(call1.get(java.util.Calendar.YEAR));
tw.local.secondDate.setMonth(call1.get(java.util.Calendar.MONTH));
tw.local.secondDate.setDate(call1.get(java.util.Calendar.DAY_OF_MONTH));
tw.local.secondDate.setHours(call1.get(java.util.Calendar.HOUR));
tw.local.secondDate.setMinutes(call1.get(java.util.Calendar.MINUTE));
tw.local.secondDate.setSeconds(call1.get(java.util.Calendar.SECOND));

log.info("New Date is: " + tw.local.secondDate.format("yyyy-MM-dd HH:mm:ss.SSS zzz"));

```

There is a general function in the Kolban JavaScript toolkit that can be used to add/subtract dates (see: Kolban JavaScript Toolkit)

Data Type – TWDocument

The TWDocument object represents a document uploaded through a Coach and stored/managed by IBPM.

Properties:

Property	Description
id:String	The internal <i>id</i> of the document. Will be unique for every document.
name:String	The <i>name</i> of the document.
version:Integer	The version number of the document.
type:String	URL or File
uri:String	The URI of the document if its type is URL.
contentType:String	The HTTP MIME type of the document.
allVersions:TWDocument[]	An array of all the versions of this document.
processInstance:TWProcessInstance	The Process Instance that this document belongs to.
hideInPortal:Boolean	True if the document should be <i>hidden</i> in Process Portal.
modifiedBy:TWUser	The user who last modified/added the document.
modificationDate:Date	The date when the document was added.

Methods:

deleteAllVersions	
gettypeName	

`writeDataToFile`

Write the content of the document to a file on the local file system where Process Server is executing.

The `TWProcessInstance` object has an `addDocument()` method to create a new document.

There does not appear to be a way to get "simple" access to the content of the document however it appears that we can write it to a file and then load the file content. As to why there is no way to access the content, there are a couple of thoughts (unverified) as to why IBM's designers chose not to provide this. One thought is that working with binary data in the context of a Business Process is simply not a desirable trait. By not providing any getters for the data, the designer of a solution is forced to immediately look into Java alternatives from the start. A second possible reason is that the JavaScript language has no binary data handlers.

A possible solution to this is to encode the data into a String representation. We appear to have two choices for this. The first is `hexBinary` which takes a single byte of data and represents it by two characters. The two characters are the hexadecimal notation for the byte value. A class is supplied as part of the Axis2 package to perform `hexBinary` encoding. The class is called "[HexBinary](#)". The second technique is to encode as `base64`. `Base64` encoding is a little more complex but basically encodes 3 bytes of data into 4 characters. `Base64` encoding can be used from the Apache [commons.codec](#) package.

See also:

- Working with document attachments in JavaScript
- Error: Reference source not found
- Error: Reference source not found
- Error: Reference source not found
- Data Type – `TWProcessInstance`

Data Type – `TWHolidaySchedule`

This data type describes a holiday schedule. A holiday schedule is a list of dates that are considered holidays. This comes into play when we are calculating a due date for a process or activity. If the date on which a task would otherwise complete is a holiday, then we need to accommodate for that by moving the due date to the right to the first non-holiday date.

 `id` - String
 `name` - String
 `dates` - `TWObject*``TWDate`
 `propertyNames` - *String
 `toString()` - String
 `toXMLString()` - String
 `toXML()` - XMLElement
 `describe()` - XMLElement
 `remove(String propertyName)` - void
 `getTypeName()` - String
 `save()` - void
 `load(String key)` - void
 `metadata(String key)` - Object
 `isDirty()` - boolean

The variable called `tw.system.holidaySchedules` contains a list of all the different holiday schedules available.

We can add/define a new holiday schedule with

```
tw.system.addHolidaySchedule(TWHolidaySchedule holidaySchedule).
```

We can find a holiday schedule by name with

```
tw.system.findHolidayScheduleByName(String name).
```

We can remove a holiday schedule with

```
tw.system.removeHolidaySchedule(String id).
```

We can update a holiday schedule with

```
tw.system.updateHolidaySchedule(TWHolidaySchedule holidaySchedule).
```

See also:

- Data Type – TWTimeSchedule

Data Type – TWLink

This item defines the referenced links in the documentation.

- [name - String](#)
- [url - String](#)
- [relationshipType - String](#)
- [assetType - String](#)
- [relationshipTypeNamespace - String](#)
- [assetTypeNamespace - String](#)
- [linkType - String](#)

Instances of this type can be found from instances of TWProcessInstance and TWStep.

See also:

- Data Type – TWProcessInstance
- Data Type – TWStep

Data Type – TWManagedFile

The managed file type represents a file that can be found as part of a toolkit or Process Application.

- [id - String](#)
- [name - String](#)
- [path - String](#)
- [url - String](#)
- [*f0* readText\(\) - String](#)
- [*f0* writeDataToFile\(String newFileLocation\) - String](#)
- [*f0* getTypeName\(\) - String](#)

Data Type – TWModelNamespace

An instance of this object can be retrieved at the BPD level using:

```
tw.system.model
```

	env
	processApp - TWProcessApp
	processAppSnapshot - TWProcessAppSnapshot
	searchMetaData - TWSearchMetaData
	findProcessAppByID(String processAppID) - TWProcessApp
	findProcessAppByName(String processAppName) - TWProcessApp
	findProcessAppByAcronym(String processAppAcronym) - TWProcessApp
	getAllProcessApps(boolean includeToolkits) - *TWProcessApp
	getAllToolkits() - *TWProcessApp
	findManagedFileByPath(String managedFilePath, String managedFileType) - TWManagedFile
	findProcessByName(String processName) - TWProcess
	findServiceByName(String serviceName) - TWService
	findParticipantGroupByName(String participantGroupName) - TWParticipantGroup
	findParticipantGroupByID(String participantGroupId) - TWParticipantGroup

See also:

- Data Type – [TWProcessApp](#)
- Data Type – [TWProcess](#)
- Data Type – [TWService](#)
- Data Type – [TWParticipantGroup](#)
- Data Type – [TWManagedFile](#)

Data Type – **TWOObject**

TWOObject appears to be the implementation of lists, arrays and Business Object types within IBPM. It supports direct assignment through square bracket notation including the idea of sparse arrays (i.e. assigning to an array element past the end of the array).

	propertyNames - *String
	listLength - Integer
	listSelectedIndex - Integer
	listAllSelectedIndices - *Integer
	listSelected - #
	listAllSelected - *#
	toXMLString() - String
	toXML() - XMLElement
	describe() - XMLElement
	remove(String propertyName) - VOID
	removeIndex(Integer listIndex) - VOID
	insertIntoList(Integer position, ANY object) - VOID
	listAddSelected(Integer index) - VOID
	listRemoveSelected(Integer index) - VOID
	listClearAllSelected() - VOID
	listIsSelected(Integer index) - Boolean
	listToNativeArray() - Array
	getTypeName() - String

If we want to add a new entry at the very end of the list, use the following pattern:

```
tw.local myListVariable[tw.local myListVariable.listLength] = value;
```

One of the more interesting default properties of this object is `propertyNames` which is an array of names of the properties in the Business Object which have values associated them. This can be used for introspection of an arbitrary Business Object to determine its internal structure. Take careful note that the `propertyNames` is **only** fields that have values. For example, if a BO is defined as having fields "a", "b" and "c" but only "b" is populated, then `propertyNames` will be the list ["b"] and will not show the potential fields of "a" or "c". If a variable is an array, all we get is the first element.

Data Type – TWParticipantGroup

It is believed that this data type has now been deprecated in favor of TWTeam.

```
id - Integer
name - String
processApp - TWProcessApp
snapshot - TWProcessAppSnapshot
associatedRole - TWRole
users - *TWUser
roles - *TWRole
allUsers - *TWUser
hasUser(Scriptable user) - boolean
addUsers(Scriptable users) - VOID
removeUsers(Scriptable users) - VOID
addRoles(Scriptable roles) - VOID
removeRoles(Scriptable roles) - VOID
refresh() - VOID
getTypeName() - String
```

The property called `users` lists the explicit users added to the group

The property called `roles` lists the explicit groups added to the group

The property called `allUsers` lists all the users in the group that were added explicitly and also those users resolved from expanding the groups in this group.

Here is an example script to retrieve all users for a participant group:

```
var myPG1 = tw.system.org.findParticipantGroupByName ("myPG1");
var allUsers = myPG1.allUsers;
for (var i=0; i<allUsers.length; i++) {
    log.info(allUsers[i].name);
}
```

The `addUsers` function doesn't appear to do anything on Process Center. It is believed that `addUsers` will work when deployed to a Process Server.

See also:

- Data Type – TWProcessApp
- Data Type – TWRole
- Data Type – TWUser
- Data Type – TWTeam
- Teams

Data Type – TWProcess

(The following as of 8.5)

```

● id - String
● name - String
● processApp - TWProcessApp
● performanceMetrics - TWProcessPerformanceMetric
● description - String
● searchMetaData - TWSearchMetaData
● fullTextSearchMetaData - TWSearchMetaData
● conditionalActivities - TWOObject
● links - *TWLink
● phases - TWOObject*Phase
● steps - TWOObject*Step
● guid - String
fl) getTypeName() - String
fl) startNew(Map inputParams) - *String

```

See also:

- Starting a new process
- Data Type – TWProcessApp
- Data Type – TWProcessInstance
- Data Type – TWProcessPerformanceMetric
- Data Type – TWProcessAppSnapshot

Data Type – TWProcessApp

(The following as of 8.5)

```

● id - String
● name - String
● acronym - String
● isToolkit - boolean
● snapshots - *TWProcessAppSnapshot
● currentSnapshot - TWProcessAppSnapshot
● defaultSnapshot - TWProcessAppSnapshot
● activeInstances - *TWProcessInstance
● links - *TWLink
fl) findSnapshotByID(String snapshotID) - TWProcessAppSnapshot
fl) findSnapshotByName(String snapshotName) - TWProcessAppSnapshot
fl) getTypeName() - String
fl) getLinks(boolean includeReferencedToolkits, function linkFilter) - *TWLink

```

See also:

- Data Type – TWProcessInstance
- Data Type – TWParticipantGroup
- Data Type – TWProcess
- tw.system.model.processApp
- tw.system.model.getAllProcessApps()

Data Type – TWProcessAppSnapshot

The following (as of 8.5):

```

● id - String
● name - String
● defaultSettings - TWProcessAppDefaults
● processApp - TWProcessApp
● dateCreated - TWDate
● dateInstalled - TWDate
● isActive - boolean
● activeInstances - *TWProcessInstance
● links - *TWLink
● findManagedFileByPath(String managedFilePath, String managedFileType) - TWManagedFile
● findProcessByName(String processName) - TWProcess
● findServiceByName(String snapshotName) - TWService
● findParticipantGroupByName(String participantGroupName) - TWParticipantGroup
● activate() - void
● deactivate() - void
● getTypeName() - String

```

- Data Type – TWProcess
- Data Type – TWProcessApp
- Data Type – TWProcessInstance
- Data Type – TWManagedFile
- Data Type – TWService
- Data Type – TWParticipantGroup

Data Type – TWProcessInstance

This data type reflects an instance of a process.

```

● id - String
● name - String
● process - TWProcess
● processApp - TWProcessApp
● snapshot - TWProcessAppSnapshot
● dueDate - TWDate
● status - String
● businessData - Map
● tasks - *TWTTask
● documents - *TWDocument
● adhocStartingPoints - *TWAdhocStartingPoint
● sharepointSiteURL - String
● selectedConditionalActivities - TWOObject
● links - *TWLink
● getTypeName() - String
● abort() - void
● suspend() - void
● resume() - void
● addComment(String comment) - void
● sendHelpRequest(String sendTo, String description) - void
● addDocument(String type, String name, String fileLocation, boolean hideInPortal, TWUser createdBy, Map properties) - TWDocument
● findDocuments(Map properties, String searchType) - *TWDocument
● findAdhocStartingPointByID(String adhocStartingPointID) - TWAdhocStartingPoint
● findAdhocStartingPointByName(String adhocStartingPointName) - TWAdhocStartingPoint
● validateMigrationTo(TWProcessAppSnapshot snapshot) - TWProcessInstanceValidation
● migrateTo(TWProcessAppSnapshot snapshot) - void

```

The "id" property contains the process instance ID of the process. This is encoded in the format:

2072.<PIID>

where the 2072 prefix is the marker that says that this is a process instance.

The addDocument method can add a document by file or by URI.

To find a process instance, we can use the JavaScript method:

```
tw.system.findProcessInstanceByID()
```

This has the syntax:

```
tw.system.findProcessInstanceByID(Number processInstanceID);  
tw.system.findProcessInstanceByID(String processInstanceID);
```

This returns an instance of a TWProcessInstance.

The current process (the one executing) can be found at the variable:

```
tw.system.currentProcessInstance
```

This variable is an instance of TWProcessInstance.

Note that there is a property called "businessData" which is a Map object containing all the variables in the process instance. It is believed that these are read-only and any attempt to change them will **not** result in changes to the variables in the process. Of deeper interest is the notion of a variable which is a Business Object. If a variable called "bo1" is of type BO1 which has properties "a" and "b", one would have imagined that we could retrieve an object called "bo1" from the map. Strangely, this isn't the case. Instead, the key to the map is the full path to each of the fields ... i.e. "bo1.a" and "bo1.b".

In addition there are some constants defined:

```
TWProcessInstance.Statuses
```

- Active - String
- Completed - String
- DidNotStart - String
- Failed - String
- Suspended - String
- Terminated - String

See also:

- Data Type – TWAdhocStartingPoint
- Data Type – TWProcess
- Cleaning/removing completed processes

Data Type – TWProcessPerformanceMetric

(The following as of 8.5)

```
fl retrieveProcessInstanceList(ProcessInstanceListProperties properties, Boolean checkAuthorization) - TWObject*ProcessInstanceListItem  
fl retrieveProcessSummary(String searchFilter, Boolean checkAuthorization) - ProcessSummary  
fl retrieveProcessHistoricalStatistics(ProcessHistoricalStatisticsProperties properties, Boolean checkAuthorization) - ProcessHistoricalStatistics  
fl retrieveActivitySummaries(String searchFilter, Boolean checkAuthorization) - TWObject*ActivitySummary  
fl retrieveInstanceTrend(String units, Integer numPeriods, String endPeriod, String timezone, String searchFilter, boolean checkAuthorization) - ChartData
```

This is a rich collection of methods. Some of these methods take rich data structures as parameters. As of 8.5, I haven't been able to find these adequately documented so here are notes on what I have found so far:

ProcessHistoricalStatisticsProperties

- includeAverageInstanceDuration (Boolean)
- includeGanttTaskStatistics (Boolean)
- searchFilter (String)

See also:

- Data Type – TWProcess
- Data Type – ProcessSummary

Data Type – TWReport

A TWReport represents or models a IBPM Report.

name - String
pageName - String
isEmbedded - Boolean
displayPage() - String
displayDefaultPage() - String
getChartInstance() - TWChart
getPageURL() - String
linkPageURL() - String
linkPageInNewWindowURL() - String
createPageLink() - String
createPageLinkInNewWindow() - String
getFilterValue() - String
setFilterValue() - VOID
removeFilterValue() - VOID
getFilterParameters() - Array
applyFilter() - String
getTypeName() - String

Static methods:

getByName(String reportName) - TWReport

A number of methods in this Object return HTML for inclusion in a page or Coach. These include:

createPageLink	
createPageInNewWindowLink	
displayDefaultPage	
displayPage	

See also:

- Error: Reference source not found

Data Type – TWRole

Within IBPM we have System Groups which are external definitions of collections of people. The TWRole object is the access to these System Groups. Loosely speaking, the phrases “Role” and “System Group” names can be used interchangeably. In discussions with users of versions of the ancestral TeamWorks product, there was a time when the concept of “Participant Groups” did not exist in the product and the only “user grouping” mechanism was the “Role”. The “Role” was used in BPMN swim lanes to describe the “role” that a user was to perform (eg. Insurance clerk vs Forklift driver)

TWRole as of 8.5:

```

● id - Integer
● name - String
● users - TWObject*TWUser
● roles - TWObject*TWRole
● containerRoles - TWObject*TWRole
● teamManagerRole - TWRole
● managedTeamRoles - TWObject*TWRole
● allUsers - TWObject*TWUser
fl addUsers(TWUser users, *TWUser users, String users, *String users) - VOID
fl removeUsers(TWUser users, *TWUser users, String users, *String users) - VOID
fl addRoles(TWRole roles, *TWRole roles, String roles, *String roles) - VOID
fl removeRoles(TWRole roles, *TWRole roles, String roles, *String roles) - VOID
fl getTypeName() - String

```

With the presumption that the `TWRole` is mapped to a System Group and a System Group is managed outside of the IBPM product (commonly as part of LDAP security), these doesn't appear to be any semantic mapping to the idea of performing the functions known as:

- `addUsers`
- `removeUsers`
- `addRoles`
- `RemoveRoles`

This data type can not be sent to `"log.info ()"`.

See also:

- Data Type – `TWParticipantGroup`
- Data Type – `TWRole`
- Data Type – `Team`
- Data Type – `TWUser`
- Security Groups

Data Type – `TWSearchColumn`

This data structure is used to define which columns to return on a JavaScript `TWSearch` request.

```

● name - String
● type - String
fl getTypeName() - String

```

The type must be one of (`TWSearchColumn.Types.*`). The available types are:

```

● Process - String
● BusinessData - String
● ProcessInstance - String
● Task - String

```

The name parameter must be one of the allowable column names:

`TWSearchColumn.ProcessInstanceColumns.*`

`TWSearchColumn.ProcessColumns.*`

`TWSearchColumn.TaskColumns.*`

Process Instance columns:

- DueDate - String
- ID - String
- Name - String
- Status - String
- ProcessApp - String
- Snapshot - String

Field	Description
DueDate	The date that the process is due to be completed
ID	The ID of this instance of the process
Name	The descriptive name of the process
Status	The status of the process
ProcessApp	The name of the ProcessApp that this process belongs to
SnapShot	The snapshot name that this process is associated with

Task columns:

- Activity - String
- AssignedToRole - String
- AssignedToUser - String
- ClosedBy - String
- ClosedDate - String
- DueDate - String
- ID - String
- Priority - String
- ReadDate - String
- ReceivedDate - String
- ReceivedFrom - String
- SentDate - String
- Status - String
- Subject - String

Field	Description
Activity	The name of the BPD activity that owns the task
AssignedToRole	
AssignedToUser	
ClosedBy	Who closed the task
ClosedDate	When was the task closed
DueDate	
ID	The ID of the task
Priority	The priority value of the task
ReadDate	
ReceivedDate	The date that the task was created
ReceivedFrom	
SentDate	
Status	The status of the task. For example Closed, Received, Comment ...
Subject	The subject of the task

Process Columns:

 [Name - String](#)

Field	Description
Name	The name of the template (BPD) of this process instance

See also:

- Searching for processes and tasks from JavaScript

Data Type – TWSearchColumnMetaData

This data type describes the data returned/available for a particular field search.

- name
- displayName
- type
- valueType – One of:
 - `TWSearchColumnMetaData.ValueTypes.String`
 - `TWSearchColumnMetaData.ValueTypes.Integer`
 - `TWSearchColumnMetaData.ValueTypes.DateTime`
 - `TWSearchColumnMetaData.ValueTypes.Boolean`
 - `TWSearchColumnMetaData.ValueTypes.Decimal`

This describes the possible data type that a value may have.

- isUsableInSearchCondition

Data Type – TWSearchCondition

The search condition names a column in the data, an operator to be executed against that column and a parameter for the operation.

-  [column - TWSearchColumn](#)
-  [operator - TWSearchConditionOperations](#)
-  [value - TWObject](#)
-  [getTypeName\(\) - String](#)

The operator is taken from the `TWSearchConditions.Operations.*`

-  [Contains - String](#)
-  [Equals - String](#)
-  [Greater Than - String](#)
-  [Less Than - String](#)
-  [Not Equals - String](#)
-  [Starts With - String](#)

See also:

- Searching for processes and tasks from JavaScript
- Data Type – TWSearchColumn

Data Type – TWSearchOptions

A TWSearchOptions instance provides additional parameters to a TWSearch(). The properties of this object are:

isTieBreakerSorting - boolean

- isTieBreakerSorting (Boolean) – When a search is performed and the results returned, there may not be enough information returned to uniquely sort the results. If the value of this option is false (default) then tied sorts are in an undefined order. However, if set to true, additional logical columns are added to the sort to disambiguate the results. The columns used for the additional sorting are:
 - instanceId
 - taskDueDate
 - taskPriority
 - taskId

Data Type – TWSearchResults

A TWSearchResults object is returned from the method calls of TWSearch().

It contains the following data:

- columns – Type TWSearchColumnMetaData
- rows – Type TWSearchResultRow

The number of results returned can be found in rows.length;

See also:

- Data Type – TWSearchResultRow
- Searching for processes and tasks from JavaScript

Data Type – TWSearchResultRow

This contains:

- values – The value returned. This is a list of objects (TWOObject). The ordinal of the value corresponds to the order of items to be returned.

The type also contains a method called getTypeName() which returns a description of the type.

See also:

- Data Type – TWSearchResults

Data Type – TWService

Properties		
id	String	
name	String	

type	String	The type of service that this object represents. See: Data Type – TWServiceTypes
------	--------	--

Methods	
Map execute(Map inputParams)	Execute the instance of this service.
String getTypeName()	Get the type that this service implements.

An instance of this object is returned by the global function called `tw.system.model.findServiceByName`. The signature of this function is:

```
TWService tw.system.model.findServiceByName(String serviceName)
```

Data Type – TWServiceTypes

Properties		
Ajax	String	An Ajax service implementation.
GeneralSystem	String	A General System service implementation.
Human	String	A Human service implementation.
Installation	String	An Installation service implementation.
Integration	String	An Integration service implementation.
Rule	String	A Rule service implementation.

Data Type – TWStep

The current step in the process (BPD) can be found from the variable:

```
tw.system.step
```

This returns a TWStep object.

(as of 8.5)

● id - int
● guid - String
● name - String
● counter - int
● processInstance - TWProcessInstance
● task - TWTask
● timer - TWTimerInstance
● error - XMLElement
● isConditionalActivitySelected - boolean
● links - *TWLink
● getTypeName() - String

The timer property is relevant if the step is a timer. The pre-assignment option of the BPD activity can be used to get the timerId and **not** the post-assignment value.

See also:

- Multi-Instance Looping
- Data Type – TWProcessInstance
- Data Type – TWTimerInstance

- Data Type – TWTTask
- Data Type – TWLink

Data Type – TWTTask

As a process instance operates, it can create tasks that users are expected to perform. The TWTTask data type represents a model of this task that can be accessed from BPM applications.

Within a service, the current task can be found from:

```
tw.system.currentTask
```

(As of 8.5.0)

<code>id - String</code>
<code>localId - String</code>
<code>subject - String</code>
<code>narrative - String</code>
<code>dueDate - TWDate</code>
<code>priority - String</code>
<code>status - String</code>
<code>transactions - *Map</code>
<code>assignedTo - TWObject</code>
<code>processInstance - TWProcessInstance</code>
<code>processInstanceStep</code>
<code>processActivityName - String</code>
<code>processActivityDescription - String</code>
<code>startDate - TWDate</code>
<code>completionTime - TWDate</code>
<code>priorityValue - Integer</code>
<code>atRiskDate - TWDate</code>
<code>isAtRisk - Boolean</code>
<code>phaseId - String</code>
<code>flowObjectId - String</code>
<code>state - String</code>
<code>activationTime - TWDate</code>
<code>originator - String</code>
<code>owner - String</code>
<code>f0 reassignTo(TWUser usersOrRoles) - void</code>
<code>f0 reassignTo(TWRole usersOrRoles) - void</code>
<code>f0 reassignTo(TWTeam usersOrRoles) - void</code>
<code>f0 reassignTo(String usersOrRoles) - void</code>
<code>f0 reassignTo(Array usersOrRoles) - void</code>
<code>f0 reassignTo(*String usersOrRoles) - void</code>
<code>f0 reassignBackToRole() - void</code>
<code>f0 start(TWUser user) - void</code>
<code>f0 complete(TWUser user, Map outputValues) - *long</code>
<code>f0 getAvailableActions(*String actionsFilter) - *String</code>
<code>f0 getTypeName() - String</code>

The assignedTo field contains the identity of the user or role to which the task is currently assigned. This may be either a TWUser or a TWRole.

In addition, there are some constant definitions found at:

TWTTask.Priorities

- Highest - String
- High - String
- Normal - String
- Low - String
- Lowest - String

TWTask.Statuses

- New - String
- Received - String
- Replied - String
- Forwarded - String
- Sent - String
- Actioned - String
- Closed - String
- Special - String
- Deleted - String

We can retrieve an arbitrary task as long as we know it's id. We can do this using:

```
tw.system.findTaskByID()
```

Notice that there are two priority properties. One is called "priority" and the other is called "priorityValue". These two values appear to be in sync with each other. The priority property contains a string representation while the priorityValue contains a numeric representation. Changing the priority of a task changes both values. The priorityValue can be used when a numeric comparison is required ... eg ... to answer questions such as "When priority is higher than 'normal'". This would not be easy to achieve if all we had were the string representation.

See also:

- Data Type – TWProcessInstance

Data Type – TWTeam

(As of 8.5)

This data type represents a team within IBM BPM. There are a number of APIs that can be used to obtain a team such as:

- tw.system.org.findTeam(teamId, processAppId)
- tw.system.org.findTeamByID(teamId)
- tw.system.org.findTeamByName(teamName)
- getAllTeams()

• id - String
• name - String
• processApp - TWProcessApp
• snapshot - TWProcessAppSnapshot
• associatedRole - TWRole
• users - *TWUser
• roles - *TWRole
• allUsers - *TWUser
• dashboard - TeamDashboardSupport
f() hasUser(TWUser user) - boolean
f() addUsers(TWUser users) - void
f() removeUsers(TWUser users) - void
f() addRoles(TWRole roles) - void
f() removeRoles(TWRole roles) - void
f() refresh() - void
f() getTypeName() - String
f() getManagerTeam() - TWTeam
f() setManagerTeam(String teamName) - void
f() asTeam() - Team
f() setAutomaticRefresh(boolean enable) - void

See also:

- Data Type – TWProcessApp
- Data Type – TWProcessAppSnapshot
- Data Type – TWTeam
- Data Type – TeamDashboardSupport
- Data Type – TWRole
- Data Type – TWUser

Data Type – TWTimePeriod

The TWTimePeriod is used to represent a period of time and which days that period is applicable.

• startTime - TWDate
• endTime - TWDate
• sunday - Boolean
• monday - Boolean
• tuesday - Boolean
• wednesday - Boolean
• thursday - Boolean
• friday - Boolean
• saturday - Boolean

- startTime – The time of the day at which the period begins
- endTime – The time of the day at which the period ends
- days – Boolean flags that indicate whether or not this time period applies for this day

See also:

- Data Type – TWDate
- Data Type – TWTimeSchedule

Data Type – TWTimeSchedule

A TWTimeSchedule represents an interval of time during which work is processed. An instance of this object can be created in JavaScript through:

```
var var1 = new tw.object.TWTimeSchedule();
```

Once created, the properties of this new object can be set. Once the TWTimeSchedule has been built, the new Time Schedule can be registered for use with the JavaScript API:

```
tw.system.addTimeSchedule(TWTimeSchedule timeSchedule);
```

To retrieve an existing schedule, execute:

```
tw.system.findTimeScheduleByName(String name)
```

This returns a TWTimeSchedule object.

To delete a previously registered Time Schedule, the JavaScript call is:

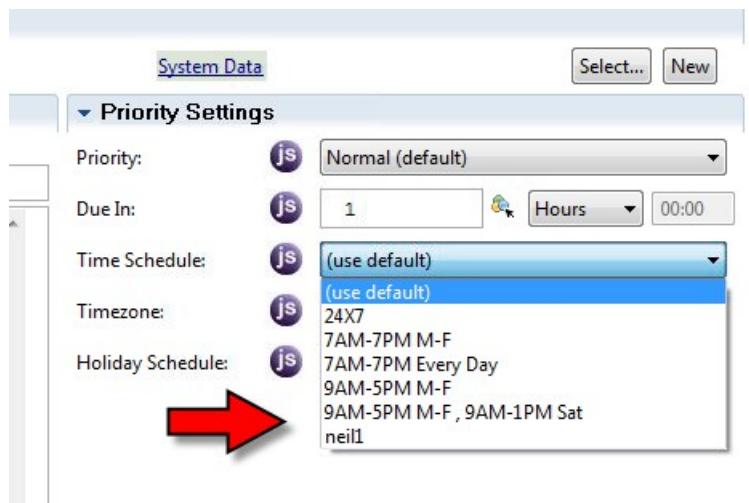
```
tw.system.removeTimeSchedule(String id);
```

```
id - String  
name - String  
excludeHolidays - Boolean  
periods - TWSObject*TWTimePeriod
```

The product provides a set of pre-existing Time Schedules

- 24X7 – 24 hours a day, every day
- "7AM-7PM M-F" – 7am to 7pm on Mon, Tue, We, Thu, Fri
- "7AM-7PM Every Day"
- "9AM-5PM M-F"
- "9AM-5PM M-F,9AM-1PM Sat"

The list of currently defined Time Schedules can be seen from the pull-down for activities:



These time schedules can be found in the list at `tw.system.timeSchedules`. The method:

```
tw.system.addTimeSchedule()
```

will add a time schedule to this list.

Here is some sample code for adding a Time Schedule:

```

var tim1 = new tw.object.TWTimeSchedule();
tim1.name = "neill";
tim1.periods = new tw.object.listOf.TWTimePeriod();

var period1 = new tw.object.TWTimePeriod();
period1.startTime = new tw.object.Date();
period1.endTime = new tw.object.Date();
period1.sunday = false;
period1.saturday = false;
period1.monday = true;
period1.tuesday = true;
period1.wednesday = true;
period1.thursday = true;
period1.friday = true;

tim1.periods[0] = period1;

var added = tw.system.addTimeSchedule(tim1);
log.info("ret = " + added);

```

There is an IBM BPM supplied function called `tw.system.calculateBusinessDate()` that will calculate a future date/time based on an original date, a logical increment and a `TWWorkSchedule`.

```

TWDate tw.system.calculateBusinessDate(
    TWDate originalDate,
    Integer delta,
    String units,
    TWWorkSchedule workSchedule)

```

The "units" property can be one of:

- Month
- Day
- Hour
- Minute

See also:

- Data Type – `TWTimePeriod`
- Due dates for process and activities

Data Type – `TWTimerInstance`

An instance of this type of object can be retrieved from an instance of the `TWStep` object through its `timer` property.

	<code>id</code> - String
	<code>gettypeName()</code> - String

The timer instance `id` may be used in the `tw.system.rescheduleTimer()` method. This takes two parameters. The first is the `id` of the timer instance. The second is the date/time at which the timer should fire. The primary purpose of this function is to re-schedule a ticking timer to fire at a different time. Take care when trying to obtain the `timerId` as it seems that we have to use the pre-assignment of the attached timer activity and not the post-assignment value.

See also:

- Data Type – `TWStep`
- Timer Events

Data Type – TWUser

Because various functions can return an instance of either a TWUser or a TWRole, the `gettypeName()` function is common to both. Calling this function for a TWUser object will return the String "TWUser". This can be used to determine which methods and properties are available based on the type of object it is determined to be.

```
• id - Integer  
• name - String  
• fullName - String  
• attributes - Record  
• roles - TWObject*TWRole  
• participantGroups - TWObject*TWParticipantGroup  
• savedSearches - *TWSavedSearch  
• scoreboards - *TWScoreboard  
• f0 setAttributeValue\(String name, String value\) - VOID  
• f0 removeAttribute\(String name\) - VOID  
• f0 isInRole\(Scriptable Role\) - Boolean  
• f0 isInParticipantGroup\(TWParticipantGroup participantGroup\)  
• f0 getTypeName\(\) - String
```

The property called `attributes` is a Record data structure that contains the properties for the user definition. Be cautious about using either the Record methods `toXML()` or `toXMLString()`. The reason for this is that the IBPM default attributes such as "Task Email Address" contain spaces in their names and the XML functions throw exceptions when they attempt to build XML using names with spaces as element names.

We can find a TWUser object if we know the userid/name of that user. The method

```
tw.system.org.findUserByName("<name>")
```

will return a TWUser instance.

This data type can not be sent to "log.info()".

See also:

- Data Type – TWParticipantGroup
- Data Type – TWRole
- JavaScript Package - `tw.system.org.*`

Data Type – XMLDocument

This data type is defined in the System Data toolkit.

```
• f0 transform\(String fileName\) - String  
• f0 createElement\(String elementName\) - XMLElement  
• f0 xpath\(String xPathExpression\) - XMLNodeList  
• f0 getElementByTagName\(String tagName\) - XMLElement  
• f0 getElements\(\) - XMLNodeList  
• f0 toString\(Boolean deep\) - String
```

See also:

- Data Type – XMLElement
- Data Type – XMLNodeList
- Working with XML in JavaScript

Data Type – XMLElement

The XMLElement object represents a node in a DOM tree. Each instance of this object provides navigators to the next/previous sibling, first/last child and the parent as well as access to the document as a whole. Through this object common DOM walks and searches can be executed to perform quite sophisticated data access and manipulation. Each node has the following:

- `tagName` – The name of the current element
- `attributes` – zero or more name/value attributes
- `text` – The textual content of the element

```
namespaceURI - String
ownerDocument - XMLDocument
parentNode - XMLElement
nextSibling - XMLElement
previousSibling - XMLElement
childNodes - XMLNodeList<XMLElement>
firstChild - XMLElement
lastChild - XMLElement
tagName - String
transform(String fileName) - String
xpath(String XPathExpression) - XMLNodeList<XMLElement>
hasChildNodes() - Boolean
toString(Boolean deep) - String
getElementsByTagName(String tagName) - XMLNodeList
getText() - String
setText(String text) - VOID
getAttribute(String attributeName) - String
setAttribute(String name, String value) - VOID
removeAttribute(String name) - VOID
getAttributeNS(String namespaceURI, String attributeName) - String
setAttributeNS(String namespaceURI, String name, String value) - VOID
removeAttributeNS(String namespaceURI, String name) - VOID
getElementsByTagNameNS(String namespaceURI, String localName) - XMLNodeList
insertBefore(XMLElement newChild, XMLElement refChild) - XMLElement
replaceChild(XMLElement newChild, XMLElement oldChild) - XMLElement
removeChild(XMLElement oldChild) - XMLElement
appendChild(XMLElement newChild) - XMLElement
cloneNode(Boolean deep) - XMLElement
```

We can walk through an XML document using the `firstChild` and `nextSibling` mechanisms. We can also access the children by name. For example:

- `x.y.z` – From the variable "x", access the child called "y" and access the property called "z"
- `x.y[0].z[1]` – From the variable "x", access the 1st (0th) child called "y" and access the 2nd (1st) property called "z".

See also:

- Data Type – XMLDocument
- Data Type – XMLNodeList

Data Type – XMLNodeList

A list of XMLElement objects.

 [length - Integer](#)
 [item\(Integer index\) - XMLElement](#)

See also:

- Data Type – XMLDocument
- Data Type – XMLElement

JavaScript Libraries

IBM BPM provides predefined JavaScript libraries that can be used in the applications.

Root:

 [tw](#)
 [log - TWLogger](#)
 [Packages](#)
 [java](#)
 [jax](#)
 [undefined](#)
 [this](#)
 [NaN](#)
 [eval\(String codestring\) - String](#)
 [isFinite\(Number testnumber\) - Boolean](#)
 [isNaN\(Number testnumber\) - Boolean](#)
 [Number\(ANY object\) - Number](#)
 [parseFloat\(String string\) - Number](#)
 [Date - Static Object](#)
 [Math - Static Object](#)
 [IntegrationComponent - Static Object](#)
 [TWReport - Static Object](#)
 [TWAttachedDocument - Static Object](#)
 [TWTTask - Static Object](#)
 [TWSearch - Static Object](#)
 [TWSearchCondition - Static Object](#)
 [TWSearchOrdering - Static Object](#)
 [TWSearchColumn - Static Object](#)
 [TWMManagedFile - Static Object](#)
 [TWSERVICE - Static Object](#)
 [TWProcessInstance - Static Object](#)

tw.*

 [local](#)
 [system](#)
 [perf](#)
 [epv](#)
 [object](#)
 [env](#)

JavaScript package - tw.system.*

The package called `tw.system.*` contains the following (as of 8.5):

From a service (8.5)

```

install - TWInstallNamespace
header
model
org - OrgNamespace
error - XMLElement
error_type - String
task_id - String
coachValidation - CoachValidation
currentProcessInstanceID - String
currentProcess - TWProcess
currentProcessInstance - TWProcessInstance
currentTask - TWTask
user_id - String
user_loginName - String
user_fullName - String
user_timeZone - String
user_timeZoneOffset - String
user_locale - String
user_localeDescription - String
user_timeZoneOffset - String
buttonPressed - String
temporary_directory - String
result - String
serializer - Serializer
user - TWUser
timeSchedules - *TWTimeSchedule
defaultTimeSchedule - TWTimeSchedule
holidaySchedules - *TWHolidaySchedule
defaultHolidaySchedule - TWHolidaySchedule
defaultTimeZone - String
allVars() - *String
rescheduleTimer(String timerId, TWDate newFireTime) - VOID
findTaskByID(Integer taskID) - TWTask
findTaskByID(String taskID) - TWTask
retrieveTaskList(TaskListProperties properties, Integer maxRows, Integer beginIndex, String timezone) - TaskListData
retrieveTaskDueData(TaskDueProperties properties, String timezone) - TaskDueData
retrieveProcessSummary(String processAppId, String processId, String searchFilter, Boolean checkAuthorization) - *ProcessSummary
retrieveProcessSummaries(String searchFilter, Boolean checkAuthorization) - *ProcessSummary
retrieveTeamSummaries(String searchFilter, String timeZoneAsString, Boolean checkAuthorization) - *TeamTaskSummary
toString() - String
variableTypeForVariable(String fullyQualifiedVariableName) - String
createFromSql(String variableTypeName, String jndiName, String sqlQuery, ANY params) - VOID
createFromXmlElement(String variableTypeName, Element xml) - VOID
addTimeSchedule(TWTimeSchedule timeSchedule) - Boolean
findTimeScheduleByName(String name) - TWTimeSchedule
removeTimeSchedule(String id) - Boolean
addHolidaySchedule(TWHolidaySchedule holidaySchedule) - Boolean
updateHolidaySchedule(TWHolidaySchedule holidaySchedule) - Boolean
updateTimeSchedule(TWTimeSchedule timeSchedule) - Boolean
findHolidayScheduleByName(String name) - TWHolidaySchedule
removeHolidaySchedule(String id) - Boolean
findProcessInstanceByID(String id) - TWProcessInstance
startProcessByName(String name, Map inputValues) - TWProcessInstance
executeServiceByName(String name, Map inputValues) - TWProcessInstance
convertIDToDB(String id) - String
findHelpRequestByID(String id) - TWHelpRequest
calculateBusinessDate(TWDate originalDate, Integer delta, String units, TWWorkSchedule workSchedule) - TWDate
bidiTransform(String src, String inputFormat, String outputFormat, Boolean symmetricSwapping) - String
addCoachValidationErrors(CoachValidation coachValidation, String errorBOPath, String errorMessage) - VOID
clearCoachValidationErrors(CoachValidation coachValidation) - VOID
removeCoachValidationErrors(CoachValidation coachValidation, String errorBOPath) - Boolean
updateCoachValidationErrorMessage(CoachValidation coachValidation, String errorBOPath, String errorMessage) - Boolean

```

From a BPD (8.5):

● process
● step
● org
● model
● user - TWUser
● timeSchedules - *TWTimeSchedule
● defaultTimeSchedule - TWTimeSchedule
● holidaySchedules - *TWHolidaySchedule
● defaultHolidaySchedule - TWHolidaySchedule
● defaultTimeZone - String
● install
● currentProcess - TWProcess
● currentProcessInstance - TWProcessInstance
● enclosingCaseInstance - CaseReference
f() rescheduleTimer(String timerId, TWDate newFireTime) - VOID
f() variableTypeForVariable(String fullyQualifiedVariableName) - String
f() createFromSql(String variableTypeName, String jndiName, String sqlQuery, ANY params) - String
f() createFromXmlElement(String variableTypeName, Element xml) - VOID
f() cancelStep(Integer stepId) - VOID
f() findTaskByID(Integer taskID) - TWTask
f() findTaskByID(String taskID) - TWTask
f() retrieveTaskList(TaskListProperties properties, Integer maxRows, Integer beginIndex, String timezone) - TaskListData
f() retrieveTaskDueData(TaskDueProperties properties, String timezone) - TaskDueData
f() retrieveProcessSummaries(*ProcessSummary searchFilter, Boolean checkAuthorization) - *ProcessSummary
f() retrieveTeamSummaries(String searchFilter, String timeZoneAsString, Boolean checkAuthorization) - *TeamTaskSummary
f() touchVariable(String name) - VOID
f() addTimeSchedule(TWTimeSchedule timeSchedule) - Boolean
f() findTimeScheduleByName(String name) - TWTimeSchedule
f() removeTimeSchedule(String id) - Boolean
f() addHolidaySchedule(TWHolidaySchedule holidaySchedule) - Boolean
f() updateHolidaySchedule(TWHolidaySchedule holidaySchedule) - Boolean
f() updateTimeSchedule(TWTimeSchedule timeSchedule) - Boolean
f() findHolidayScheduleByName(String name) - TWHolidaySchedule
f() removeHolidaySchedule(String id) - Boolean
f() findProcessInstanceByID(String id) - TWProcessInstance
f() startProcessByName(String name, Map inputValues) - TWProcessInstance
f() executeServiceByName(String name, Map inputValues) - TWProcessInstance
f() convertIDToDB(String id) - String
f() findHelpRequestByID(String id)
f() calculateBusinessDate(TWDate originalDate, Integer delta, String units, TWWorkSchedule workSchedule) - TWDate
f() bidiTransform(String src, String inputFormat, String outputFormat, Boolean symmetricSwapping) - String

Method: executeServiceByName(name, inputValues)

The documentation on this method appears to be wrong. It says it returns a TWProcessInstance where in reality, it seems to return a Map object which contains the output values. This makes more sense.

See also:

- Data Type – Map

Method: retrieveTaskList()

Retrieve a list of tasks.

Signature:

```
retrieveTaskList (TaskListProperties properties,
    Integer maxRows,
    Integer beginIndex,
    String timezone) - TaskListData
```

- properties – The properties passed to control how the retrieval is performed.
- maxRows – The maximum number of rows to return. If unspecified or 0 then all tasks are returned.
- beginIndex – The index of the start of the tasks to return. Use this parameter in combination with the TaskListData.total property to allow us to page through a retrieval.

See also:

- Data Type – TaskListData
- Data Type – TaskListProperties

JavaScript Package - tw.system.step.*

●	id - Integer
●	name - String
●	counter - Integer
●	processInstance - TWProcessInstance
●	task - TWTask
●	timer - TWTimerInstance
●	error - String
fl	getTypeName() - String

JavaScript Package - tw.system.org.*

(as of 8.5)

●	team
●	sla
fl	findUserById(String Integer userId) - TWUser
fl	findUserName(String userName) - TWUser
fl	findRoleById(String Integer roleId) - TWRole
fl	findRoleByName(String roleName) - TWRole
fl	findTeamByID(String teamId) - TWTeam
fl	findTeamByName(String teamName) - TWTeam
fl	findTeam(String teamId, String processAppId) - TWTeam
fl	findParticipantGroupByID(Integer participantId) - TWParticipantGroup
fl	findParticipantGroupByName(String roleName) - TWParticipantGroup
fl	findAllUsersForRoles(*TWRole roles) - *TWUser
fl	findCommonUsersForRoles(*TWRole roles) - *TWUser
fl	getAllUsers() - *TWUser
fl	getAllRoles() - *TWRole
fl	getAllTeams() - *TWTeam
fl	getAllParticipantGroups() - *TWParticipantGroup

See also:

- Data Type – TWUser
- Data Type – TWTeam

JavaScript Package - tw.system.model.*

As of 8.5.

env
processApp - TWProcessApp
processAppSnapshot - TWProcessAppSnapshot
searchMetaData - TWSearchMetaData
fullTextSearchMetaData - TWSearchMetaData
findProcessAppByID(String processAppID) - TWProcessApp
findProcessAppByName(String processAppName) - TWProcessApp
findProcessAppByAcronym(String processAppAcronym) - TWProcessApp
getAllProcessApps(boolean includeToolkits) - *TWProcessApp
getAllToolkits() - *TWProcessApp
findManagedFileByPath(String managedFilePath, String managedFileType) - TWManagedFile
findProcessByName(String processName) - TWProcess
findProcess(String bpdId, String processAppId) - TWProcess
findServiceByName(String serviceName) - TWSERVICE

This is a particularly interesting package. From within here we can interrogate the meta data of our environment.

See also:

- Data Type – TWProcess

Getting a list of Process Apps

In the following snippet, we see us getting the list of all process apps known to the system and logging their names. Each process app is described by a TWProcessApp object which has many other properties beyond name.

```
var listOfProcessApps = tw.system.model.getAllProcessApps();
for (var i=0; i<listOfProcessApps.length; i++) {
    log.info(listOfProcessApps[i].name);
}
```

See also:

- Data Type – TWProcessApp
- Data Type – TWProcessAppSnapshot
- Data Type – TWProcess
- Data Type – TWSERVICE
- Data Type – TWManagedFile

JavaScript Package - tw.object.*

[SQLResultSetRow\(\)](#) - SQLResultSetRow
[SLAViolationRecord\(\)](#) - SLAViolationRecord
[SQLParameter\(\)](#) - SQLParameter
[NameValuePair\(\)](#) - NameValuePair
[String\(\)](#) - String
[XMLDocument\(\)](#) - XMLDocument
[TWTimePeriod\(\)](#) - TWTimePeriod
[ANYO](#)
[Time\(\)](#) - Time
[Boolean\(\)](#) - Boolean
[TWHolidaySchedule\(\)](#) - TWHolidaySchedule
[Integer\(\)](#) - Integer
[SQLDatabaseType\(\)](#) - SQLDatabaseType
[Decimal\(\)](#) - Decimal
[Map\(\)](#) - Map
[XMLElement\(\)](#) - XMLElement
[SQLResultSetColumn\(\)](#) - SQLResultSetColumn
[Date\(\)](#) - Date
[TWTimeSchedule\(\)](#) - TWTimeSchedule
[XMLNodeList\(\)](#) - XMLNodeList
[TWWorkSchedule\(\)](#) - TWWorkSchedule
[Record\(\)](#) - Record
[SQLResult\(\)](#) - SQLResult
[SQLStatement\(\)](#) - SQLStatement
[IndexedMap\(\)](#) - IndexedMap
[listOf](#)

JavaScript Package - tw.system.model

- [TWManagedFile](#)
- [TWProcess](#)
- [TWProcessApp](#)
- [TWProcessAppSnapshot](#)
- [TWSavedSearch](#)
- [TWScoreboard](#)
- [TWSearch](#)
- [TWSearchColumn](#)
- [TWSearchColumnMetaData](#)
- [TWSearchCondition](#)
- [TWSearchMetaData](#)
- [TWSearchOrdering](#)
- [TWSERVICE](#)

Methods:

All methods part of the `tw.system.model` package

Method	Description
--------	-------------

findManagedFileByPath	Returns an instance of the managed file represented by the call <code>TWManagedFile</code>
findProcessAppByAcronym	Returns a <code>TWProcessApp</code> based on the acronym of the Process Application or Toolkit
findProcessAppByID	Returns a <code>TWProcessApp</code> based on the identifier of the Process Application or Toolkit
findProcessAppByName	Return a <code>TWProcessApp</code> based on its name.
findProcessByName	Returns a <code>TWProcess</code> based on the name.
findServiceByName	Returns a <code>TWSERVICE</code> based on the name.
findParticipantGroupByName	
findPartticipantGroupByID	
getAllProcessApps	Returns an array of <code>TWProcessApp</code> for all the Process Apps and Toolkits
getAllToolkits	Returns an array of <code>TWProcessApp</code> for all the Toolkits

tw.system

- `TWLogger`
- `TWAdhocStartingPoint`
- `TWDocument`
- `TWEVENT`
- `TWHelpRequest`
- `TWHolidaySchedule`
- `TWProcessInfo`
- `TWProcessInstance`
- `TWProcessInstanceSharepoint`
- `TWProcessStepInfo`
- `TWReport`
- `TWSearchResults`
- `TWSharepointForum`
- `TWTASK`
- `TWTimePeriod`
- `TWTimerInstance`

- TWTimeSchedule
- TWWorkSchedule

Methods for tw.system

Method	Description
rescheduleTimer	
variableTypeForVariable	
createFromSql	
createFromXmlElement	
cancelStep	
touchVariable	
addTimeSchedule	
findTimeScheduleByName	
removeTimeSchedule	
addHolidaySchedule	
findHolidayScheduleByName	
removeHolidaySchedule	
startProcessByName	Start an instance of a named process. Returns a TWProcessInstance.
executeServiceByName	
convertIDToDB	
findHelpRequestById	
calculateBusinessDate	

TWProcess attributes			
id	String	ro	The ID of the BPD
name	String	ro	The name of the BPD
description	String	ro	The description of the BPD
searchMetaData	TWSearchMetaData	ro	
processApp	TWProcessApp	ro	
snapshot	TWProcessAppSnapshot	ro	

TWProcess methods	
getTypeName	
startNew	

Creating Business Object instances in JavaScript

In IBPM PD we can create custom Business Object data type definitions. When a Business Object type variable is created, it must be initialized. This can be done through JavaScript with the code:

```
tw.local.myVariable = new tw.object.ComplexType();
```

A similar story is also true for data of type list:

```
tw.local myListOfStrings = new tw.object.listOf.String();
```

Variables in a service

Variables in a service are defined in the variables section of that service. Three types of variables can be defined. These three types are input, output and private.

A JavaScript function called `tw.local.allVars()` returns a list of string names of the locally defined variables in the current service.

A function called `tw.local.toXML()` returns an XML Element object containing all the locally defined variables.

It is important to note that even though these look like JavaScript variables, the complex ones are **not**. They are in fact Java classes accessed through JavaScript syntax. This has an important ramification if JavaScript methods are used against them thinking that they are JavaScript objects. Many such functions imply won't work.

Data Type	JavaScript "typeof" result
String	string
Boolean	boolean
Date	object
Decimal	number
Integer	number
Time	object

Complex data types can be serialized to XML with the following:

The Dojo Toolkit for JavaScript

Primarily of value to client-side JavaScript coding which executes in the browser, we may wish to use the services of the Dojo Toolkit. The Dojo Toolkit is supplied and loaded by Coach pages.

Dojo is an open source set of JavaScript libraries that enhance the capabilities of native JavaScript. Dojo supplies both basic and advanced JavaScript functions. For details on the Dojo toolkit see:

- [Dojo API](#)
- [Dojo Documentation](#)
- [Dojo Home Page](#)

Because Dojo is included with the IBPM coaches, the Dojo Toolkit functions can be used with custom solutions. Here are some of the more common Dojo patterns:

```
var x = dojo.byId("name of control");
```

This returns the DOM node of the control by name.

Commonly, Dojo is thought of as a visual language for building rich web pages. Although this is an important component of the Dojo Toolkit, it is not its sole function. However, if you are -

approaching Dojo for building rich user interfaces, strongly consider using IBM's Rational Application Developer v8 (RAD) for development of the HTML and JavaScript for those functions. RAD includes a rich UI designer for Dojo development.

Searching for processes and tasks from JavaScript

The ability to search within IBPM's data is all about asking IBPM for information about the current operating environment. Examples of a search may include:

- Finding a task in the current process

At a high level, the JavaScript object of type `TWSearch` can be used to execute a search. It has three primary methods:

- `execute()`
- `executeForProcessInstances()`
- `executeForTasks()`

The `TWSearch` object has a number of properties that are used to govern the data queried for and returned.

- `TWSearch.columns` – The columns of data to be returned
- `TWSearch.conditions` – The queries to be executed (of type `TWSearchCondition`)

(As of 8.5)

organizedBy - String
columns - *TWSearchColumn
conditions - *TWSearchCondition
orderBy - *TWSearchOrdering
options - TWSearchOptions
execute(TWUser TWRole String userOrRole, Integer maxRows, Integer beginIndex) - TWSearchResults
executeForTasks(TWUser TWRole String userOrRole, Integer maxRows, Integer beginIndex) - *TWTasks
executeForProcessInstances(TWUser TWRole String userOrRole, Integer maxRows, Integer beginIndex) - *TWProcessInstances
getTypeName() - String

A call to `TWSearch` using the `execute()` method can return a `TWSearchResults` object.

For reference, the list of columns is shown below:

Tasks	Instances	Process	Business Data
Activity	DueDate	Name	
AssignedToRole	ID		
AssignedToUser	Name		
ClosedBy	Status		
ClosedDate	ProcessApp		
DueDate	Snapshot		
ID			
Priority			

ReadDate			
ReceivedDate			
ReceivedFrom			
SentDate			
Status			
Subject			

Example:

```

log.info("Starting to find other tasks ....");
log.info("This process: " + tw.system.currentProcessInstance.id);

var col1 = new TWSearchColumn();
col1.name = TWSearchColumn.ProcessInstanceColumns.ID;
col1.type = TWSearchColumn.Types.ProcessInstance;

var search = new TWSearch();
search.columns = [col1];

var condition = new TWSearchCondition();
condition.column = new TWSearchColumn();
condition.column.name = TWSearchColumn.ProcessInstanceColumns.ID;
condition.column.type = TWSearchColumn.Types.ProcessInstance;
condition.operator = TWSearchCondition.Operations.Equals;
condition.value = "270";

search.conditions = new Array(condition);

var order1 = new TWSearchOrdering();
order1.column = col1;
order1.order = TWSearchOrdering.Orders.Descending;

search.orderBy = new Array(order1);
search.organizedBy = TWSearch.OrganizeByTypes.ProcessInstance;

var results = search.execute();
log.info("Result.rows.length = " + results.rows.length);
for (var i=0; i<results.rows.length; i++)
{
}

```

Notice that the `TWSearch execute` methods have an optional user/role filter. This allows the list of tasks/processes to be retrieved as though they were for a specific user. If this parameter is not specified, then a list of **all** Tasks/Instances are returned for all users.

Here is an example of some JavaScript which will retrieve all the tasks for all users:

```

log.info("Starting to find other tasks ....");

var col1 = new TWSearchColumn();
col1.name = TWSearchColumn.ProcessInstanceColumns.ID;
col1.type = TWSearchColumn.Types.ProcessInstance;

var col2 = new TWSearchColumn();
col2.name = TWSearchColumn.TaskColumns.ID;
col2.type = TWSearchColumn.Types.Task;

var col3 = new TWSearchColumn();
col3.name = TWSearchColumn.TaskColumns.Subject;
col3.type = TWSearchColumn.Types.Task;

var search = new TWSearch();
search.columns = [col1, col2, col3];

search.organizedBy = TWSearch.OrganizeByTypes.Task;

var order1 = new TWSearchOrdering();

```

```

order1.column = col2;
order1.order = TWSearchOrdering.Orders.Descending;
search.orderBy = new Array(order1);

var results = search.execute();
log.info("Result.rows.length = " + results.rows.length);
for (var i=0; i<results.rows.length; i++)
{
    //log.info(results.rows[i].values);
    var currentTaskInstance = new tw.object.TaskInstance();
    currentTaskInstance.taskId = results.rows[i].values[1];
    currentTaskInstance.processId = results.rows[i].values[0];
    currentTaskInstance.taskSubject = results.rows[i].values[2];
    tw.local.taskInstance[tw.local.taskInstance.listLength] = currentTaskInstance;
}

```

Here is another example. In this one, we are looking for all tasks associated with the process called "BAM1" which are not completed. From this, we want to know how many tasks are claimed and how many are not claimed.

```

var col1 = new TWSearchColumn();
col1.name = TWSearchColumn.ProcessColumns.Name;
col1.type = TWSearchColumn.Types.Process;

var col2 = new TWSearchColumn();
col2.name = TWSearchColumn.TaskColumns.Activity;
col2.type = TWSearchColumn.Types.Task;

var col3 = new TWSearchColumn();
col3.name = TWSearchColumn.TaskColumns.Status;
col3.type = TWSearchColumn.Types.Task;

var col4 = new TWSearchColumn();
col4.name = TWSearchColumn.TaskColumns.AssignedToUser;
col4.type = TWSearchColumn.Types.Task;

var search = new TWSearch();
search.columns = [col1, col2, col3, col4];

var condition = new TWSearchCondition();
condition.column = new TWSearchColumn();
condition.column.name = TWSearchColumn.ProcessColumns.Name;
condition.column.type = TWSearchColumn.Types.Process;
condition.operator = TWSearchCondition.Operations.Equals;
condition.value = "BAM1";

var condition2 = new TWSearchCondition();
condition2.column = new TWSearchColumn();
condition2.column.name = TWSearchColumn.TaskColumns.Status;
condition2.column.type = TWSearchColumn.Types.Task;
condition2.operator = TWSearchCondition.Operations.Equals;
condition2.value = "Received";

search.conditions = [condition, condition2];

var order1 = new TWSearchOrdering();
order1.column = col1;
order1.order = TWSearchOrdering.Orders.Descending;

search.orderBy = new Array(order1);
search.organizedBy = TWSearch.OrganizeByTypes.Task;

var results = search.execute();
log.info("Result.rows.length = " + results.rows.length);

var claimedCount = 0;
var notClaimedCount = 0;
for (var i=0; i<results.rows.length; i++) {
    if (results.rows[i].values[3] == null) {
        notClaimedCount++;
    } else {
        claimedCount++;
    }
    log.info(results.rows[i].values[0] + ":" + results.rows[i].values[1] + ":" +
results.rows[i].values[2] + ":" + results.rows[i].values[3]);
}

```

```
log.info("Claimed = " + claimedCount + ", Not Claimed = " + notClaimedCount);
```

Note: There is an unconfirmed forum post that claims that more than 14 columns requested to be returned breaks the search.

If some data you are looking for is not part of the data available in a search, remember that a search may return a process instance or task instance id that can then be used to retrieve a lot more details on that process or task instance that may include the information you are looking for.

See also:

- Data Type – TWSearchColumn
- Data Type – TWSearchColumnMetaData
- Data Type – TWSearchCondition
- Data Type – TWSearchResults
- Data Type – TWSearchResultRow
- Getting Task and Instance details through REST

Calling Java through LiveScript

JavaScript in IBPM can invoke Java through the Live Connect technology (see: (ext) [LiveConnect](#)). This is not a performance optimal solution but can provide critical functions/features when needed.

Examples of using this include:

Getting the current host-name:

```
tw.local.hostname = Packages.java.net.InetAddress.getLocalHost().getHostName();
```

Working with XML in JavaScript

The IBPM supplied framework provides capabilities to work with XML in the JavaScript environment. These capabilities are based on three supplied classes:

- XMLDocument
- XMLElement
- XMLNodeList

The XMLDocument is the representation of the DOM model of an XML document. An instance of the XMLDocument can be created from its constructor with:

```
var xmlString = "<A><B>123</B></A>";
var myDocument = new tw.object.XMLDocument(xmlString);
```

An XPath expression interpreter is supplied which can walk into the document and return a list of nodes in the document corresponding to the pattern:

```
var myList = myDocument.xpath("/A/B");
```

Once we have a node list, we can determine how many (if any) members are in it with its length property. We can retrieve an XMLElement object with the item(index) method.

```
var myElement = myList.item(0);
```

We can also get the value of elements with the `getText()` method:

```
var myText = myElement.getText();
```

We can walk the tree using “dot” notation. So if `myVar` is an `XMLElement`, we can code:

```
myVar.A.B.getText() to get the value of <A><B>....
```

See also:

- Data Type – `XMLDocument`
- Data Type – `XMLElement`
- Data Type – `XMLNodeList`

Working with document attachments in JavaScript

Document attachments can be associated with Process Instances. These can be added through Coach Controls or through programmatic additions. The relevant JavaScript components are:

- `TWDocument` – Data type. See: Data Type – `TWDocument`.
- `tw.system.findDocumentByID()`
- `TWProcessInstance.documents` – This property is an array of `TWDocument` objects associated with an instance of a process.
- `TWProcessInstance.addDocument()` – This function adds a document to a process instance.
- `TWProcessInstance.findDocuments()` – This function locates documents associated with the current instance.

The `addDocument()` function has the following parameters:

<code>type:String</code>	This is either <code>TWDocument.Types.URL</code> or <code>TWDocument.Types.File</code> .
<code>name:String</code>	This is the name of the document.
<code>fileLocation:String</code>	
<code>hideInPortal:Boolean</code>	This is a flag which describes whether or not the document is to be visible in the Process Portal.
<code>createdBy:TWUser</code>	
<code>properties:Map</code>	

Note: Experimentation seems to show that the `addDocument()` function will **not** work when called from JavaScript locally coded in a BPD activity but seems to work just fine when coded in a General Service.

Here is an example of calling `addDocument()`:

```
var myMap = new tw.object.Map();
var hide = false;
var user = tw.system.user;
tw.system.currentProcessInstance.addDocument(
    TWDocument.Types.File,
    "name",
    "C:\\\\Projects\\\\WLE\\\\Images\\\\ibm.jpg",
    hide,
    user,
```

```
myMap) ;
```

Interestingly, there is no exposed API to obtain the content of a document or set the content of an attached document from within JavaScript. We can write the content of the document to a local file but that is about it.

See also:

- Error: Reference source not found
- Error: Reference source not found

Working with JSON in IBPM

JSON is a string representation of a JavaScript object that can be transmitted over the network. It is primarily used for parameter passing. An Open Source JavaScript implementation is available to both parse and construct JSON strings.

<http://www.json.org/js.html>

Part of this package is a JavaScript source file called "json2.js". This file should be added as a Server managed file to your toolkit or process application. This will provide some new global methods/objects. The core of these is:

```
JSON.stringify(JavaScript Object) ;
```

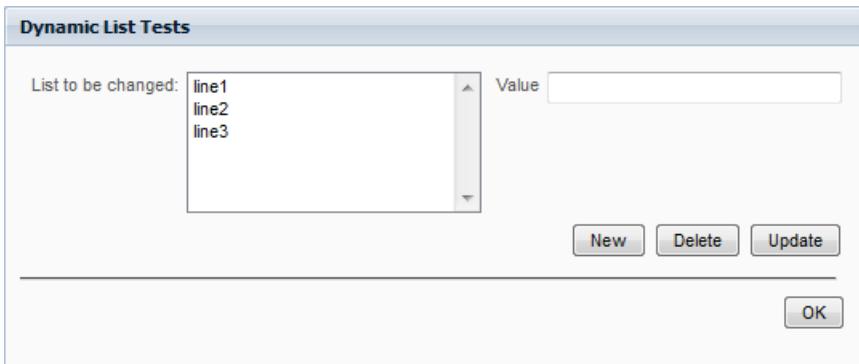
When invoked, this will return a String representation of the JavaScript object parameter. This string is a JSON string.

Unfortunately, even though many data structures within IBPM behaves in many ways like JavaScript objects, under the covers they are not JavaScript objects and don't always work but we can often code work-arounds. Here is an example of taking a List of Complex data structures and building a JSON representation:

```
var newArray = new Array();
for (var j=0; j<tw.local.myPurchase.listLength; j++)
{
    var newObj = new Object();
    for (var property in tw.local.myPurchase[j].propertyNames)
    {
        var name = tw.local.myPurchase[j].propertyNames[property];
        newObj[name] = tw.local.myPurchase[j][name];
    }
    newArray.push(newObj);
}
var jsonText = JSON.stringify(newArray);
log.error("jsonText = " + jsonText);
```

Within a browser/Coach, we can use the Dojo classes to work with JSON. The method `dojo.fromJson(string)` will parse a JSON string to return a JavaScript object while `dojo.toJson(Object)` will return a JSON string given a JavaScript object.

JSON can also be used in conjunction with Coaches. For example, imagine a Coach that looks as follows:



In this story, we have a list presented. We want to be able to perform add, delete and update operations on the list. This is not normally possible with the basic Coach controls as they are usually display only. The solution to this puzzle is to allow the list to be modified at the DOM/HTML level but, when the list is to be sent back to the server, we encode the list as a JSON string and return that. At the server, the JSON is parsed and a corresponding String list is rebuilt at the JavaScript level.

The way that a JSON string is returned from the Coach is to include a text field in the coach but flag it as hidden. It will not appear on the screen but can be set at the HTML level in the browser. This value will be returned back to the server. The following code fragment can then be executed at the server. This expects a JSON encoded array of strings to be returned into the variable `tw.local.returnText`. The result will be a populated `tw.local.myData` array of strings.

```
// Decode the returned text in case it is HTML encoded
var jsonText =
Packages.org.apache.commons.lang3.StringEscapeUtils.unescapeHtml4(tw.local.returnText);

// Parse the text as a JSON string
var localList =
JSON.parse(Packages.org.apache.commons.lang3.StringEscapeUtils.unescapeHtml4(jsonText));

// It is expected that the JSON returned is an array of strings
// now we want to convert the JSON string into a BPM array of strings
tw.local.myData = new tw.object.listOf.String();
for (var i=0; i<localList.length; i++)
{
    tw.local.myData[i] = localList[i];
}
```

One interesting area to note is the use of the `org.apache.commons.lang3.StringEscapeUtils` package. This is a JAR file added to the server managed files. The JAR comes from the Apache Commons Lang environment. It is used to decode HTML returned data. For example, the data returned:

`"hello"`

would be returned as

`"hello"`

The `unescapeHtml4()` method converts it back to the expected format. Although this example illustrates the use of simple data types, more complex structures can also be worked upon.

A really useful tool for working with raw JSON data is the web page found here:

<http://www.jsoneditoronline.org/>

Into this page, one can post JSON data and have it reformatted/parsed for readability.

JavaScript Fragments

When implementing a Process Application, it is likely that JavaScript coding will be required in some small or large part. This section provides a series of commonly occurring fragments of JavaScript code and recipes that can be reused in those solutions.

Starting a new process

A new instance of a process can be started by using the `tw.system.startProcessByName()` function. This function takes two parameters. The first is the name of the Process to be started. The second is a Map containing the input parameters for the process. An instance `TWProcessInstance` is returned. For example, this will start a new instance of a process with data:

```
var inputs = new tw.object.Map();
inputs.put("parm1", "parm1 value");
inputs.put("parm2", "parm2 value");
tw.system.startProcessByName("StartProcess2", inputs);
```

Note that the process will start executing concurrently.

Getting the current process instance

The current process instance can be retrieved through the variable called:

```
tw.system.currentProcessInstance
```

The data type of that object is `TWProcessInstance`.

See also:

- Data Type – `TWProcessInstance`

Getting the current userid

The current userid is available as a `TWUser` object from:

```
tw.system.user
```

Realize that at the BPD level, JavaScript based BPD activities do not run as a particular user so `tw.system.user` will return 'undefined'.

Starting an external application

If we wish to start an application that is external to IBPM such as a batch file or shell script, we can perform that task by using the JavaScript LiveScript mechanism. The following fragment will execute a batch file on a Windows system and can be placed in a JavaScript fragment:

```
var runtime = java.lang.Runtime.getRuntime();
runtime.exec("C:\\RunMe.bat");
```

Returning the owner of a task

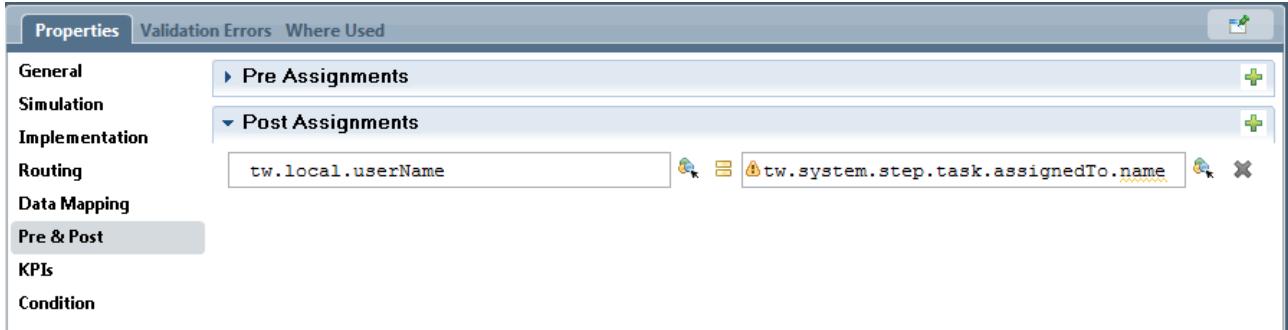
When a task is completed, we may wish to know the identity of the user that completed the task. One way to achieve this is to define an output parameter for the Human Service that is the name of the user that completed the task. Within the Human Service, prior to its termination, this identity can be determined with the following code:

```
var user = tw.system.currentTask.assignedTo;
```

The data type returned is an instance of `TWUser`.

The property called `name` of this data type can be used to determine the userid of the user that claimed the task.

As an alternative to returning the userid of the task from within the Human Service, a post-assignment can be performed on the Human Service. In the Post Assignment, the variable `tw.system.step.task.assignedTo` is again a `TWUser` that holds the `TWUser` that owned the Human Task.



Note: As of 2014-01, there appears to be a bug in the entry assist for this variable. When we ask for entry assist for `tw.system.step.task.assignedTo`, the PD tool seems to want to think of this as a "list" when it is in fact a straight `TWUser`.

See also:

- Data Type – `TWUser`

Extracting a managed file

If a Process App or a Toolkit contains a managed file, on occasion we may want to extract that from the run-time to see what it contains. One way to achieve this is to execute the following JavaScript:

```
var managedFile = tw.system.model.findManagedFileByPath("integration.jar",
TWManagedFile.Types.Server);
log.error("Got the managed file!: " + managedFile);
managedFile.writeDataToFile("C:\\\\integration.jar");
```

This will retrieve the named managed file and write its contents of a local file.

Generating a Random Number or string

There are times, especially during development and testing, where we might want a random number in our solution. This could perhaps be used to determine which path to take in a process or some similar notion. An easy solution is to leverage the `java.util.Random` class.

```
var rand = new java.util.Random();
var value = rand.nextInt(10);
```

This will generate a random integer between 0 and 9 (inclusive).

If we want a random string of a certain length, there are excellent JavaScript fragments on the Internet to achieve that. See the references below.

See also:

- stackoverflow – [Generate a string of 5 random characters in JavaScript](#)

Key Performance Indicators (KPIs)

Key Performance Indicators (KPIs) are data items that are saved for later examination and are captured during the execution of instances of processes. These KPIs can be used to measure the performance or other quality metrics of an IBPM solution. As a Process Server runs solutions, the KPIs for these processes are recorded. The data for the KPIs is stored in the Performance Data Warehouse database associated with the Process Server. The product provides a set of pre-defined KPIs.

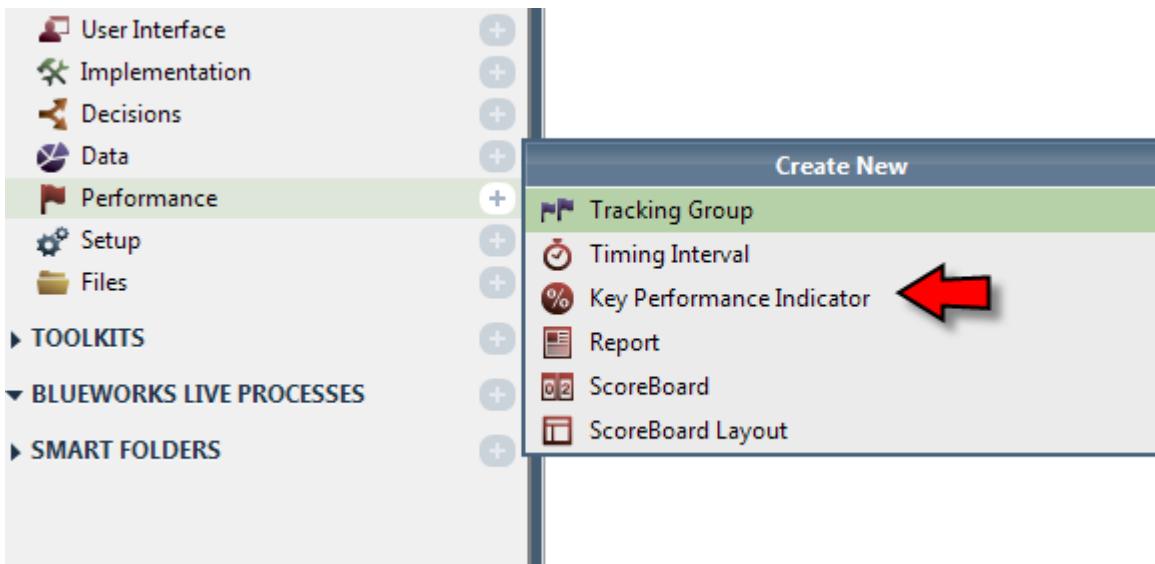
▼ Key Performance Indicator	
	Cost - 7.5.1
	Execution Time (Clock) - 7.5.1
	Labor Cost - 7.5.1
	Resource Cost - 7.5.1
	Rework - 7.5.1
	Total Time (Clock) - 7.5.1
	Value Add - 7.5.1
	Wait Time (Clock) - 7.5.1

KPI values are recorded when auto-tracking is enabled. The values for the KPIs are recorded in the Auto Tracking tables in the columns:

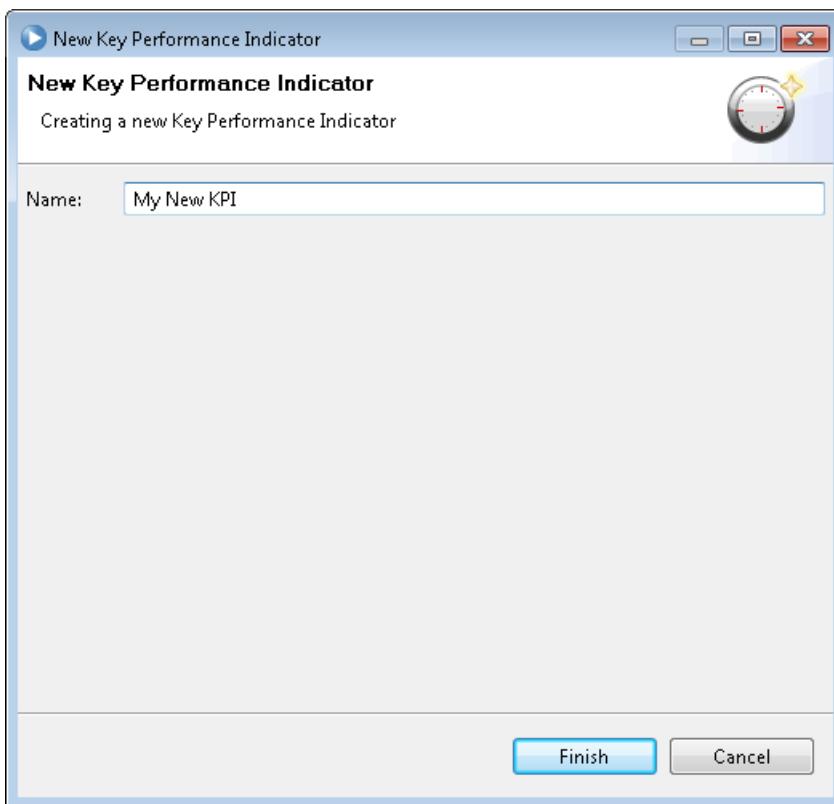
KPI Name	Table Column Name
Cost	KPICOST
Execution Time	KPIEXECUTIONTIMECLOCK
Labor Cost	KPILABORCOST
Resource Cost	KPIRESOURCECOST
Rework	KPIREWORK
Total Time	KPITOTALTIMECLOCK
Value Add	KPIVALUEADD
Wait Time	KPIWAITTIMECLOCK

Custom KPIs

IBPM provides a series of KPIs out of the box but it also allows for the creation of custom ones. KPIs can be created from the Performance category of the Library.



When the KPI is being created, it can be given a name:



The details section of the KPI looks as follows:

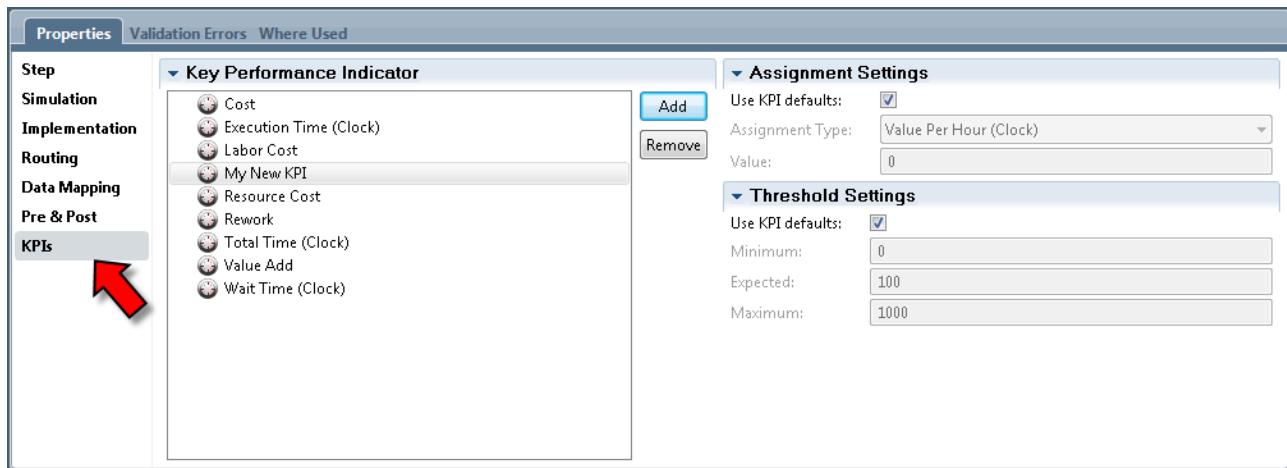
The screenshot shows the 'Key Performance Indicator' configuration screen. The 'Common' tab is selected, displaying fields for Name (My New KPI), Modified (tw_admin (September 22, 2010 10:18:41 AM CDT)), and Documentation. The 'Details' tab is also visible, showing Unit (Count), Roll-up KPI (Resource Cost, System Data), and Roll-up multiplier (1.0000000). A percentage icon (%) is located in the top right corner of the window.

The Unit data types available are:

- Boolean
- Count
- Currency
- Time

Associating KPIs with BPD activities

When an activity in the BPD diagram is selected, we find that it has a KPIs tab. This allows us to associate a KPI with that activity.



As the activity completes, there has to be an assignment to the KPI associated with that activity.

For KPIs that are of unit type Time, there are no further options. The KPI is the amount of time consumed by the execution of the activity.

For the other types (Boolean, Count and Currency), the assignment to the KPI value can be governed by the settings for that activity.

The assignment types allowable are:

- Absolute Value
- Value Per Hour (Clock)
- Value Per Hour (Calendar)
- Custom JavaScript

Service Level Agreements (SLAs)

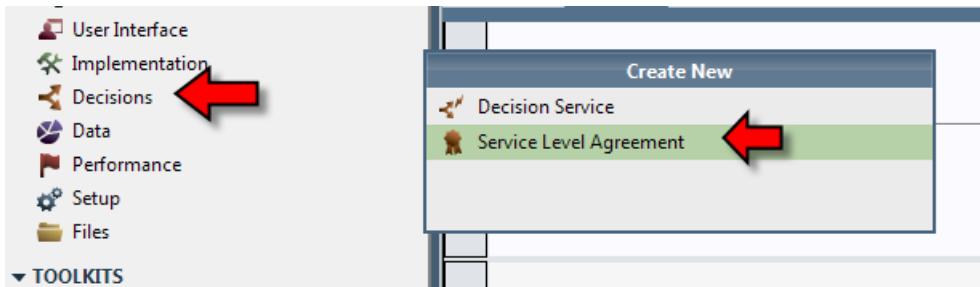
A Service Level Agreement (SLA) can be thought of a target of service performance or quality that is desired to be met during the execution of a process. An example of an SLA might be "When an order is received, it is shipped within 3 days". As processes execute, we *hope* that they meet these SLAs. IBPM can be configured to monitor the operation of processes and if an SLA condition is not being met, trigger some attention that will inform us that we have an issue that needs addressed. When an SLA is not met, we say that the SLA has been **violated**.

IBPM does **not** trigger an SLA exception when that logical exception occurs. Instead, IBPM

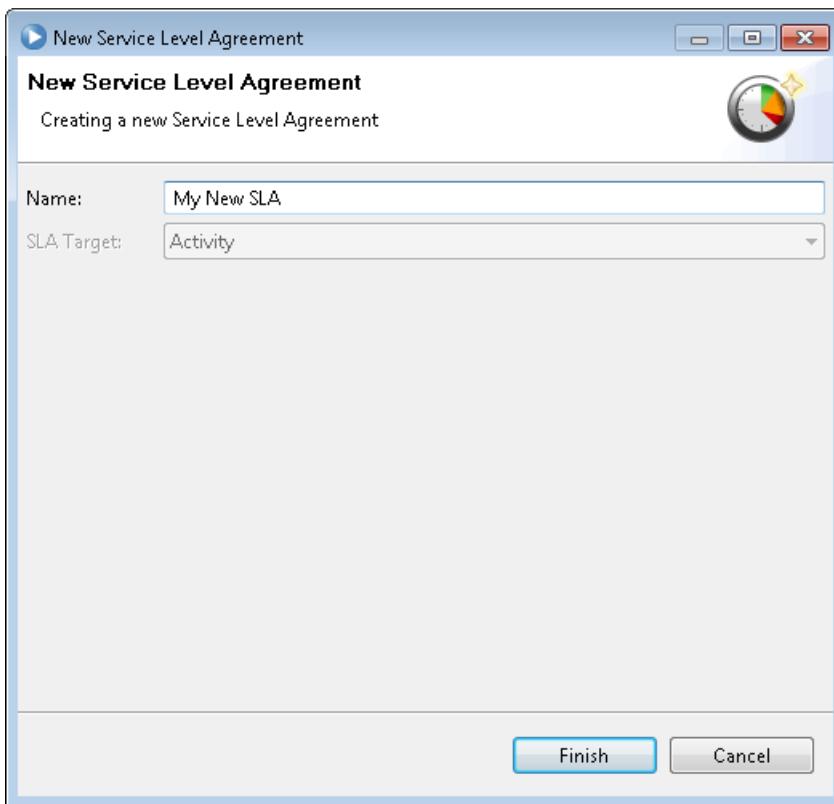
evaluates SLAs **only** when an activity starts or completes. What this means is that if an SLA is violated for a process, we aren't informed until perhaps long after an SLA has failed. The IBPM environment informs only that past SLAs were not met. If real time notification that a time based SLA has been violated is required then Timer Events can be associated with activities to trigger actions if the activities have not been completed within a certain period.

Creating SLAs

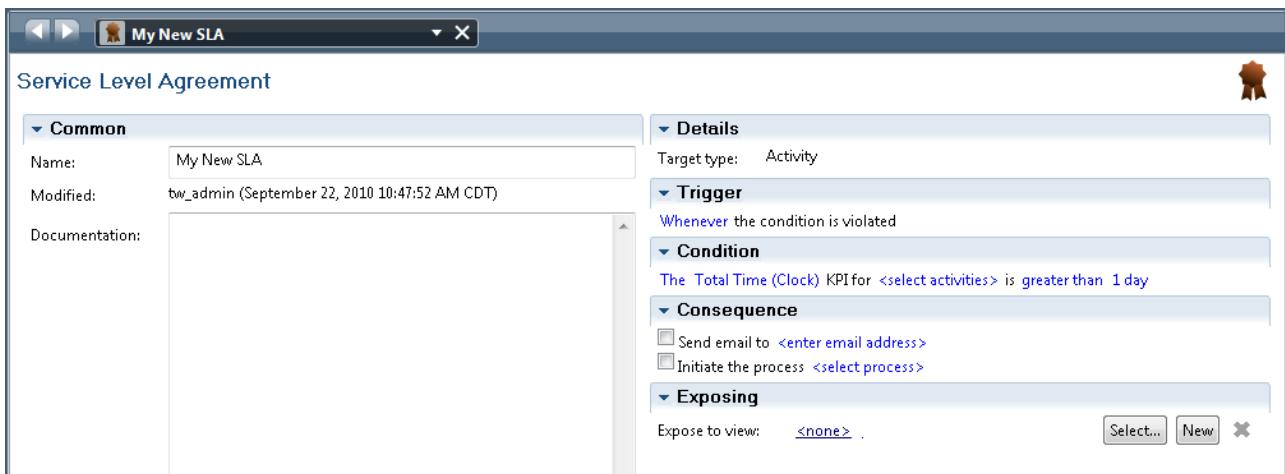
SLA definitions are created from within the Decisions category of the Library.



The wizard for the creation of a new SLA prompts us for a name of the SLA.



The SLA has its own editor into which the details of that SLA can be entered.



The trigger option can be one of the following:

- Whenever violated
- Violated A of the last B times
- Violated N times over period
- Violated % of the time over period
- Violated % more of the time period over period
- Violated % less of the time period over period

The condition is a complex combination expression

It starts with a predicate which can be:

- Single value (The ...)
- Sum of values over time (The sum of the ...)
- Average value over time (The average value of the ...)

The Consequence section defines what happens if the SLA is violated. It can be one or both of:

- Send an email to a user
- Execute a BPD process

If we define a process to be started, the process **must** be defined as having following inputs:

Input variable name	Data Type
violationRecord	SLAViolationRecord
parameters	XMLElement

This is shown in the following image of the Variables for a BPD.

Variables

<ul style="list-style-type: none">VariablesLocal<ul style="list-style-type: none">Input<ul style="list-style-type: none">violationRecord (SLAViolationRecord)<ul style="list-style-type: none">slaName (String)violationTime (Date)violationLevel (Decimal)value (Decimal)parameters (XMLElement)OutputPrivateExposed Process Variables	<ul style="list-style-type: none">Add PrivateAdd InputAdd OutputLink EpvRemoveMove UpMove Down
---	--

Dash-boarding

As process instances run in a Process Server environment, they can store *tracked performance data*. This data is stored as rows in a variety of tables in the Performance Data Warehouse database and can be used for subsequent reporting. Through Process Portal or other tools, reports can be generated that will visualize this performance information.

The performance data that is available is composed of KPIs, SLAs and automatically/explicitly recorded data.

Reporting as provided natively by IBPM is only one way to track business performance. An alternative is to utilize the power of the IBM Business Monitor product which seamlessly integrates with IBPM. See: IBM Business Monitor.

Architecture

As Business Processes run, over time we want feedback on how those processes are behaving. This feedback is generically termed a *dashboard* and usually consists of one or more charts displaying some data associated with the process.

Within IBPM, as processes execute, we can define that one or more variables in a BPD are to be recorded. This means that the historic values of these variables are stored and continue to be available past the end of the lifetime of the process. The way that this is achieved is that process data is written to one or more tables in the Performance Data Warehouse database. The data structure of these tables are publicly exposed and documented by IBM and database applications can be written to query the content of these tables to obtain data that can be used for reporting.

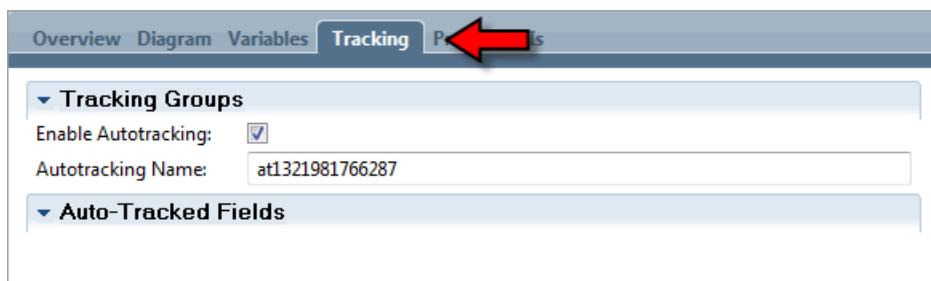
Tracking data

Generically, the data that a BPD exposes for reporting is called *tracked data*. There are a number of options provided to us to allow data to be tracked. The first of these is called *auto-tracking*.

Auto-tracking is the ability of IBPM to track performance data automatically without having to define additional data writing points explicitly within the solution. Auto-tracking is defined on a per BPD basis and it is enabled by default. With auto-tracking enabled, the start and end of *each* activity encountered in the BPD solution is written as a new table row into the Performance Data Warehouse database.

When we define variables in a BPD we can elect to define none, some or all of them as *tracked*. This means that when a BPD changes the values of these variables, these changes will be saved and later examined in reporting.

The settings for auto-tracking can be found in the Tracking definitions selected from the menu of the BPD editor.



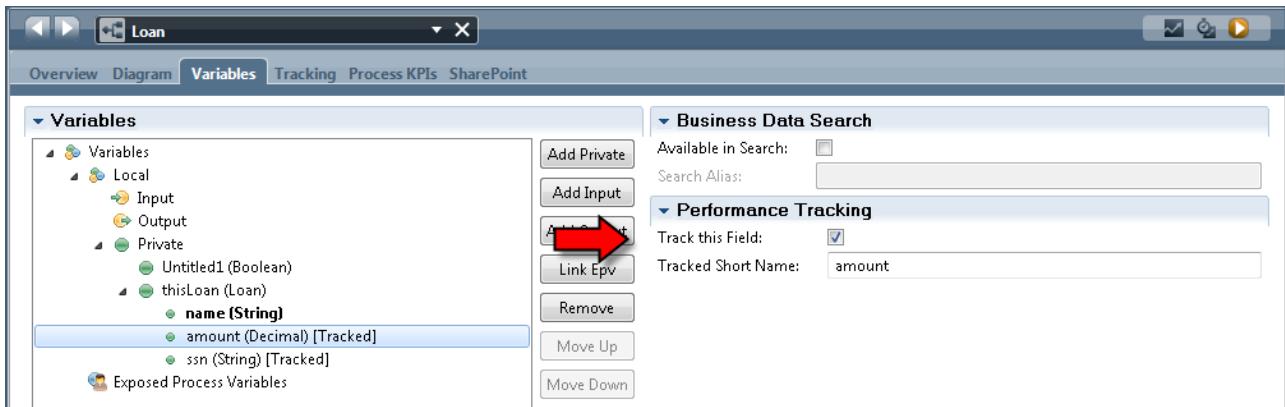
The Autotracking Name attribute value becomes the name of a database view in the Performance Data Warehouse (PDW) database. Because of this, the Autotracking Name should be chosen carefully to ensure that a table view created with this name is permitted as a legal and valid table view name by the database provider.

There is one database table created per BPD and each table name has to be unique.

Variables within the process can be flagged as *tracked*. Only variables of certain types can be tracked. These include:

Tracking Type	Allowed Data Types
Auto-tracking	String, Integer, Decimal, Boolean, Date
Tracking Groups	String, Number, Date

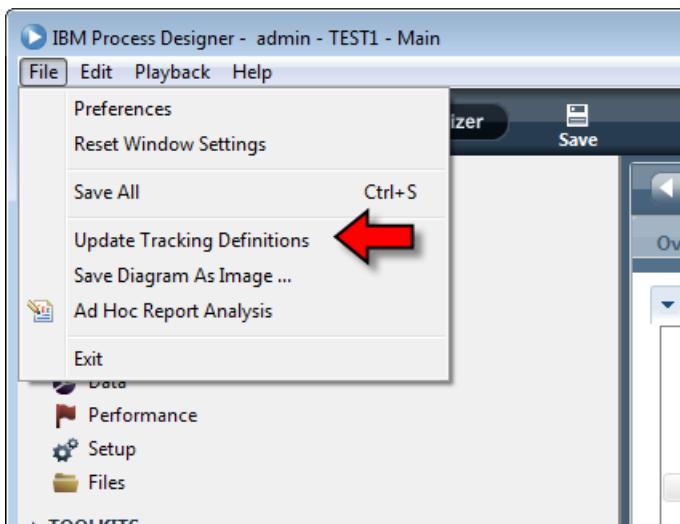
To enable tracking either check the box in the Performance Tracking section or right click on the variable in the Variables list and select Track this Field. Variables which are tracked are flagged as such in the variables list.



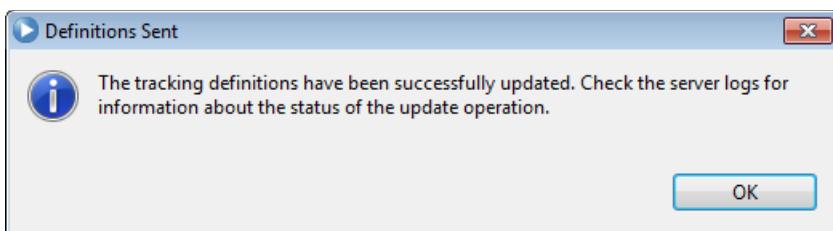
There are some additional limitations on the types of variables that can be tracked. These include:

- No support for list (array) variables
- No support for tracking a complete business object

After having made definitions about the variables to be tracked, these definitions must be "pushed" out to the Performance Data Warehouse. The File menu in IBPM PD contains a menu item called "Update Tracking Definitions". Selecting this will cause IBPM PD to contact the Performance Data Warehouse database to inform it about the new or modified tracking definitions. It is the sending of the tracking definitions to the Performance Data Warehouse that causes it to dynamically create the tables and views it needs for recording the information. This capability will only work if the database userid defined to IBPM has sufficient authority to create tables on the database.



Once the request to push information to the Performance Data Warehouse has been made, a confirmation message box will be shown.



In the event of problems with tracking data, it is a good idea to use a table viewer tool and validate that the expected tables were created and that after some runs, they contain data.

The physical database structure for an auto-tracked view is as follows:

Column name	Description
NODE_ID	What is a "NODE_ID"?
STEP	What is a "STEP_ID"?
TRACKING_GROUP_ID	
TRACKING_POINT_ID	
TASK_ID	
FUNCTIONAL_TASK_ID	
TIME_STAMP	The date/time this record was written.
SNAPSHOT	The name of the snapshot associated with this auto tracked table.
ACRONYM	The name of the Process Application acronym associated with this auto tracked table.
TIME_STAMP_YEARS	Time stamp with only "years" field varying.
TIME_STAMP_QUARTERS	Time stamp rolled up to years/months(by quarters) changing
TIME_STAMP_MONTHS	Time stamp with years/months changing
TIME_STAMP_WEEKS	Time stamp with years/months/days changing with time grouped to weeks
TIME_STAMP_DAYS	Time stamp with years/months/days changing only (hours/minutes/seconds dropped)
KPICOST	KPI value for cost

KPIEXECUTIONTIMECLOCK	KPI value for execution time
KPILABORCOST	KPI value for labor cost
KPIRESOURCECOST	KPI value for resource cost
KPIREWORK	KPI value for rework
KPITOTALTIMECLOCK	KPI value for total time
KPIVALUEADD	KPI value for value add
KPIWAITTIMECLOCK	KPI value for wait time
Process Designer named tracked variables	A column for each variable flagged as "tracked"

Let us imagine an example usage scenario. We want to see the list of activities that were executed on behalf of a process instance. We know the process instance ID (eg. 1061).

We start by executing this query:

```
SELECT FUNCTIONAL_TASK_ID FROM TASKS WHERE SYSTEM_TASK_ID = '1061'
```

This retrieves a value called the FUNCTIONAL_TASK_ID. This will act as a key into the rest of our data. Imagine this returns the value 1724. We can then query the Auto Tracking table that is associated with a BPD with:

```
SELECT * FROM AT1318439584743
WHERE FUNCTIONAL_TASK_ID=1724
```

More precisely, we are interested in the TRACKING_POINT_ID value:

```
SELECT TRACKING_POINT_ID FROM AT1318439584743
WHERE FUNCTIONAL_TASK_ID=1724
```

We can then use this as a key into the TRACKING_POINTS view to obtain the details of the steps executed.

Since the query from the TASKS table should result in a single FUNCTIONAL_TASK_ID we can combine these queries:

```
SELECT * FROM AT1318439584743
WHERE FUNCTIONAL_TASK_ID=
  (SELECT FUNCTIONAL_TASK_ID FROM TASKS WHERE SYSTEM_TASK_ID = '1061') AND
  NODE_ID IS NOT NULL
```

(The addition of the NODE_ID IS NOT NULL removes strange duplicates)

We can now get even fancier by building an INNER JOIN to perform our work:

```
SELECT * FROM AT1318439584743
INNER JOIN TRACKINGPOINTS ON
AT1318439584743.TRACKING_POINT_ID = TRACKINGPOINTS.TRACKING_POINT_ID
WHERE FUNCTIONAL_TASK_ID=
  (SELECT FUNCTIONAL_TASK_ID FROM TASKS WHERE SYSTEM_TASK_ID = '1061') AND
  NODE_ID IS NOT NULL
ORDER BY AT1318439584743.TIME_STAMP
```

Finally we get to what may be the last query which is a cleaned up version of the previous:

```
SELECT AUTOT.TIME_STAMP, TRACKINGPOINTS.NAME FROM AT1318439584743 AS AUTOT
INNER JOIN TRACKINGPOINTS ON
AUTOT.TRACKING_POINT_ID = TRACKINGPOINTS.TRACKING_POINT_ID
WHERE FUNCTIONAL_TASK_ID=
  (SELECT FUNCTIONAL_TASK_ID FROM TASKS WHERE SYSTEM_TASK_ID = '1061') AND
  NODE_ID IS NOT NULL
```

```
ORDER BY AUTOT.TIME_STAMP
```

This variant adds an alias on the Auto Tracking table and returns only the time and name of the activity executed:

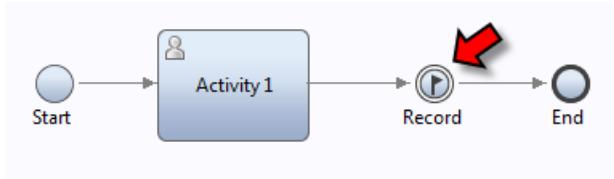
TIME_STAMP	NAME
Oct 12, 2011 1:12:11 PM 558000	Start (PRE)
Oct 12, 2011 1:12:11 PM 570000	Start (POST)
Oct 12, 2011 1:12:11 PM 659000	Task A (PRE)
Oct 12, 2011 1:12:25 PM 418000	Task A (POST)
Oct 12, 2011 1:12:25 PM 581000	Log (PRE)
Oct 12, 2011 1:12:25 PM 725000	Log (POST)
Oct 12, 2011 1:12:25 PM 857000	Task B (PRE)

See also:

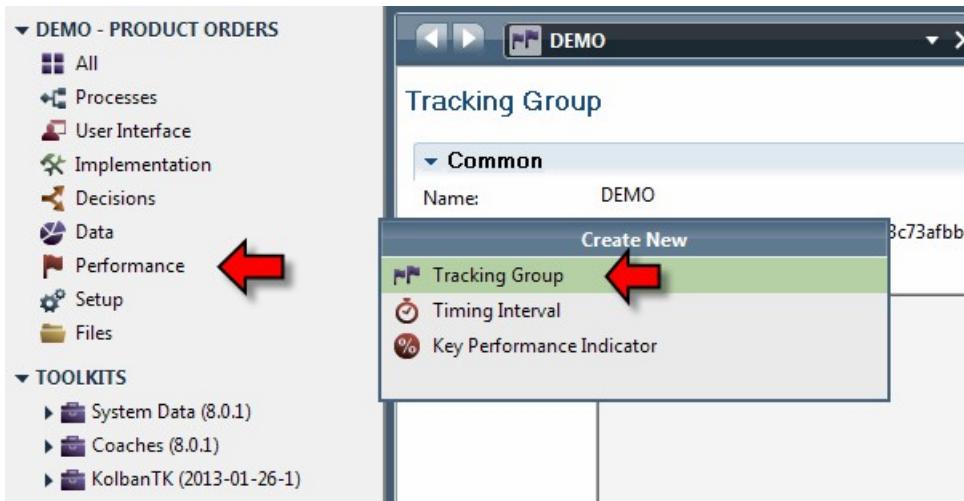
- TechNote: [The differences between the functional_task_id, task_id, system_task_id, and system_functional_task_id](#) - 2010-08-30

Tracking Groups Overview

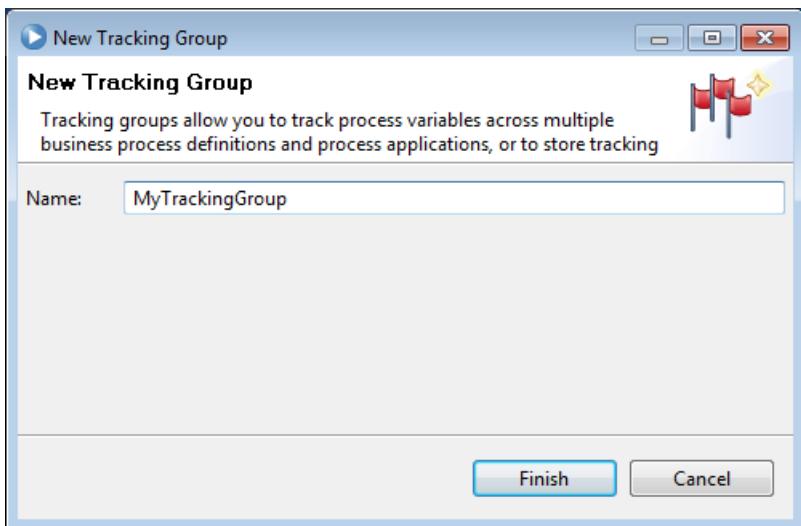
The idea behind Tracking Groups is to define a set of values that are stored in Performance Data Warehouse tables under explicit control of the process application solution. The Tracking Intermediate Event primitive is used to achieve this goal:



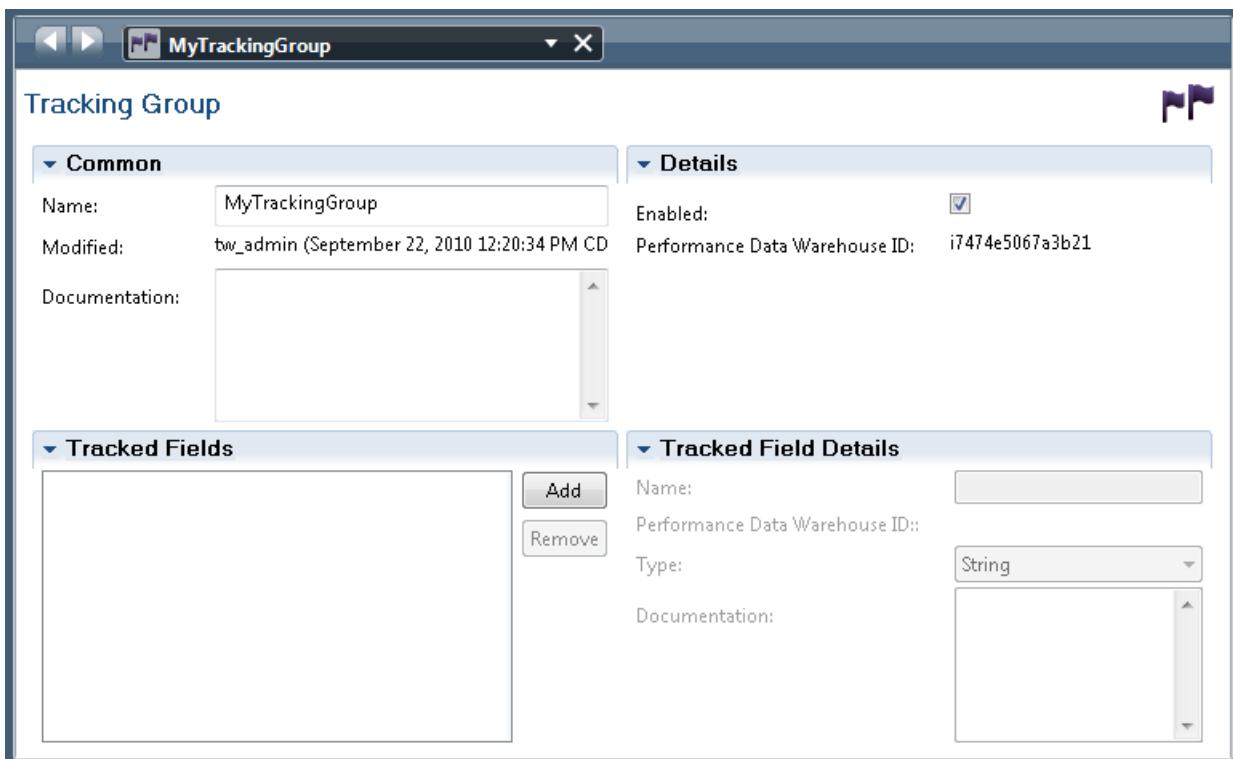
A Tracking Group can be explicitly created in Process Designer under the Performance category.



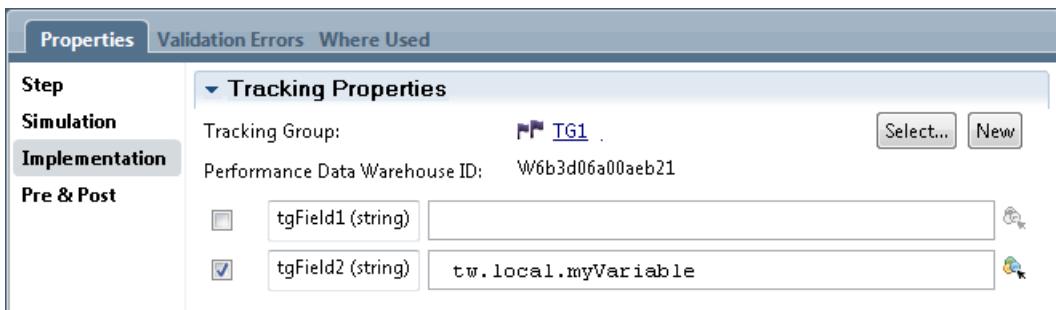
Once selected for creation, a dialog is shown which allows the name of the Tracking Group to be entered.



The parameters of a Tracking Group definition include a list of field name that are to be considered a row in the tracking data table.

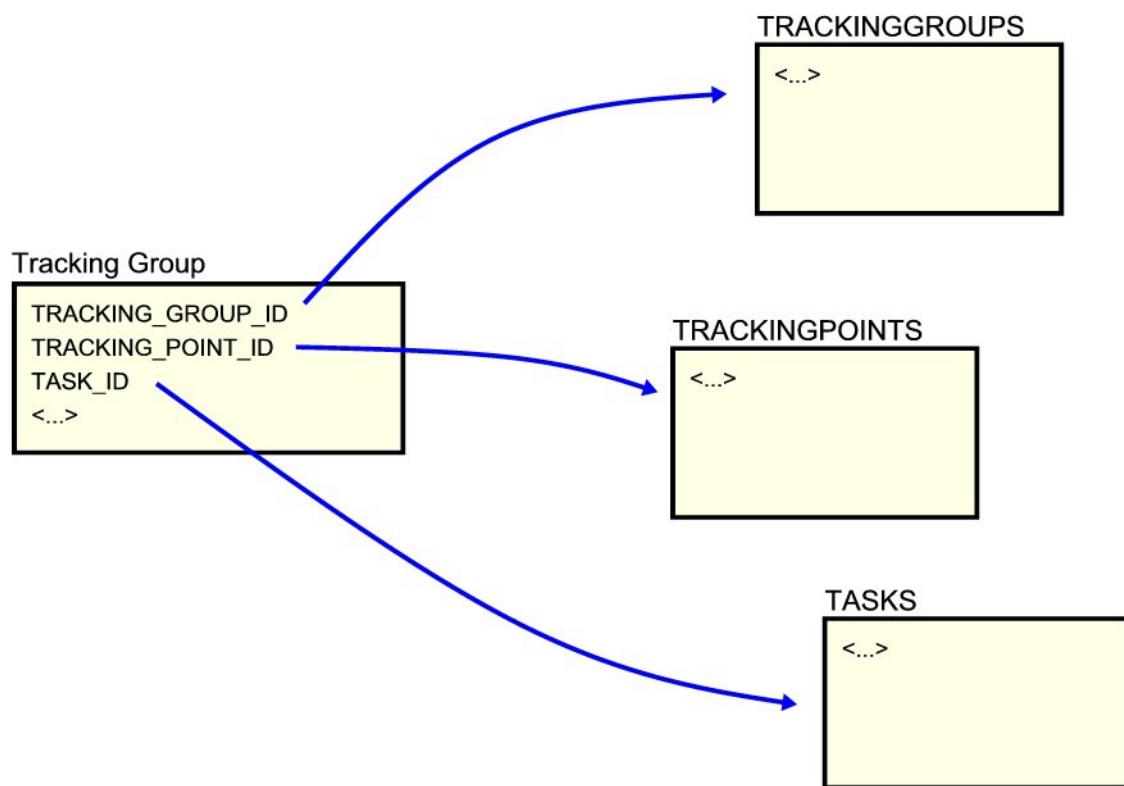


When an Intermediate Tracking Event (see: Intermediate Tracking Event) is created in a BPD, it provides a property page in its implementation where a Tracking Group can be named and the mapping between local data and the Tracking Group field names can be entered:



When a tracking point in a BPD is reached, if the field is checked, then its value is contributed to the new row otherwise there will be a NULL value for that column.

Database Structure for a Tracking Group



The primary view of interest to us is a database view given the same name as the Tracking Group definition.

Column name	Description
Columns for each of the tracked fields in the Tracking Group.	There will be a column in the View for each of the named tracked fields in the Tracking Group definition.
TRACKING_GROUP_ID	This is the primary key for the Tracking Group entry. This links to the TRACKINGGROUPS view
TRACKING_POINT_ID	This is the primary key for the Tracking Point entry. This links to the TRACKINGPOINTS view.
TASK_ID	This is the primary key into the TASKS view. Rows with the same TASK_ID were tracking items generated by the same process instance.
FUNCTIONAL_TASK_ID	
TIME_STAMP	The Time Stamp of the last update of this record.

SNAPSHOT	The snapshot of the process app that caused the record to be written.
ACRONYM	The Acronym of the Process Application
TIME_STAMP_YEARS	The time stamp rounded to the year. Month and day become 1/1.
TIME_STAMP_QUARTERS	Time stamp rounded to quarter.
TIME_STAMP_WEEKS	The Time stamp rounded to the week.
TIME_STAMP_DAYS	The Time stamp rounded to the day. The hours, minutes, seconds and milliseconds are zeroed.

Database Structure for TRACKINGGROUPS view

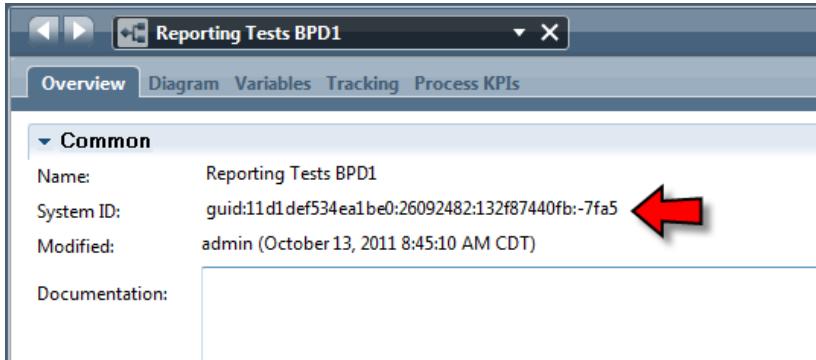
Column Name	Description
TRACKING_GROUP_ID	The identity of this tracking group
NAME	The name of this tracking group. This appears to be the PDW view name for records for this tracking group.
DESCRIPTION	A description of this tracking group. This should also be considered the name of the BPD for autotracking.
SNAPSHOT	The snapshot name (if any) of the Process Application. Apps running on the Process Center TIP appear to be NULL.
ACRONYM	The Acronym Name of the Process Application

Database Structure for TRACKINGPOINTS view

Column Name	Description
TRACKING_POINT_ID	This is the primary key for this table.
TRACKING_GROUP_ID	This is a foreign key to TRACKINGGROUPS table.
NAME	This appears to be the name of the activity that caused the data to be written.
DESCRIPTION	A description.
SNAPSHOT	The snapshot (if any) of this process application
ACRONYM	The acronym of the Process Application that caused the event to be written.

Database Structure for TASKS view

Column Name	Description
TASK_ID	Primary key for this table.
FUNCTIONAL_TASK_ID	
CREATION_TIME	The date/time at which the task was created.
START_TIME	The date/time at which the task was started. This can be null.
END_TIME	The date/time at which the task ended. This can be null.
SYSTEM_USER_ID	

USERNAME	
BPD_NAME	
STARTING_PROCESS_ID	This appears to be a GUIID of the template of a process (BPD). 
ACTIVITY_NAME	
SYSTEM_TASK_ID	
SYSTEM_FUNCTIONAL_TASK_ID	This appears to be the Process Instance ID that owns this task.
SNAPSHOT	
ACRONYM	
IS_INSTANCE	
MAX_STEP	

Database Structure for PROCESSFLOWS view

This view has a record appended to it whenever a BPD process "navigates" a sequence line in a BPD diagram. Note that **only** the link navigation flows are logged. We will see entries for the ids of the links followed.

Column Name	Description
BPD_ID	The UUID of the BPD template/definition. There is currently no known correlation to this value.
BPD_INSTANCE_ID	The BPD instance ID of this instance of the process (eg. 1061). This is the instance ID that shows up in Process Inspector and elsewhere.
LOCAL_SNAPSHOT_ID	The UUID of the snapshot containing the sequence link being traversed. Links to the SNAPSHTOS view.
SEQUENCE_FLOW_ID	The source UUID of the activity from which the sequence line came from.
SOURCE_EUID	
STEP_NUMBER	
TRACKING_GROUP_ID	
TRACKING_POINT_ID	
TASK_ID	
FUNCTIONAL_TASK_ID	

TIME_STAMP	
SNAPSHOT	
ACRONYM	
TIME_STAMP_YEARS	
TIME_STAMP_QUARTERS	
TIME_STAMP_MONTHS	
TIME_STAMP_WEEKS	
TIME_STAMP_DAYS	

See also:

- Tracking Intermediate Event
- Monitor Tracking point data
- [DeveloperWorks - Monitoring business processes by using tracking groups in IBM BPM V7.5](#) - 2011-08-10

Miscellaneous Tracking Data Notes

To disable tracking data emission at the product level, there is a setting in `100Custom.xml` that can be added/changed:

```
<properties>
  <common merge="mergeChildren">
    <monitor-event-emission>
      <enabled merge="replace">false</enabled>
    </monitor-event-emission>
  </common>
</properties>
```

Tracking data is not immediately written to the database. Instead, the data is cached at the Process Server side and then is periodically flushed as a unit to the database. The interval between flushes is 30 seconds by default. The value can be found in the `<...>/performance-data-warehouse/config/system/00Static.xml` file.

An entry called:

```
<transfer-execution-interval>30</transfer-execution-interval>
```

defines the time between successive transfers of data to the database. A second parameters called:

```
<transfer-block-size>500</transfer-block-size>
```

defines how many records should be transferred during a single cycle.

Removing tracking data tables

There is no known recipe for removing tracking data tables and views. This feels like a gross omission but it is what it is. The best recipe currently known is to "reset" the performance data warehouse which will lose all previous data.

First drop all the tables in PDW that are related to Process Center. This will usually be all the tables owned by "db2admin".

Next drop all the views.

Now we can re-execute the script to recreate the dropped tables:

Minimizing Tracking Group entries

It is tempting to create a tracking group definition with all the possible entries that you may want. However, it may be simpler to create a generic tracking group with columns for keys and columns for values and use those as markers for the data you want extracted:

The screenshot shows the Oracle Data Integrator interface with the title bar "EMORYTG". The main window is titled "Tracking Group". It has two main sections: "Common" and "Details".

Common Section:

- Name: EMORYTG
- Modified: admin (Oct 29, 2012 10:12:37 AM)
- Documentation: (Edit) - A large text area with placeholder text: "Click Edit to add or edit text."

Details Section:

- Enabled:
- Performance Data Warehouse ID: c3a73bd0e41daa31

Tracked Fields Section:

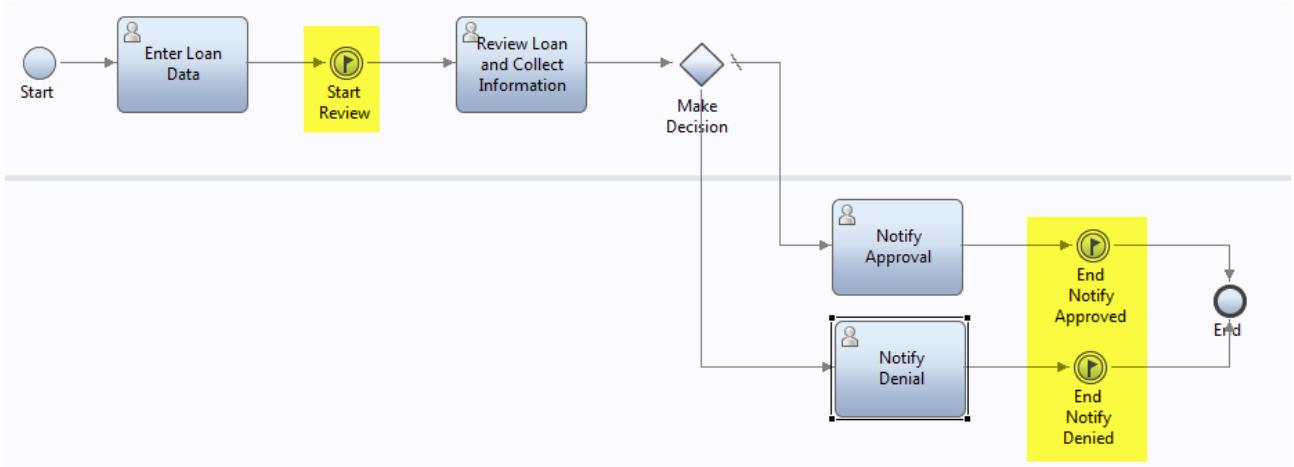
- A list of tracked fields: key1 (string), key2 (string), numKey1 (string), numKey2 (string), dateKey1 (string), dateKey2 (string), value1 (string), value2 (string), numValue1 (number), numValue2 (number), dateValue1 (datetime), dateValue2 (datetime).
- Buttons: Add, Remove.

Tracked Field Details Section:

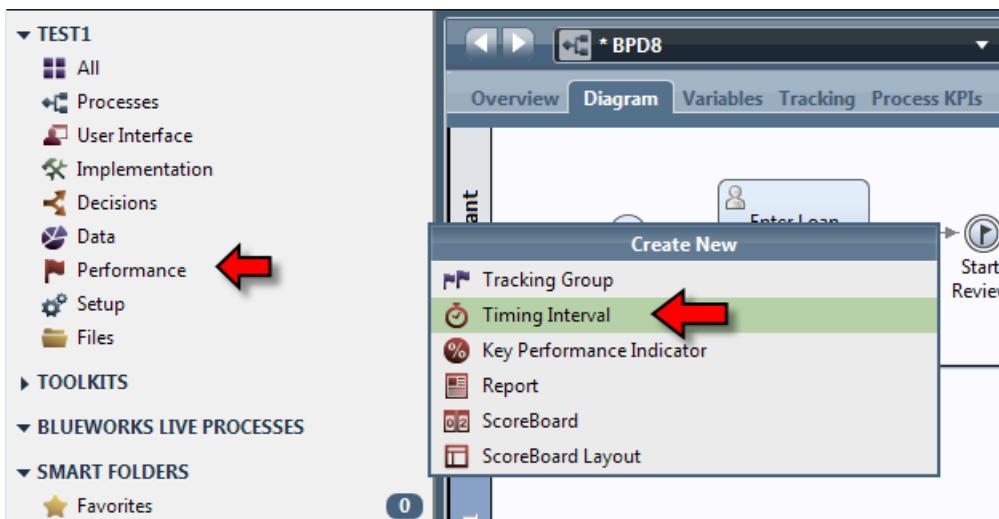
- A single tracked field entry:
 - Name: dateValue2
 - Performance Data Warehouse ID: B3a7ff1bc71daa31
 - Type: Date/Time
 - Documentation: (Edit) - A text area with placeholder text: "Click Edit to add or edit text."

Timing Intervals

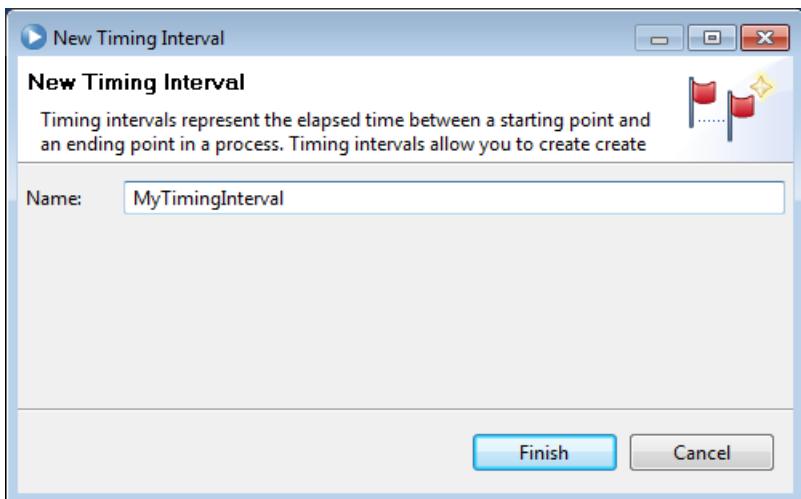
Consider the following example BPD:



Notice the three occurrences of Intermediate Tracking Events. These cause Performance Data to be generated. If we wish to determine the interval between the start of a section of the process and the conclusion of that section, we can generate a Tracking Event at the start and a corresponding tracking event at the end. Once done, we can create a Timing Interval definition to measure the elapsed time between them. The Timing Interval can be created from the Performance section:



When created, we give it a unique name.



Next we specify which tracking points will be used to flag the start if the interval and which the end.

Process/Service	Tracking Point	Tracking Group
Loan	Start Review	MyTrackingGroup

Process/Service	Tracking Point	Tracking Group
Loan	End Notify Approved	MyTrackingGroup
Loan	End Notify Denied	MyTrackingGroup

At run-time, we now need to understand how this information exposes itself. The first thing to understand is a database view called `TIMINGINTERVALS`. This table has the following structure:

Column name	Data type	Description
<code>TIMING_INTERVAL_ID</code>	Decimal	A unique id for the timing interval itself
<code>NAME</code>	Varchar	The name of the timing interval
<code>DESCRIPTION</code>	Clob	A description

For example, if we create a Timing Interval definition called "MyTimingInterval" then a search of the `TIMINGINTERVALS` view with "`NAME='MyTimingInterval'`" will take us to a `TIMING_INTERVAL_ID` value. There is a one-to-one relationship between `TIMING_INTERVAL_ID`s and their associated `NAME` values.

The next table we are interested in is called TIMINGINTERVALVALUE

Column Name	Data type	Description
TIMING_INTERVAL_ID	Decimal	The timing interval ID. This is a unique value that is system generated for each type of Timing Interval. See the TIMINGINTERVALS view for a mapping from a name to the TIMING_INTERVAL_ID value.
START_TRACKING_POINT_ID	Decimal	
END_TRACKING_POINT_ID	Decimal	
START_TIME	Timestamp	The start time of the interval
END_TIME	Timestamp	The end time of the interval
DURATION	Decimal	The duration of the interval in milliseconds
START_TASK_ID	Decimal	
END_TASK_ID	Decimal	
FUNCTIONAL_TASK_ID	Decimal	
START_SNAPSHOT	Varchar	
END_SNAPSHOT	Varchar	

Be careful when creating names of Timing Intervals. If we have two process apps both of which have the same named Timing Interval, it will be difficult to tell one from another because the "NAME" column in the TIMINGINTERVALS table will not be unique.

Performance Data Warehouse (PDW) SQL Snippets

There are common patterns that keep appearing when reports are asked for. Here we will discuss and describe these patterns as well as describing some simple solutions for incorporating these into the reports.

When using the built-in SQL processing services provided by the product, the JNDI entry for the PDW DB is "jdbc/PerformanceDB".

See also:

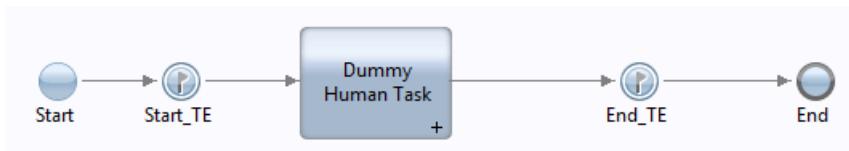
- Database Integration

How many have started but not finished (in progress)

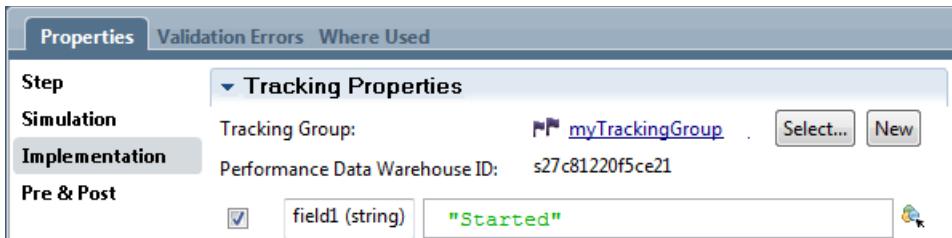
Consider a process which has work to be performed and in the process the work can start and at some time later be completed. A common question is how many process parts have started but not yet completed?

One way to achieve this is to create a field in a tracking group used to track this information. Before the work item is started, set the field to "START" and write a record to the PDW. When work completes, set the field to "END" and write a second record to the PDW. The number of items thus started but not yet finished is the number of rows that have the value "START" minus the number of rows that have the value "END".

Here is a sample BPD that shows us writing two events.



The Start_TE event looks like:



while the End_TE event looks like:



Querying the view for the tracking group will now result in a column called "FIELD1" which has either "Started" or "Ended" values within it.

Now let us focus on the query.

We want to find the count of "Started" entries and subtract from this the number of "Ended" entries. A suitable query for this would be:

```
SELECT count(*) - (select count(*) FROM MYTRACKINGGROUP where field1='Ended') as
inFlight from MYTRACKINGGROUP where field1='Started'
```

As you may see, if you can't speak SQL, you may be in trouble.

Number of items of different types/day

Consider a company that sells red, green and blue widgets. During the course of a period of time (say a week), they sell a certain amount of widgets in total on each day. We can imagine a bar chart with an X-Axis of days and a Y-Axis of widgets sold. To add more detail, we want to break the Y Axis down into counts of how many were red, how many green and how many blue. How can we do this?

One way is to expose a variable in a tracking group. The variable's value will either be "Red", "Green" or "Blue". A report data source can then be.

Sales Rep	Color
A	Red
B	Green
A	Green
A	Red
B	Green

A query then may be:

Sales Rep	#Red	#Green
A	2	1
B	0	2

A suitable SQL query is:

```
SELECT minute(time_stamp), (select count(*) FROM test2 as t2
    WHERE color='green' AND minute(t1.time_stamp) = minute(t2.time_stamp)) as green,
(select count(*) from test2 as t2 where color='red' and minute(t1.time_stamp) =
minute(t2.time_stamp)) as red
from TEST2 as t1
group by minute(time_stamp)
```

Getting the last state of a process

As a process instance executes, it passes from one state (activity) to another. A common request is to be able to see which processes are in which states. As the process runs, we can imagine it generating Tracking Events as it enters a new state. This means that for a given process instance we will have a growing list of records of the general format:

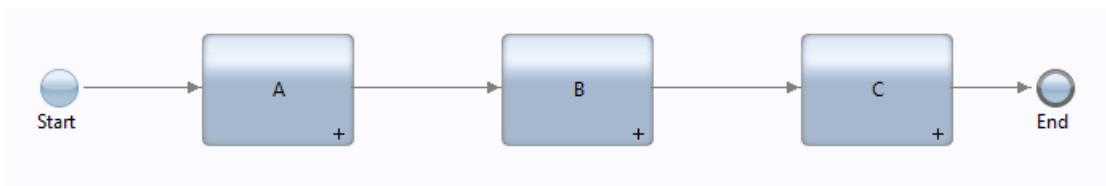
PIID	Time	State
P1	T1	State A
P1	T2	State B
P1	T3	State C
...
P1	T _{end}	State End

So if we want to find the current state of the process, what we want to do is find the most recent record written for a given process.

```
SELECT * FROM TABLE ORDER BY TIME DESC FETCH FIRST 1 ROW ONLY
```

Breakdown of how many in which step

Consider the following illustrative process which has three steps.



Now imagine that there are many instances of this process running. For simplicity, let us say that there are 100 instances running. Of these instances, some number of them will be in step A, some in step B and some in step C. Now imagine that we want to report on where are we as a business? We could imagine producing a report that might say:

My Process

Step A	25.00%
Step B	40.00%
Step C	35.00%

From this, and at a glance, we can now tell a lot more about the operation of our business. So how can we build such a thing?

Imagine now a Tracking Group called "APPSTATETG". This has a column/field called "appState" which will hold the state of a process.



```

SELECT task_id as t1,MAX(time_stamp) as t2 FROM DB2ADMIN."APPSTATETG"
GROUP BY task_id
SELECT appstate,count(appstate) from DB2ADMIN.APPSTATETG
WHERE time_stamp in
(SELECT max(time_stamp) FROM DB2ADMIN."APPSTATETG" group by task_id)
group by appstate

```

Last Autotracked row

The following will return data for the LAST autotracked rows:

```

select * from <AutoTrackTable> as T1, (select TASK_ID, MAX(TIME_STAMP) AS TMAX from <AutoTrackTable>
GROUP BY TASK_ID) as T2 where T1.TASK_ID = T2.TASK_ID and T1.TIME_STAMP = T2.TMAX and T1.KPICOST is
not null

```

The last predicate in the query appears to be necessary because Task Ids and Timestamps don't appear to be unique.

Counts of processes started vs completed over an interval

Consider a process which writes a start event when it begins and an end event when it completes. What we wish to build is a list of how many process instances started and ended within a given time interval. When the process starts the "datekey1" TG field is written with 'startProcess'. When the process completes, the "datekey1" TG field is written with 'endProcess'. The following SQL will illustrate this concept:

```

-- Return a table of 3 columns. The columns are:
-- o MINUTE      - The interval over which the measurement is being calculated
-- o STARTCNT    - The number of process instances started in this interval
-- o ENDCNT      - The number of process instances ending in this interval
select minute(time_stamp) as minute,
       (select count(*) from TGTABLE as t2 where datekey1 = 'startProcess' and minute(t1.time_stamp) =
minute(t2.time_stamp)) as startcnt,
       (select count(*) from TGTABLE as t2 where datekey1 = 'endProcess'     and minute(t1.time_stamp) =
minute(t2.time_stamp)) as endcnt
  from TGTABLE as t1 group by minute(t1.time_stamp)

```

General DB/SQL useful functions for reports

Date formatting

A timestamp can be formatted into a string with the `varchar_format()` function. This takes both a timestamp and a format string and the result is a string complying to the format.

Mapping BPM date types to DB TIMESTAMP types

In the tables generated for PDW, there are many places where the TIMESTAMP DB data type is used. Within BPM, we use a Date data type. There are times when we want to build a SQL query string and what we have is a BPM Date object and what we need is a DB TIMESTAMP object. Here is the recipe to map from one to the other.

In a SQL String, we can create a TIMESTAMP object using:

```
TIMESTAMP('yyyy-mm-dd hh:mm:ss')
```

so what we need to do is take a BPM Date object and map that to a compatible string. Fortunately, the BPM Date object has a method called "format" that will format the data for us.

```
var mySQL = "select X from MYTABLE where" +
    "TIME_STAMP = TIMESTAMP('" + tw.local.myDate.format("yyyy-MM-dd HH:mm:ss") + "')"
```

See also:

- Data Type – TWDate

Number of items in the current week

```
select count(*)   from tg1 where week(TIME_STAMP) = week(current date)
```

Counting when a column equals a value

Consider the idea of wanting to count columns where a column value is A or B. What we want is something like:

```
select count(number of columnC = A values), count(number of columnC = B values) from myTable;
```

Unfortunately, this simply won't work. The trick for this is:

```
select sum(case columnC when 'A' then 1 else 0 end),
       sum(case columnC when 'B' then 1 else 0 end) from myTable;
```

Selecting records within a date range

Consider the notion of selecting records that fall within a particular day. If we have a Tracking Group table, then it contains a column called "TIME_STAMP". To select rows in a particular day, the following can be used:

```
SELECT * FROM "DB2ADMIN"."DEMOTG1" where TIME_STAMP between TIMESTAMP('2013-12-26 00:00:00') and
TIMESTAMP('2013-12-26 23:59:59')
```

See also:

- developerWorks – [DB2 Basics: Fun with Dates and Times](#) - 2003-08-28

Reporting with Microsoft Excel

The Microsoft Excel spreadsheet package can perform powerful charting functions and if we can import the data to be charted, we can then graph from there.

Developing Custom Dash-boards

As we have seen there are a number of techniques for recording process data into the PDW tables. Here we now start to think about thinking about how to design dash-boards including data collection, SQL queries and visualization.

Let us start by thinking about what the dashboard is to contain. Think of it as a screen which should show business information to business users. The information to be shown should be requested by

the end-users of that information. It is usually wrong for the process designer to make up metrics for display without consulting with the eventual consumers. It is those consumers who will have the best notion of what it is they wish to see. Once we have some understanding of what is desired, we will need to ensure that we have a plan to be able to obtain the data for those metrics that can be used to show the results. It is no use to hear that we want to show "average salary increase by department" when no-where in the process do we deal with the salary.

Once we have seen that the process is able to generate data to achieve the metrics, we must ensure that the data is actually generated. This will usually mean the injection of new tracking points into the solution.

Next we should experiment with building the SQL statements necessary to retrieve the data. It is strongly recommended to build such queries using tools such as IBM Data Studio.

With the SQL statements built, now we can turn our attention to building out the Human Service that will execute that SQL.

With the data now in IBM BPM, we can populate charts and graphs with the data retrieved.

At a high level, the following steps are recommended:

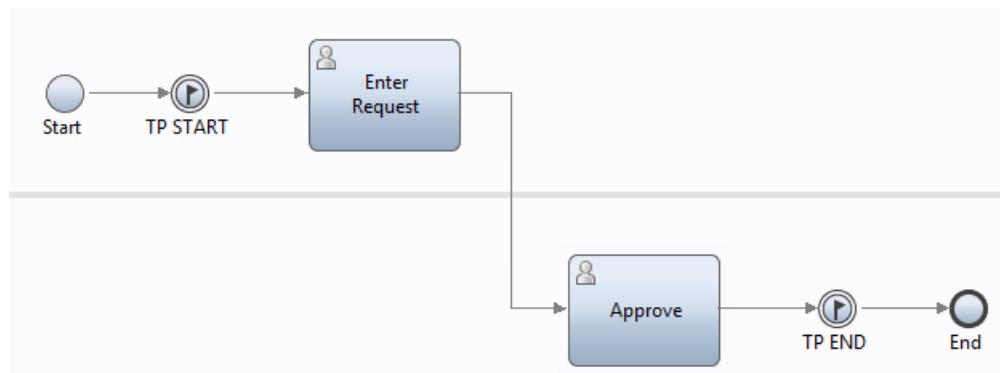
1. Clearly capture the specification of the metric to be shown. This will include discussions with eventual consumers and napkin drawings of what the metric may look like.
2. Cause the BPM process to generate the data necessary for the metric.
3. Write SQL statements to parse the data and retrieve the data.
4. Build BPM services to make the SQL calls to retrieve the data.
5. Show the data in charts and tables.

A Sample Dashboard

Consider the story of a procurement process. In the process a user can request to buy hardware or software, specifies the price, the department to be charged and a description. Next a manager approves or declines the purchase. A simple process.

Now let us consider the metrics that may be calculable:

- The number of procurement requests per month
- The number of procurements approved or declined per month
- The amount of money spent on procurement per month



The tracking group is defined as:

```
department (string)
place (string)
itemType (string)
price (number)
```

Which results in a table that looks like:

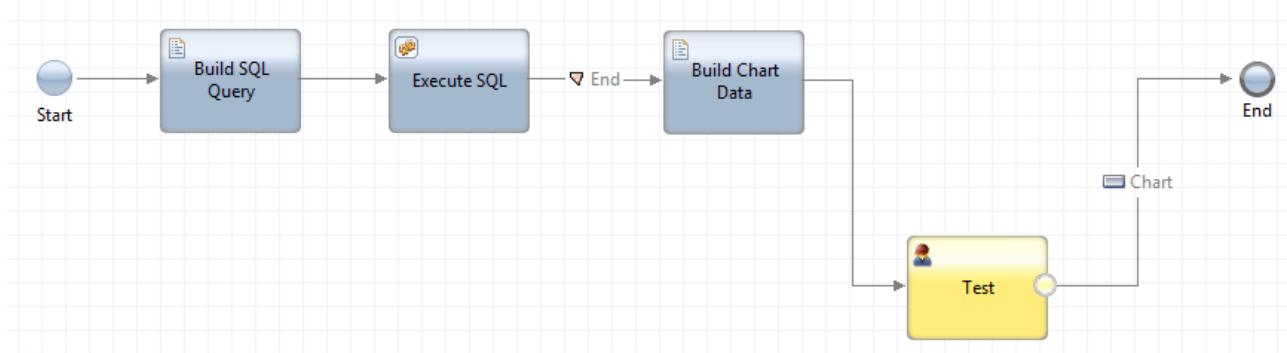
PROCURETG	
Columns	
DEPARTMENT	[VARCHAR(256) Nullable]
ITEMTYPE	[VARCHAR(256) Nullable]
PLACE	[VARCHAR(256) Nullable]
PRICE	[DECIMAL(31, 6) Nullable]
TRACKING_GROUP_ID	[DECIMAL(31, 0)]
TRACKING_POINT_ID	[DECIMAL(31, 0)]
TASK_ID	[DECIMAL(31, 0)]
FUNCTIONAL_TASK_ID	[DECIMAL(31, 0)]
TIME_STAMP	[TIMESTAMP Nullable]
SNAPSHOT	[VARCHAR(256) Nullable]
ACRONYM	[VARCHAR(256) Nullable]
TIME_STAMP_YEARS	[DATE Nullable]
TIME_STAMP_QUARTERS	[DATE Nullable]
TIME_STAMP_MONTHS	[DATE Nullable]
TIME_STAMP_WEEKS	[DATE Nullable]
TIME_STAMP_DAYS	[DATE Nullable]

Let us now take out first query. This is "The number of procurement requests per month". Suitable SQL for this would be:

```
SELECT COUNT(*) AS COUNT, MONTH(TIME_STAMP_MONTHS) AS MONTH
  FROM DB2ADMIN.PROCURETG
 WHERE PLACE = 'END'
 GROUP BY TIME_STAMP_MONTHS
```

Now, let us build a service to work with the data. The way I like to work is to build Human Service which, when executed will return a ChartData object instance (see: Chart Control). This is the data type used by a Chart control to draw a chart instance. While I am building the service, I insert a Coach in the flow simply to show a chart while building and testing. Once I am happy with the service, I skip the Coach.

The Service I build has three primary parts. See the following example:



Part 1 called "Build SQL Query" defines the SQL statement to be executed.

Part 2 called "Execute SQL" actually executes the SQL statement to retrieve the data from the PDW.

Part 3 called "Build Chart Data" takes the results from the SQL query and builds an instance of the ChartData object.

For Part 1, the SQL query is as shown above and assigned to a local private variable called "sql".

For Part 2, we use an Execute SQL nested service call defined as follows:

The screenshot shows the Service Designer interface with the 'Data Mapping' tab selected. Under 'Input Mapping', there are five entries: 'tw.local.SQL' (linked to 'sql (String)'), 'parameters(List of S...)' (linked to 'parameters'), 'maxRows (Integer)' (linked to 'maxRows'), 'returnType (String)' (linked to 'returnType'), and 'dataSourceName (String)' (linked to 'dataSourceName'). Under 'Output Mapping', there is one entry: 'tw.local.Results' (linked to 'results (ANY)').

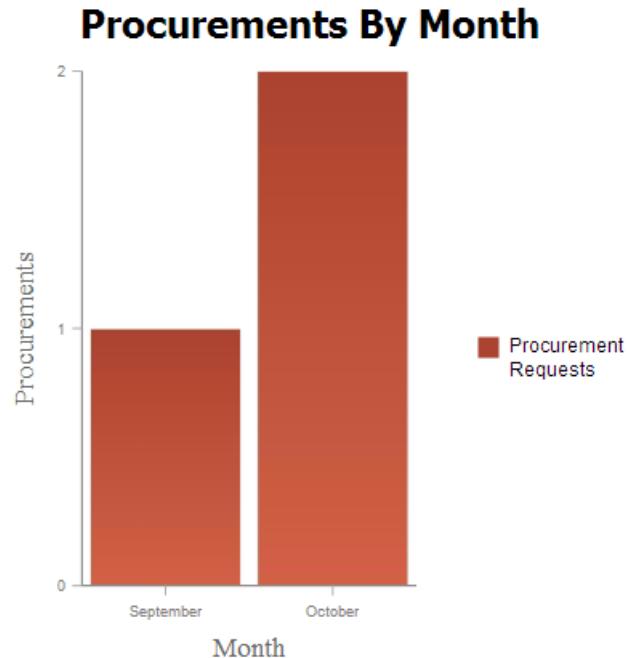
Note especially the use of the "Record" data type to hold the results of the query.

Finally for Part 3, here is the populate of the ChartData object:

```
tw.local.chartData = {
    plots: [
        {
            series: [
                {
                    label: "Procurement Requests",
                    data: []
                }
            ]
        }
    ];
};

var chartData = [];
for (var i=0; i<tw.local.Results.listLength; i++) {
    chartData[i]={
        "name": "" + tw.local.Results[i].MONTH,
        "value": tw.local.Results[i].COUNT
    };
}
tw.local.chartData.plots[0].series[0].data = chartData;
```

And finally, the results of this work would be:



To be able to build your own monitor charts, I maintain that one needs to be able to read the above and fully understand all the parts involved. The reality is that it isn't as complex as it may at first appear ... however ... it does initially appear somewhat bewildering with lots of "parts". Take the

time to understand each part by itself and then, only when ready, see how they integrate together to achieve the desired overall goal.

Generating sample data

When building out custom reports, we may wish to create sample data. Unfortunately, running processes to create the data, although ideal, takes too long so what we want is a way to have the data created for us without having to run all those processes. When we realize that when a process is run, its history is saved within the PDW, we can work towards injecting rows into the PDW to build our data for the system. Again, unfortunately, it isn't as easy as that. We run into the danger of corrupting our PDW database. In addition, many of the tables have foreign key relationships to other tables which results in a great deal of confusion on simply inserting rows.

Perhaps the easiest way is to cheat. What we can do is create a second table with the same columns used in our queries. When we build our queries, we can then use a SQL "UNION ALL" statement to select from the real PDW table as well as the test table.

Here is an example. In the following we see a Tracking Group table called "DEMOTG1".

DEMOTG1	
	Columns
▷	BRANCH [VARCHAR(256) Nullable]
▷	PROCESSSTEP [VARCHAR(256) Nullable]
▷	STATETAG [VARCHAR(256) Nullable]
▷	TRACKING_GROUP_ID [DECIMAL(31 , 0)]
▷	TRACKING_POINT_ID [DECIMAL(31 , 0)]
▷	TASK_ID [DECIMAL(31 , 0)]
▷	FUNCTIONAL_TASK_ID [DECIMAL(31 , 0)]
▷	TIME_STAMP [TIMESTAMP Nullable]
▷	SNAPSHOT [VARCHAR(256) Nullable]
▷	ACRONYM [VARCHAR(256) Nullable]
▷	TIME_STAMP_YEARS [DATE Nullable]
▷	TIME_STAMP_QUARTERS [DATE Nullable]
▷	TIME_STAMP_MONTHS [DATE Nullable]
▷	TIME_STAMP_WEEKS [DATE Nullable]
▷	TIME_STAMP_DAYS [DATE Nullable]

Within this table, in our example, we care about a small set of columns:

- BRANCH
- PROCESSSTEP
- STATETAG
- TIME_STAMP

We can now create a new table called "TESTDATA" that looks like:

TESTDATA	
	Columns
▷	Column Masks
▷	Columns
▷	BRANCH [VARCHAR(256) Nullable]
▷	PROCESSSTEP [VARCHAR(256) Nullable]
▷	STATETAG [VARCHAR(256) Nullable]
▷	TIME_STAMP [TIMESTAMP Nullable]

See that the columns that we care about have the same names.

We can now think about how to populate the new "TESTDATA" table. We have a number of options including scripts and programs but there are some useful tools available on the Internet including:

- <http://www.generatedata.com/>
- <http://databene.org/databene-benerator.html>

These can generate comma separated value text files that can then be loaded into the database.

When we now wish to write our SQL query, instead of coding:

```
select BRANCH, PROCESSSTEP, STATETAG, TIME_STAMP from DEMOTG1
```

we can code:

```
select BRANCH, PROCESSSTEP, STATETAG, TIME_STAMP from
  (select BRANCH, PROCESSSTEP, STATETAG, TIME_STAMP from DEMOTG1
   union all select BRANCH, PROCESSSTEP, STATETAG, TIME_STAMP from TESTDATA)
```

Simulation and Optimization

As processes are designed and built but before they are deployed for real use, it is desirable to know ahead of time how they might perform in reality. To help achieve this, IBPM provides a simulation capability built into the IBPM Process Designer which is called the **Optimizer**. The Optimizer can be used to examine how a process may perform by using simulated executions with hypothetical data. In addition, the Optimizer can be used with actual data collected over time from runs of the processes in production. Using this data, the Optimizer can visualize how the processes actually ran and provide visual indications of which areas may be good candidates for further improvement.

Drilling down further into the notion of simulation, we see that it is the *fictitious* execution of multiple processes. The IBM product pretends (internally) to execute the steps in the process and calculates timings and other numerical data. After the simulation has ended, the results are then shown to the user in graphs, charts and tables.

A key notion of a simulation is that it pretend to run *faster* than it would in reality. For example, if a step in a process would take an hour for a person to complete, simulating that step should not take an hour ... rather, the product's simulator should internally track it as though it had taken an hour.

See also:

- Vimeo – [Technical Demonstration – Using Simulations](#) - 2012-03-14

Defining simulation values

Think of each step in the process as having associated simulation attributes that the designer of the process explicitly provides. These attributes declare how long it may take to execute.

If we had a simple process that looks as follows:



We see that it is composed of three steps ... Step 1, Step 2 and Step 3. If we define that each step in the process will take 10 minutes to execute then when we run a simulation, we find that the average duration of a process is 30 minutes.

We can define simulation attributes of a BPD step by selecting the simulation tab in the Properties view:

Properties Validation Errors Where Used

- General
- Simulation**  **Simulation Profile**
- Implementation
- Routing
- Data Mapping
- Pre & Post
- KPIs

Saying that a step in the process takes a fixed amount of time may not be that useful so we can define that a step can take a dynamic/variable amount of time based on three available distribution types:

Fixed distribution type

Execution Time

Distribution Type: Fixed

Value	0 Days	0 Hours	30 Minutes
-------	--------	---------	------------

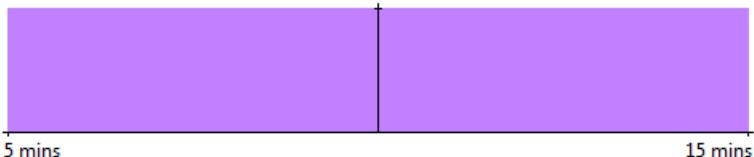
The fixed distribution time is the simplest of all. Its value is a constant (or fixed). For a wait or duration time, this means that the wait or duration will **always** be the constant value supplied.

Uniform distribution type

Execution Time

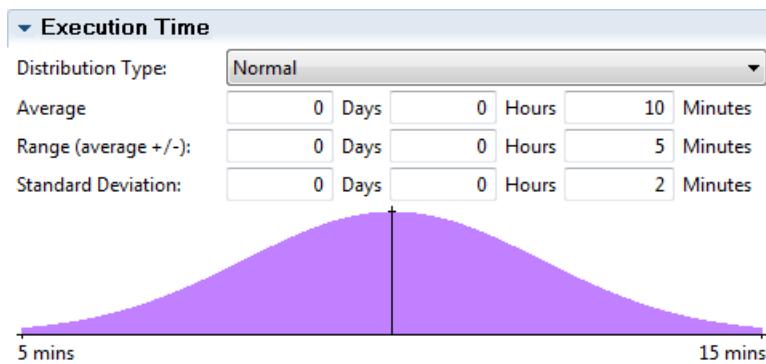
Distribution Type: Uniform

Average	0 Days	0 Hours	10 Minutes
Range (average +/-):	0 Days	0 Hours	5 Minutes



The Uniform distribution type says that its value will between a lower and an upper bound but the probability of any given value is evenly distributed across the range. For example, if the lower bound is 5 minutes and the upper bound is 15 minutes, then any individual value will be equally likely between these ranges. The probability of the value being 5 minutes is the same as that of being 10 minutes or that of being 15 minutes.

Normal distribution type

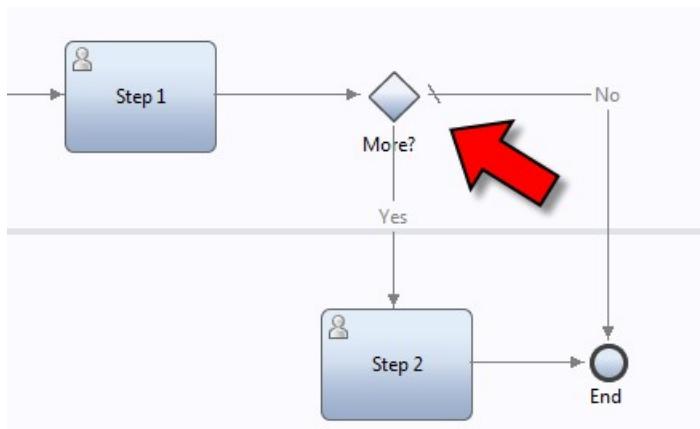


This is the most complex of the types of distribution. The distribution is in the form of a bell curve with the center of the bell curve set to be the Average value. The slope of the curve is based upon the Standard Deviation value. The Range is an amount of time that is the threshold above and below the average and marks the possible start and ends of the time interval. For example, an average of 10 with a range of 5 will produce a series of values between $10-5=5$ and $10+5=15$. The distribution of these values is governed by the standard deviation. If the range should be negative (for example, average=10, range=20 which gives a lower bound of -10) this obviously has no meaning in simulation so any distribution values less than zero are also ignored.

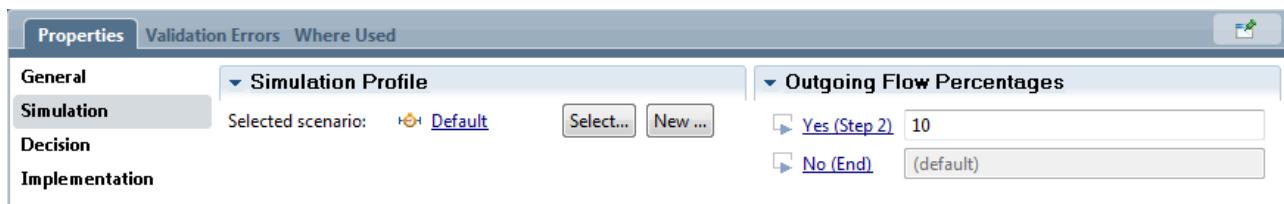
The value chosen will be between the lower and upper range but the probability of any specific value will not be a constant across the range. The probability will be as described by the curve.

Gateways and simulation

When a gateway is added into a process:



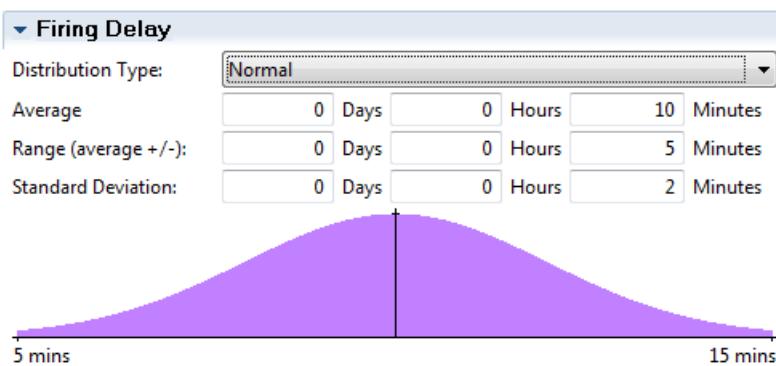
the simulation settings for such an activity allow us to define a percentage of time one path or another is followed. A percentage is simply another expression for a probability. A value of 10% Is the same as saying 1-in-10. It is important to realize that during simulation, application data for a process is completely ignored and, as such, gateway paths are followed based on statistical likely-hood and not on actual evaluated conditions. It is the choice of the designer to choose the statistical distribution to be assigned to each of the possible paths.



Arrival rate of simulation items

When a simulation is executed, imagine we wish to process 100 requests and then end the simulation. One of the first questions that comes to mind is "how fast" do we want these requests to appear to arrive? If we are running a simulation, it doesn't seem to make much sense that they should all arrive simultaneously as that would not happen in reality.

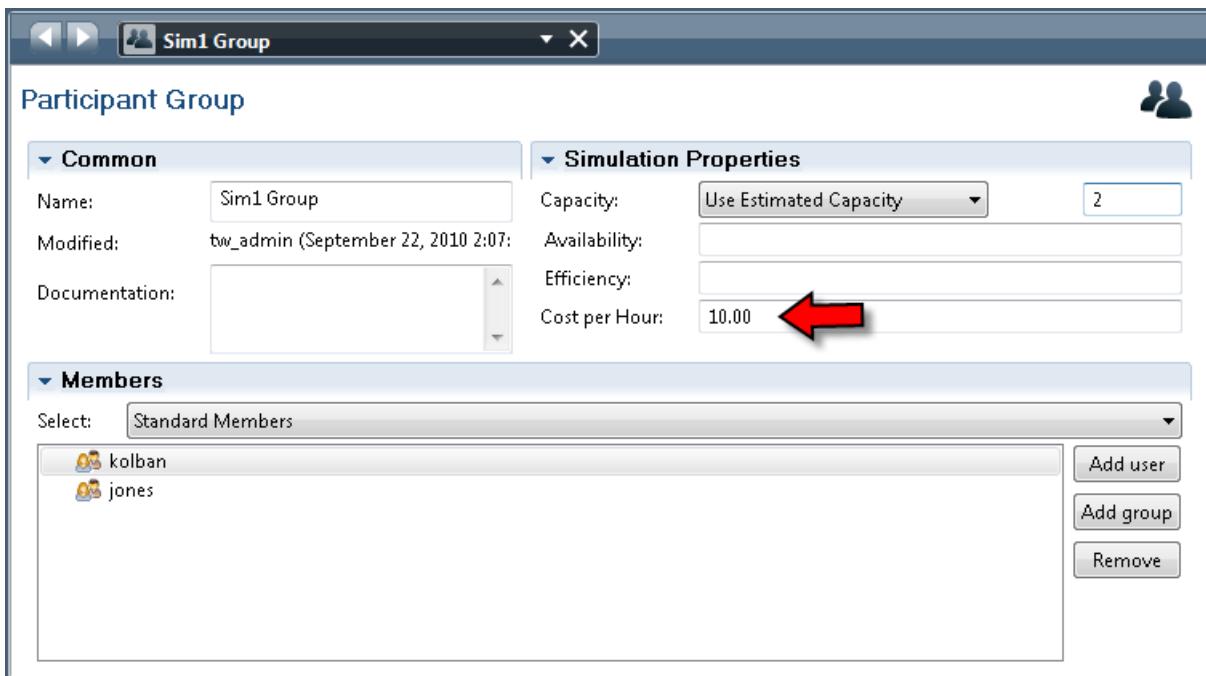
The Start Event BPMN symbol of a process can be used to describe the arrival rate of the simulated processes. This has an attribute called the "Firing Delay". If we defined this as "Fixed" with a value of 30 minutes, we would effectively be simulating 100 processes being started in total with each process arriving every 30 minutes. Of course, we could use other distribution values to represent alternate arrival distributions as opposed to a constant process arriving each unit of time.



Simulating the cost of execution

Since Human Tasks are performed by staff members and each staff member has a salary, there has to be a cost associated with executing a human task. For example, if a person is paid \$20/hour and it takes them 30 minutes to perform a task, then the cost of performing that particular task would be \$10.

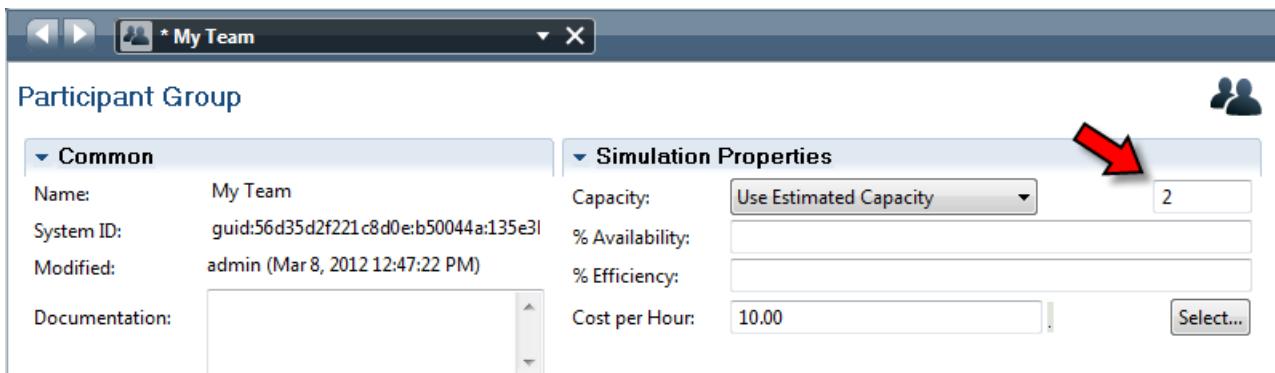
In IBPM, we associate an activity with a Participant Group which defines the staff members that are eligible to perform that task. One of the configurable attributes of a Participant Group is the idea of a cost per hour. This attribute defines the cost of using a member of that group for an hour long period.



Since we can now model the expected duration of an activity and have also defined the staff members for that activity, we can calculate the cost of executing that activity. This can be reflected in the simulation results.

Available staff members for tasks

For simulation purposes, we can also define the capacity of a Participation Group.



This is the number of staff members that are contained within that group. This comes into play when there is more work to be done than there are people to perform it. Here is an example.

Imagine it takes 10 minutes to review a work request. Imagine that there 5 members in the team that can review work requests. If 5 requests arrive then all 5 members of the team may be busy reviewing a request. What happens if a 6th request arrives while the previous 5 are still being worked? The answer is simple, the 6th request will have to wait until one of the preceding requests completes at which time releasing the staff member from their previous task to work on the new task.

The Capacity of the Participation Group defines the number of staff in the group. It can be set to an estimated number within the simulation settings of the group or else it can be set to the exact names/number of the users that are actually defined to the group. Using an estimated capacity allows us to execute "what if" scenarios such as increasing or decreasing the number of staff

members in the group.

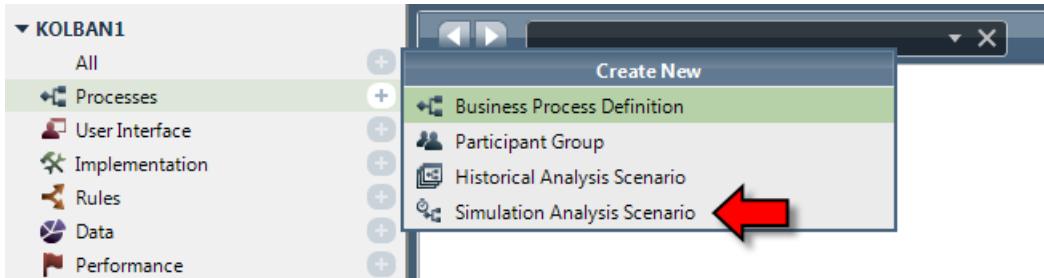
A worker in a group is unlikely to be able to devote 100% of their day simply to working on IBPM tasks. They will have other responsibilities such as team meetings, education, management reports and more. The Availability setting of the Participant Group defines a percentage of their working hours that can be spent working on IBPM originated tasks.

We are human and can't work at 100% efficiency. We take bathroom breaks, chat with colleagues and go for coffee. To allow for this in the simulations, we can define an attribute called Efficiency which is again expressed as a percentage of available working time.

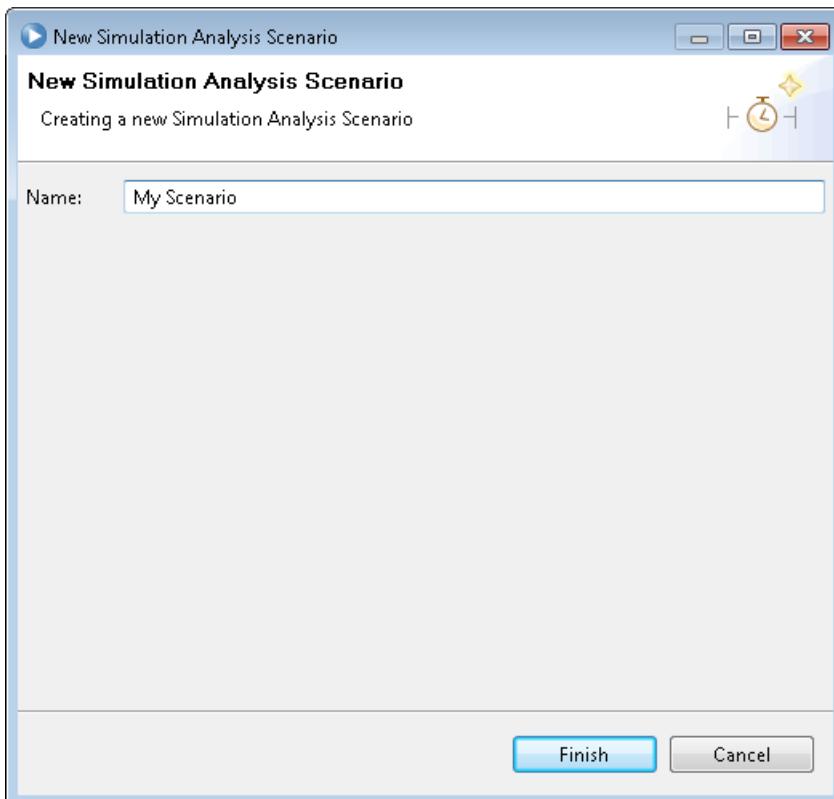
Simulation Analysis Scenarios

A simulation scenario is basically a collection of settings that are used to govern the execution of the simulation. A scenario describes what we are going to simulate and for how long we want that simulation to run.

A simulation analysis scenario is created from the Processes category within the library:



The scenario is given a name:



Within the details of the scenario a variety of attributes can be supplied.

First there is the simulated start date and time. This is the date and time when the simulation is set to begin. No processes will be simulated as having started before this time. This will be the epoch from which start events will be calculated.

When a simulation is to be performed, there must be a point at where the simulation should end. There are two exclusive choices for this. The first is a run time. This is the amount of elapsed time starting at the start time over which the simulation is to execute. For example, if we say that the start date/time will be 2012-08-25 from 09:00 and will run for 8 hours then the simulation will conclude at 2012-08-26 at 17:00. The number of instances of processes started will be governed by the firing rate of the process start event.

An alternative to duration is the idea of stopping after some number of instances of a named process have completed. For example, we can say that the simulation should end after 100 instances of process ABC have come to an end.

Within a simulation scenario we **must** supply the names of the Process Applications, Processes and Simulation Profiles. These are not optional. These will be the entities that are simulated.

The screenshot shows the 'Simulation Analysis Scenario' configuration window. It has a toolbar with back, forward, and search buttons, and a title bar 'My Scenario'. The main area is divided into sections:

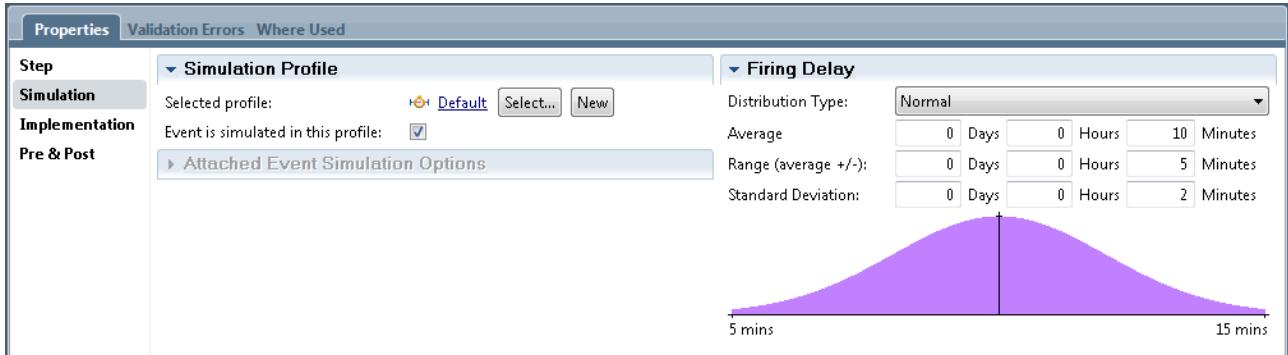
- Common:** Name: My Scenario, Modified: tw_admin (September 22, 2010 2:54:46 PM CDT), Documentation: (empty).
- Simulation Data Filters:** Start Time: 9/22/2010 14:54:47, Limit running time: Running time: 1 Days 0 Hours 0 Minutes, Limit process instances: By process: (empty), Max instances: 100.
- Process Apps to Include in Analysis:** A table with columns 'Name' and 'Profile' (empty). Buttons: Add, Remove, Up, Down.
- Processes to Include in Analysis:** A table with columns 'Name' and 'Profile' (empty). Buttons: Add, Remove, Up, Down.
- Participant Group Overrides:** A table with columns 'Name', 'Capacity', '% Availab...', '% Efficien...', and 'Cost \$/hr' (empty). Buttons: Add, Remove, Up, Down.

Simulation Profiles

Imagine we wish to run simulations with different sets of data to be able to ask and answer "what

if' kinds of questions. We can create a named grouping of simulation settings. This grouping is called a profile.

Associated with elements in the BPD diagram are simulation profiles:



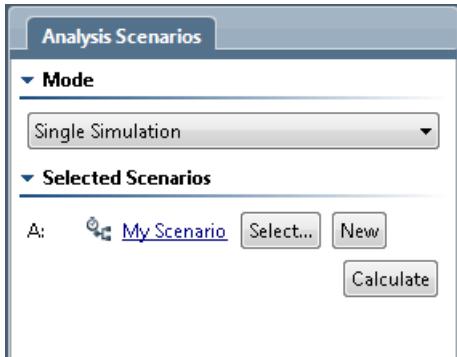
A simulation profile is the set of all simulating settings for a BPD. There can be multiple profiles and hence multiple sets of simulation settings.

Running Simulations

Once the environment for simulation has been created, we are now ready to execute those simulations and examine the results. Switching to the Optimize perspective shows the simulation details.

In this perspective, there are a number of views each with their own tab titles.

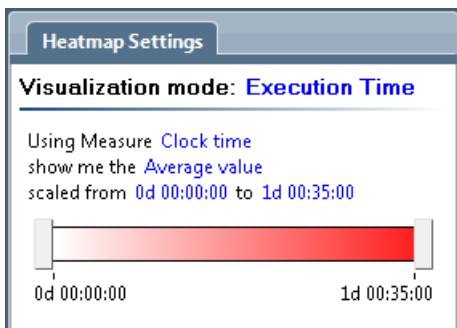
The Analysis Scenarios allows us to select a Scenario for execution and then execute it:



Clicking on the Calculate button causes the simulation metrics to be calculated.

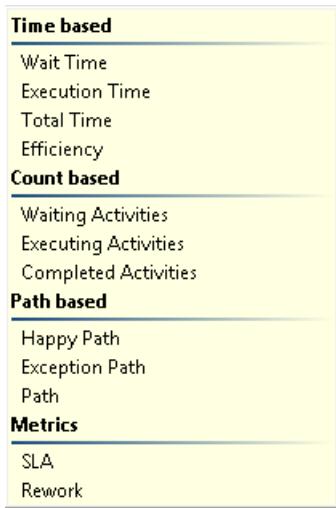
The interpretation of the results of the simulation is combination of examining the BPD diagram, the Heatmap Settings and the Live Reports view.

First, let us examine the Heatmap Settings.



In this area of the window the items marked in blue are changeable values. It is not known why they are not pull-downs.

The Visualization mode is of primary importance. It contains the following:



Selecting one of these report types changes the content in both the BPD diagram and the Live Reports view.

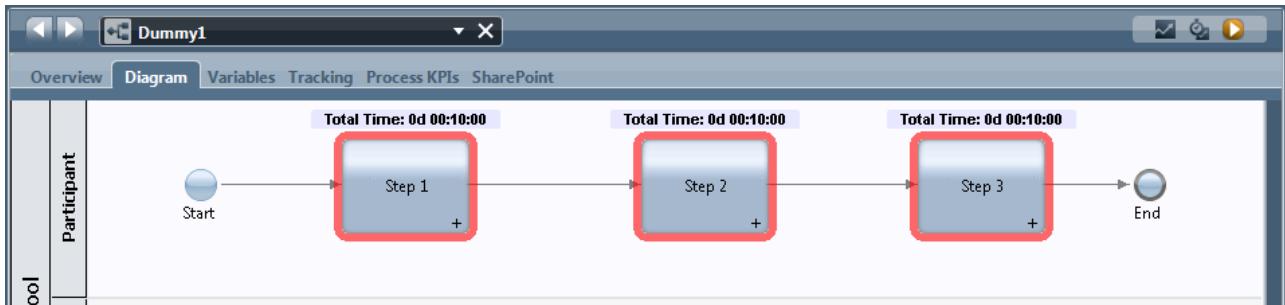
The first category is Time based.

Wait Time is the time that a task had to wait for execution because a staff member was not available to work upon it.

The Execution Time is the time taken to work on a task after a staff member became available.

The Total Time is the time taken including both waiting and execution for the task to complete.

Depending on the Visualization Mode selected, the BPD diagram will also change to be annotated with metrics about the simulation:

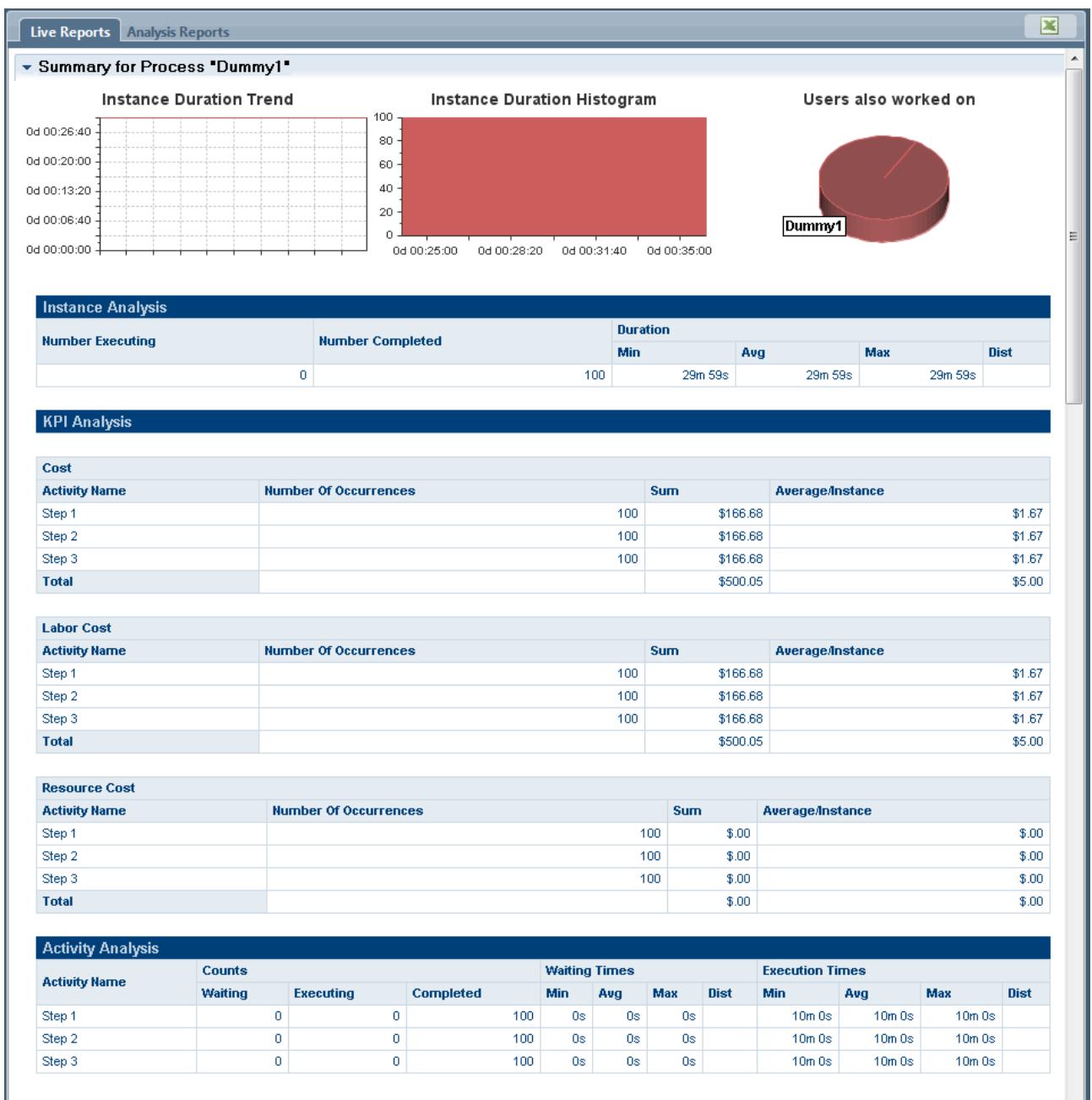


This provides an easy to read examination of the process as a whole to understand where time was spent.

IBPM uses "heatmap" diagrams to draw attention to important areas. Heatmap diagrams use saturation of color to depict the highs and lows. Activities will have richer colors around them if they need more attention than activities with softer colors.

Clicking in the BPD diagram will produce different reports in the Live Reports section.

Clicking in the white space of the process as a whole will produce a summary of the overall execution of the process.



From here we can get a great overall picture of the process including the costs, average durations and breakdown of the activities.

Clicking on a lane in the process will produce a report for all the activities contained in that lane including their details.

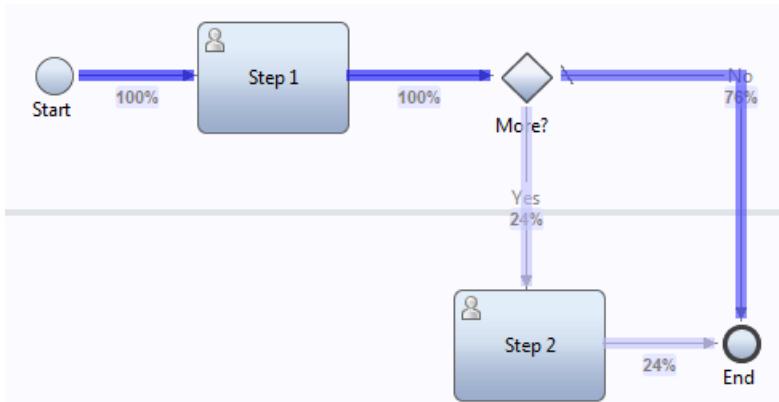
Finally, clicking on an activity step will produce a report on just that single activity.

Details are shown for various steps. These details include:

- The name of the activity
- The Instance ID of the process
- The time that the activity was created
- The time that the activity started
- The time that the activity ended

- Which Participation Group the activity was associated with
- The user within the group that was assigned the task
- The amount of time that the task spent waiting for a user
- The elapsed execution time for the task after being assigned a user
- The total amount of time that the task existed for

Another category of visualization is Path based.



This shows us visually the paths taken by the instances of the process. Heat-maps are used to indicate the relative frequency of paths taken. There are three path choices:

- Path
- Happy Path
- Exception Path

These filter all paths, only happy paths and only exception paths when choosing to show the heat-maps.

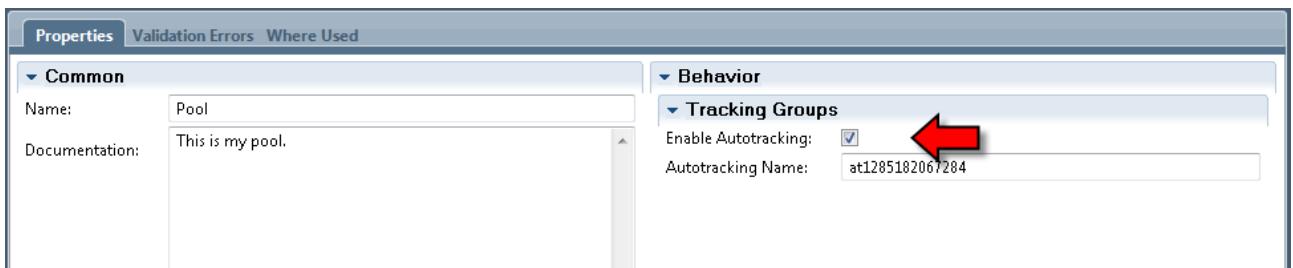
Optimizing a Process

Simulation is great for thinking about and asking questions for future processes or future changes to processes but this is only part of the story. As equally important a question is to determine where within the process improvements can be made. If a concern is raised that it is taking too long for customers to get their ordered products, unless you can tell where within that process time is being spent you won't know which areas would benefit from improvement.

The Optimizer component of IBPM serves double duty beyond that of being a simulation engine. The Optimizer can retrieve historical data from the Performance Data Warehouse and use that information to allow the process analyst to examine the reality of previously executed processes. The data that can be examined includes the same information as was previously seen in the simulations.

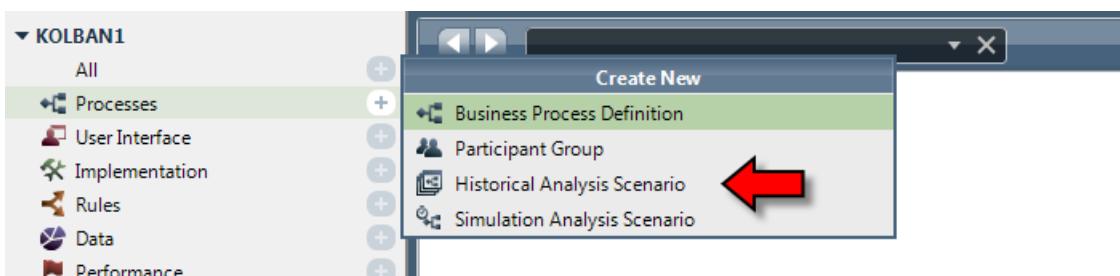
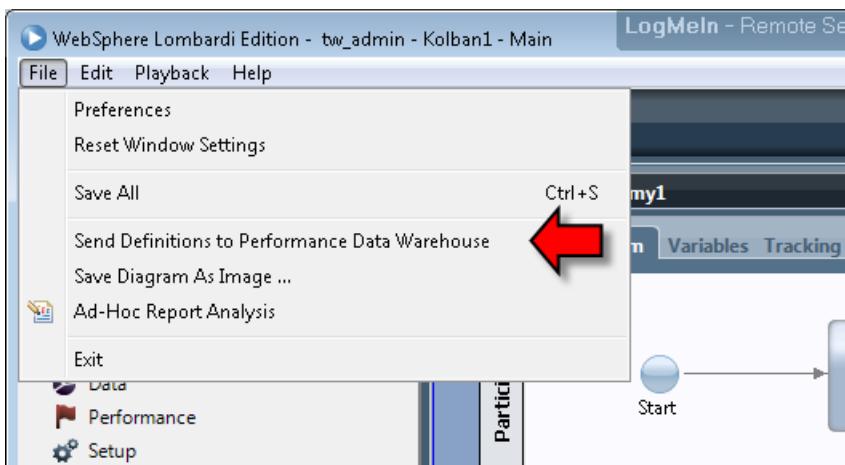
There is some setup required in order to have Process Server capture this information.

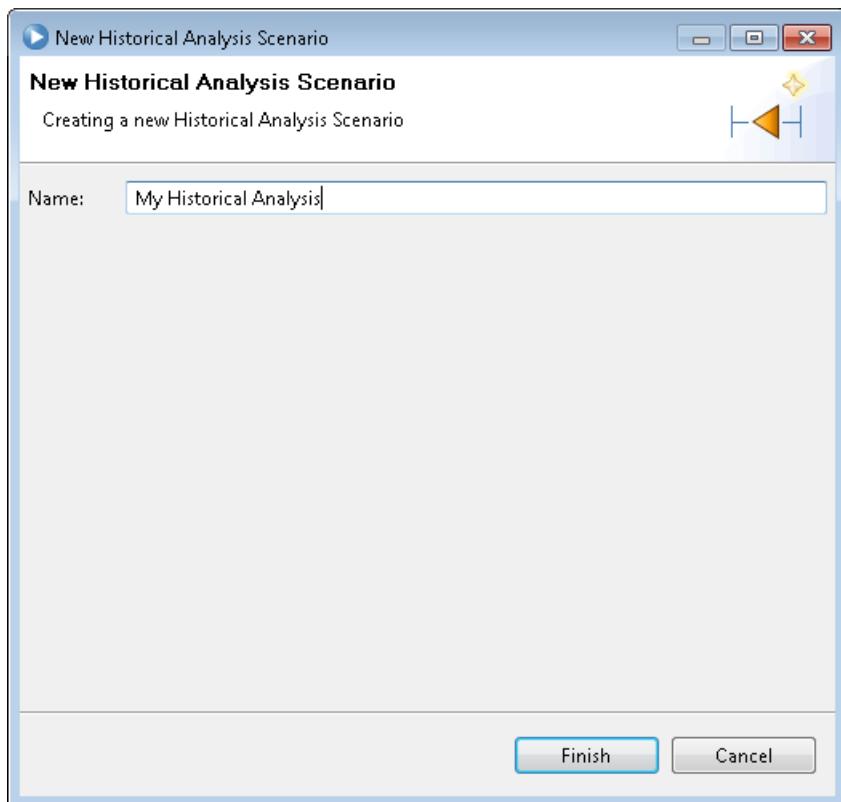
First, Auto-tracking must be enabled. This is set in the properties of the Pool:



This is enabled by default.

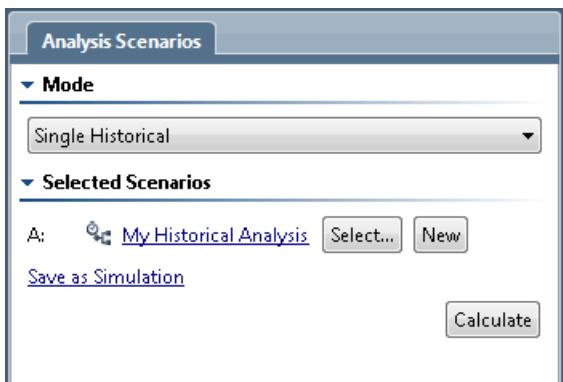
The tracking definitions must be sent to the Performance Data Warehouse:





The screenshot shows a configuration interface for a historical analysis scenario. The title bar says "* My Historical Analysis". The main area is titled "Historical Analysis Scenario".
Common:
- Name: My Historical Analysis
- Modified: tw_admin (September 22, 2010 6:31:28 PM CDT)
- Documentation: (empty text area)
Historical Data Filters:
- Include Process Instances: All In-Flight Only Completed Only
- Time Range:
- Start Date: 9/22/2010
- End Date: 9/22/2010
Process Apps to Include in Analysis:
- If the following table is empty, ALL process apps will be included.
- Kolban1 (All)
- Add Remove Up Down
Processes to Include in Analysis:
- If the following table is empty, ALL processes will be included.
- Name: Dummy1
- Add Remove
Business Data:
- Filter analysis results by Business Data (Auto-Tracked Field) across all processes.
- A table with columns: Variable, Comparison, Value. The table is currently empty.

In the Optimized Analysis Scenarios:



Start Event	Firing Delay
Activity	Execution Time
Condition	Outgoing Flow Percentages
End Event	Firing Delay

Simulation Tutorials

The areas of simulation and optimization make great reading and theory but an ounce of tutorial has been found to help many folks understand these ideas with greater clarity. In addition, working through a tutorial can result in more solidity when trying to remember these features in the future.

Experiment 1 – A basic start

In order to better understand the nature of simulation and optimization, here is a simple tutorial. First we build a trivial process that looks as follows:



We accept all the defaults. Now it is time to change some settings. For our purposes, we will change the arrival rate of new process instances to be one new process every 5 minutes.

Firing Delay

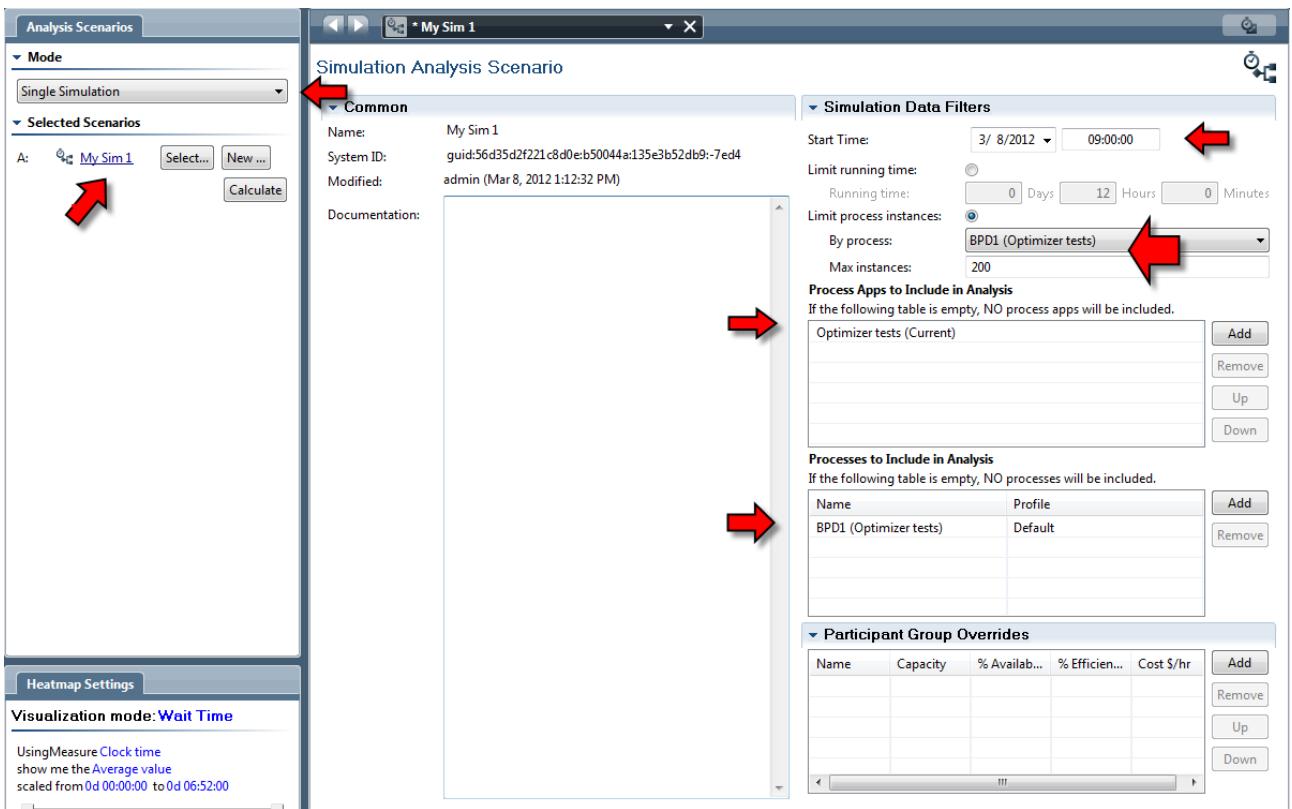
Distribution Type:	Fixed
Value	0 Days 0 Hours 5 Minutes

We will also say that the time it takes "Step 1" to execute will be 10 minutes.

Execution Time

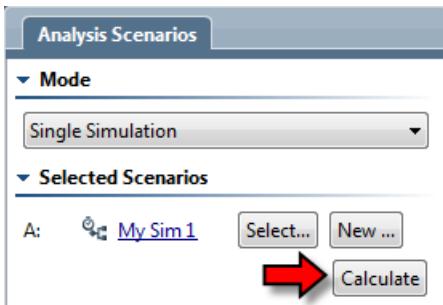
Distribution Type:	Fixed
Value	0 Days 0 Hours 10 Minutes

We now have a bare bones process model as well as a simulation model set up. In order to be able to run a simulation, we now switch to the Optimizer view. We select "Single Simulation" in mode and create a new simulation scenario. The simulation scenario describes what we are going to simulate and for how long.

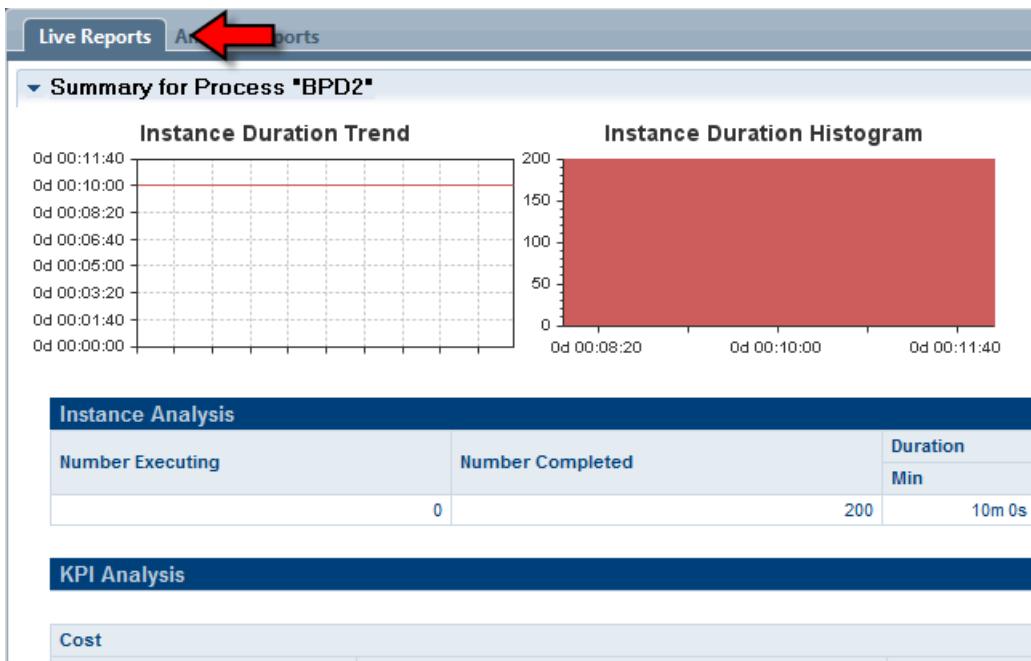


We are now ready to run a simulation. Before actually running it, what do you think the outcome may show?

After running the simulation by clicking the calculate button:



We can look at the Live Reports tab for the process as a whole.



One of the more interesting tables is the Instance Analysis table:

Instance Analysis		Duration			
Number Executing	Number Completed	Min	Avg	Max	Dist
0	200	10m 0s	10m 0s	10m 0s	10m 0s

This shows that 200 instances were completed (which was the number we asked to run) and that the minimum and maximum run-times for all instances were both 10 minutes. This means that each instance of the process took exactly 10 minutes to execute. This also makes sense to us as there is only one step in the process and that step is simulated to be exactly 10 minutes long. So ... is this what we expected to see? The answer would have been yes ... had it not been for the notion that we configured a new process to arrive every 5 minutes. If there had only been two process instances, the first one would arrive at 09:00 and take 10 minutes to complete finishing at 09:10. The second process would arrive at 09:05 ... but ... and here is the interesting thing, would we not have assumed that it could not complete until the first one had finished? If we had thought that would have happened, we would have been wrong. Clicking on an individual step in our process, in our case Step 1, we can see the simulated items:

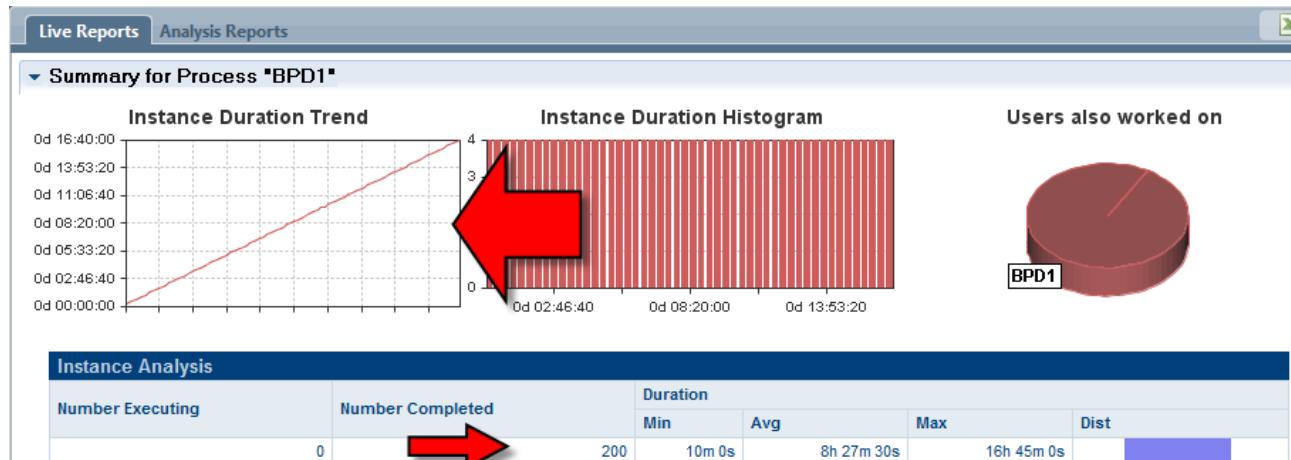
Summary for Activity "Step 1"							
Selected items summary							
1 items visited by 200 unique instances. Minimum duration: 0d 00:10:00 (Step 1)							
Maximum duration: 0d 00:10:00 (Step 1)							
Average duration: 0d 00:10:00							
Total activity executions: 200							
Activity	Instance ID	Creation time	Start time	End time	Assigned to group	User	
Step 1	1	Mar 8, 2012 9:05:00 AM	Mar 8, 2012 9:05:00 AM	Mar 8, 2012 9:15:00 AM	All Users	All Users_1	
Step 1	2	Mar 8, 2012 9:10:00 AM	Mar 8, 2012 9:10:00 AM	Mar 8, 2012 9:20:00 AM	All Users	All Users_2	
Step 1	3	Mar 8, 2012 9:15:00 AM	Mar 8, 2012 9:15:00 AM	Mar 8, 2012 9:25:00 AM	All Users	All Users_1	
Step 1	4	Mar 8, 2012 9:20:00 AM	Mar 8, 2012 9:20:00 AM	Mar 8, 2012 9:30:00 AM	All Users	All Users_2	
Step 1	5	Mar 8, 2012 9:25:00 AM	Mar 8, 2012 9:25:00 AM	Mar 8, 2012 9:35:00 AM	All Users	All Users_1	
Step 1	6	Mar 8, 2012 9:30:00 AM	Mar 8, 2012 9:30:00 AM	Mar 8, 2012 9:40:00 AM	All Users	All Users_2	

And there we see our answer. Step 1 was defined to be executed by members of a participant group called "All Users" which contains an unlimited number of theoretical users. As such, if a user is needed to perform that step there will never be a case that none are available.

We can constrain the number of users available in our simulation scenario:

Participant Group Overrides			
Name	Capacity	% Available	% Efficient
All Users (System Data)	1		

Re-running our simulation now shows us something new.

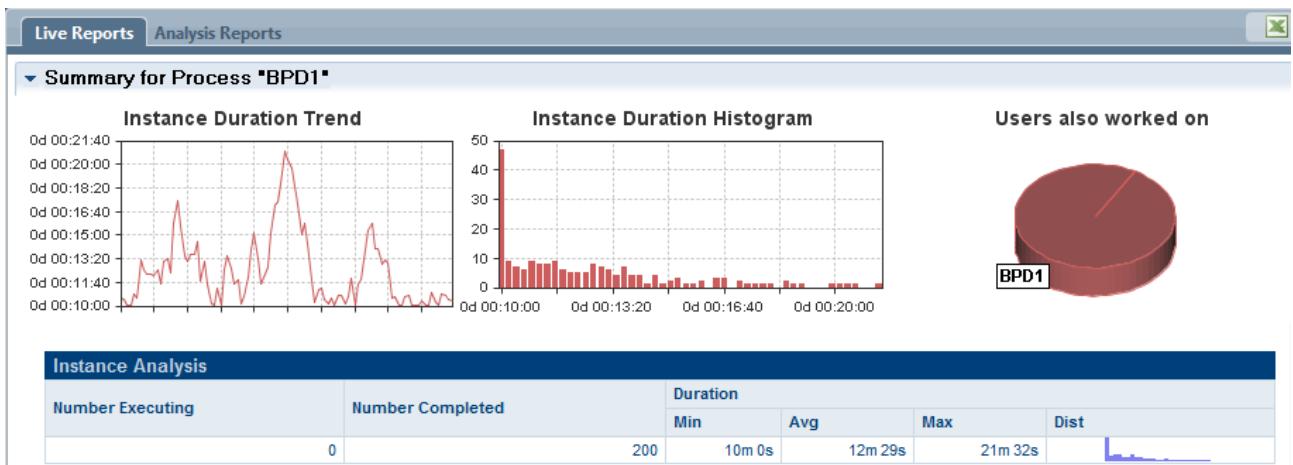


We now see that the time taken to complete the process is trending upwards and there is great deviation in completion minimum and maximum times.

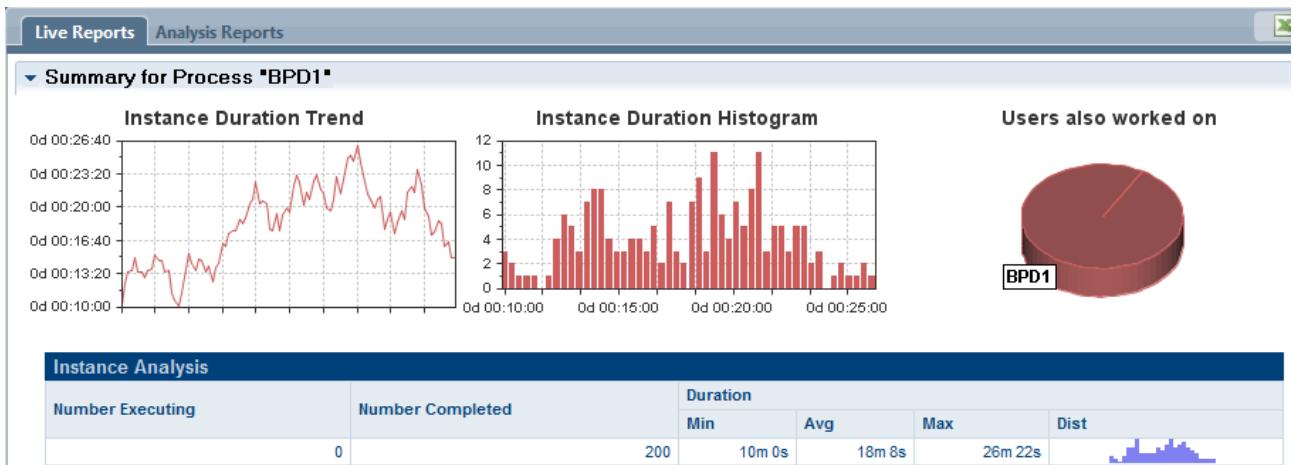
Experiment 2

Let us now change the starting rate of new process instances. Currently, it is set at a fixed rate of one new process every 5 minutes. Let us change this to be one new process on average every 10 minutes. The key words here are "on average". This means on occasion the next process instance will start in less than 10 minutes and on other occasions the next process instance will start in more than 10 minutes. Changing the process start distribution type to Uniform, we can control these settings:

In our example, we say that the average will be 10 minutes with a minimum of 8 minutes and a maximum of 10 minutes. Running a simulation and looking at the data, we get the following:



This is our most interesting graph yet. What it is showing is that the duration of process instances is no longer cleanly predictable but instead shows quite a broad range of results. If we run the simulation again, we will get different values because of the randomization of start times for the process instances.



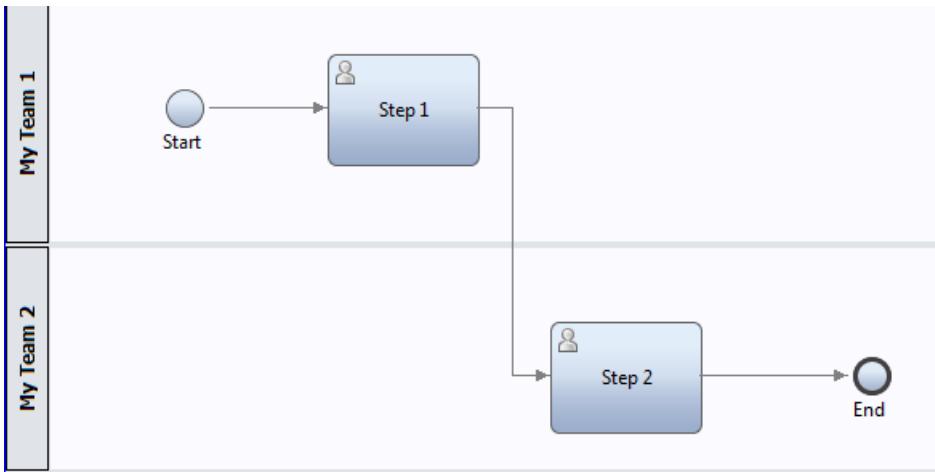
If we look further in the table, we see the following:

Activity Name	Counts			Waiting Times				Execution Times			
	Waiting	Executing	Completed	Min	Avg	Max	Dist	Min	Avg	Max	Dist
Step 1	0	0	200	0s	8m 8s	16m 23s		10m 0s	10m 0s	10m 0s	

What we see is that the waiting time for this step in the process is not zero. This means that on occasion this step was waiting for staff members to become available for the process to continue. If we think back to what we previously did, we said that this step only has one person available to do it. If we increased the number of staff members available to do this work, we will see the process completion rate become even again.

Experiment 3

We will now change some more settings. First we will add a new swim lane and two participant groups. Let us call them "My Team 1" and "My Team 2". We will associate each swim lane with a team. We will also add a second step.



For the second step, we will also define it as having a simulation value of 10 minutes exactly just as we did with Step 1.

In the simulation scenario, we will define each of the two teams as having only one member.

Participant Group Overrides

Name	Capacity	% Availab...	% Efl
My Team 1 (Optimizer tests)	1		
My Team 2 (Optimizer tests)	1		

Add **Remove** **Up**

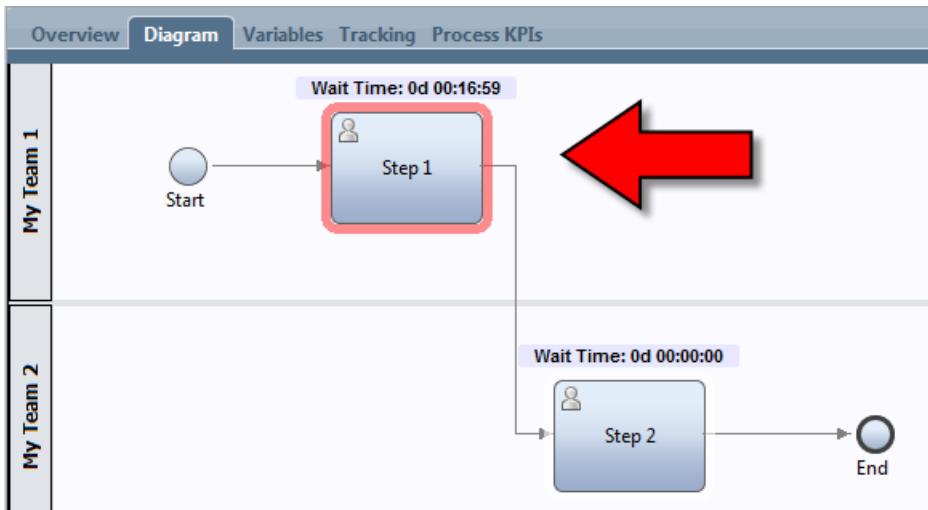
Again, we will re-run the tests. This time, we want to examine another aspect of the simulation tooling called heat-maps. In the Heat-map Settings, ensure that the Visualization mode is set to "Wait Time" and that the scaling range is less than 30 minutes.

Heatmap Settings

Visualization mode: Wait Time

Using Measure Clock time
show me the Average value
scaled from 0d 00:00:00 to 0d 00:22:08

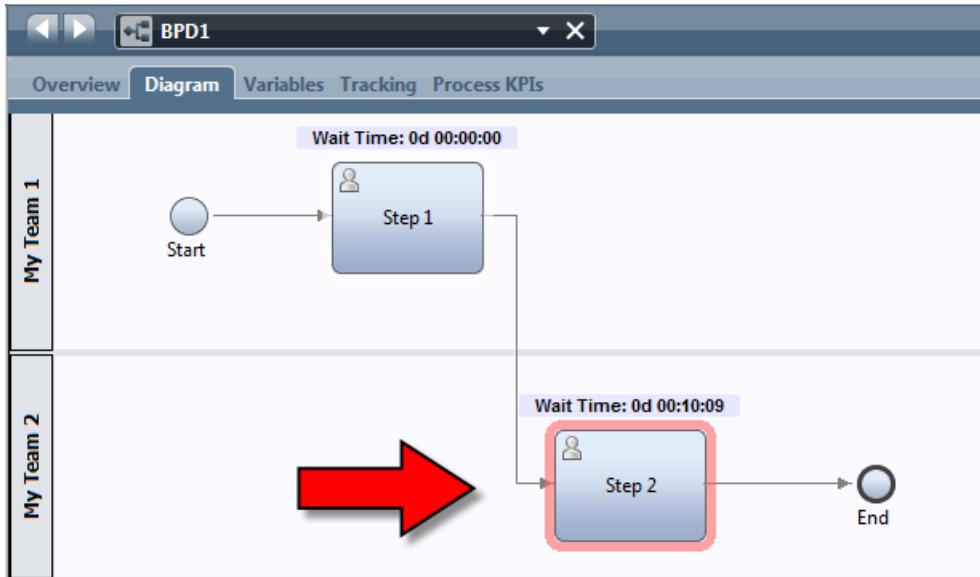
Looking at the BPD diagram in Optimizer mode we see an interesting piece of information:



We see that Step 1 has been highlighted using heat map visualization as one that needs our attention.

In our story, we already know where our bottlenecks are because we artificially created the process to illustrate such things. However, imagine a much larger process with potentially a lot of activities within it. By simply glancing at heat map diagrams like this, we can immediately see which areas of our process could use attention.

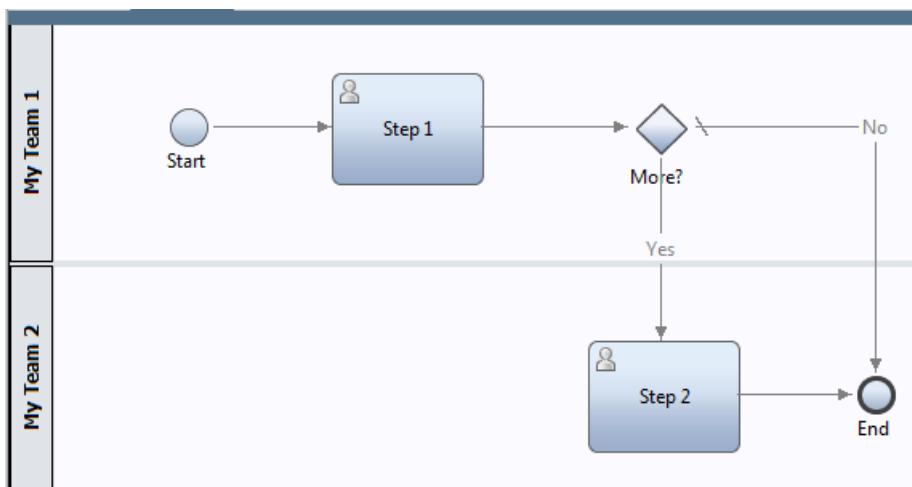
Next we increase the capacity of My Team 1 to be 2 members and rerun.



Notice that the heat map indication has moved from Step 1 to Step 2 indicating that we have resolved the bottleneck for Step 1 but now we have a bottleneck in Step 2. We could continue this onwards but the goal here was to show how heat maps can be used to illustrate where attention should be provided.

Experiment 4 – Path analysis

Again following on from the last experiment, we are now going to look at decision gateways. We add a gateway into the process flow to end up looking as follows:



What this represents is a simple process that has a decision within it. In the simulation settings for the gateway we define that 25% of the time, the decision will direct us to Step 2.

Properties Validation Errors Where Used

General **Simulation Profile** **Outgoing Flow Percentages**

Simulation Selected scenario: **Default** **Select...** **New ...**

Decision

Implementation

Pre & Post

Yes (Step 2) 25 No (End) (default)

Since these are exclusive decisions, this means that 75% of the time, the decision will take the other path. Having made this change, once again we execute the simulation.

In the Visualization mode, we select Path:

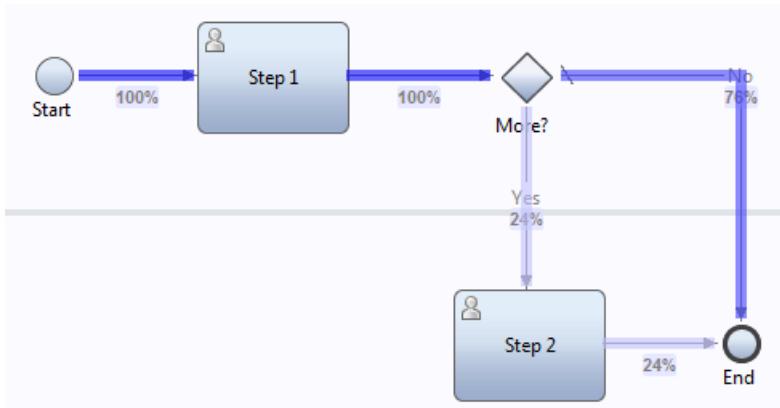
Heatmap Settings

Visualization mode: Path

Show me the % of instances that traversed each line at least 1 time

0% 100%

The BPD diagram now shows us the path's taken by the process, again visualized by heat maps and labels:



We can quickly see the weight of execution in different paths

IBPM Web API – Web Services API Access

When a Process Server runs it exposes a set of Application Programming Interfaces (APIs) that allow external applications to interact with the functions of the server. The binding interface to Process Server is provided through Web Services allowing these client applications to execute remotely from the server. IBPM calls these APIs the "Web API". Full JavaDoc is provided along with a copy of the Web Services WSDL file. The JavaDoc and the WSDL file can be found in the installation of the server at:

```
<WLE Install>/web-api/docs
```

Although there is a copy of the WSDL supplied on the file system it is **not** recommended that this be the file used to build the Web API clients. Instead, the WSDL file that is **actually** in use by the Process Server should be obtained. This can be retrieved from the following URL:

```
http://<Hostname>:<Port>/webapi/services/WebAPIService?wsdl
```

Where Hostname is the host serving the Process Server. The Port number can be found by examining the SystemOut console log for the server and looking for a line which reads (for example):

```
[9/23/10 11:07:07:273 CDT] 00000017 webcontainer I  
com.ibm.ws.wswebcontainer.VirtualHost addWebApplication SRVE0250I: Web Module  
Lombardi Software Web API has been bound to twprocsrv_host[*:19086].
```

The port number is shown at the end of the line.

Brining this page up in a browser shows something similar to the following:

This screenshot shows a Mozilla Firefox browser window displaying the WSDL document for the WebAPIService. The URL in the address bar is <http://win7-x64:19086/webapi/services/WebAPIService?wsdl>. The page content is as follows:

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

- <wsdl:definitions name="WebAPIService" targetNamespace="http://webapi.lombardisoftware.com">
  - <wsdl:documentation>
    Teamworks API for Web Service Clients Version: 7.0.1 Description: This API allows web service clients to access and manipulate Teamworks data. This service uses the document/literal convention. It complies with the WS-I Basic Profile 1.0 standard for web service interoperability. Other conventions used in this API that have been chosen for cross-platform ease of use are as follows:
      - Elements and type declaration use Pascal case (upper case first letter)
      - Elements within a type representing class properties use Pascal case
      - Elements within a type representing API parameters use Camel case (lower case first letter)
      - Enumeration values are in all caps with words separated by underscores
      - Arrays of elements use the ArrayOfElement naming convention
      - Element values that can be null use both the minOccurs="0" and nillable="true" for broadest compatibility with clients
      - Elements that can be partially loaded declare their Id property as required and other properties as allowing nulls
      - Where possible, internal Teamworks APIs have been rationalized to be consistent with each other and with the naming conventions chosen for this API
      - Id properties are always specific to the Teamworks environment and should NOT be used across multiple environments. Ids are usually not visible in the Teamworks authoring environment for library items. They are visible for process instances (Instance Id) and tasks (Task Id) via the Process Inspector.
      - SystemId properties are not environment specific and can be used across multiple Teamworks environments. They are
```

From here, the WSDL can be saved by right clicking in the page and selecting "Save Page As...".

Here is a summary of the APIs provided:

Method name	Description
AbortProcessInstance	
AbortProcessInstances	
AddComment	
AddHelpRequest	
AssignTask	
AssignTasks	
ChangeProcessInstanceDueDate	
ChangeProcessInstancesDueDate	
ChangeTaskDueDate	
ChangeTaskPriority	
ChangeTasksDueDate	
ChangeTasksPriority	
CompleteTask	
CreateDocument	
DeleteDocument	
DueDateOkForTask	
ExecuteSearch	Execute a search for tasks or processes
GetActionPermissions	
GetAllRoles	Retrieve a list of all the roles/groups
GetAllUsers	Retrieve a list of all users. Be cautious using this API especially if the system is configured with an LDAP provider.
GetCurrentUser	Retrieve the current user and the roles that user is in
GetDocument	
GetDocumentsForProcessInstance	
GetDocumentsForProperties	
GetExposedItems	
GetFavorites	
GetPriorities	
GetProcessBySystemId	
GetProcessInstance	
GetProcessInstanceBusinessData	
GetProcessInstanceForSavedSearch	
GetRoleByName	Retrieve details of a role given its name
GetSavedSearches	
GetScoreboardData	
GetScoreboards	
GetSearchMetaData	

GetSearchMetaDataForProcess	
GetServerInfo	
GetTask	
GetTasksForSavedSearch	
GetUserByName	
GetUserConfiguration	
IgnoreHelpRequest	
ReassignTask	
ReassingTasks	
ReasignTasksToRole	
ReassignTasksToUser	
ReasignTaskToRole	
ReassignTaskToUser	
RefreshGroupMemebership	
ReplyHelpRequest	
ResumeProcessInstance	
ResumeProcessInstances	
RunFavorite	
SendTask	
StartExposedItem	
StartProcess	
StartProcessWithInfoPathForm	
StartTask	
SubmitInfoPathForm	
SubmitInfoPathForms	
SuspendProcessInstance	
SuspendProcessInstances	
SynchronizeTaskLists	
TestConnection	Tests the connection from the Web Service client to the Process Server hosting the Web API
UpdateDocument	

Here are the APIs by category:

Process Interaction

abortProcessInstance	
abortProcessInstances	
changeProcessInstanceDueDate	
changeProcessInstancesDueDate	
getProcessBySystemId	

getProcessInstance	
getProcessInstanceBusinessData	
getProcessInstancesForSavedSearch	
resumeProcessInstance	
resumeProcessInstances	
startProcess	
startProcessWithInfoPathForm	
suspendProcessInstance	
suspendProcessInstances	

Task Interactions

assignTask	
assignTasks	
changeTaskDueDate	
changeTaskPriority	
changeTasksDueDate	
changeTasksPriority	
completeTask	
dueDateOkForTask	
getTask	
getTasksForSavedSearch	
reassignTask	
reassignTasks	
reassignTasksToRole	
reassignTasksToUser	
reassignTaskToRole	
reassignTaskToUser	
sendTask	
startTask	
submitInfoPargForm	
submitInfoPathForms	
synchronizeTasksLists	

Documents and Comments

addComment	
addHelpRequest	
createDocument	
deleteDocument	
getDocument	
getDocumentsForProcessInstance	

getDocumentsForProperties	
ignoreHelpRequest	
replyHelpRequest	
updateDocument	

Searching

executeSearch	
getExposedItems	
getFavorites	
getPriorities	
getSavedSearches	
getSearchMetaData	
getSearchMetaDataForProcess	
getServerInfo	
runFavorite	
startExposedItem	

Users, groups and security

getActionPermissions	
getAllRoles	Retrieve a list of all the roles/groups
getAllUsers	Retrieve a list of all users. Be cautious using this API especially if the system is configured with an LDAP provider.
getCurrentUser	
getRoleByName	Retrieve details of a role given its name
getUserByName	
getUserConfiguration	
refreshGroupMembership	

Performance data

getScoreboardDara	
getScoreboards	

A mandatory SOAP header is required when sending requests to IBPM. This SOAP Header is a ClientInfo structure that looks as follows:

```

<soapenv:Header>
  <web:ClientInfo>
    <web:TimeZone>CST</web:TimeZone>
    <web:Locale>EN</web:Locale>
    <web:AcceptsTimeZone>false</web:AcceptsTimeZone>
  </web:ClientInfo>
</soapenv:Header>

```

Where the web suffix is bound to the name space called "http://webapi.lombardisoftware.com".

An example SOAP request to the WebAPI looks as follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:web="http://webapi.lombardisoftware.com">
  <soapenv:Header>
    <web:ClientInfo>
      <web:TimeZone>CST</web:TimeZone>
      <web:Locale>EN</web:Locale>
      <web:AcceptsTimeZone>false</web:AcceptsTimeZone>
    </web:ClientInfo>
  </soapenv:Header>
  <soapenv:Body>
    <web:GetAllRoles/>
  </soapenv:Body>
</soapenv:Envelope>
```

See Also:

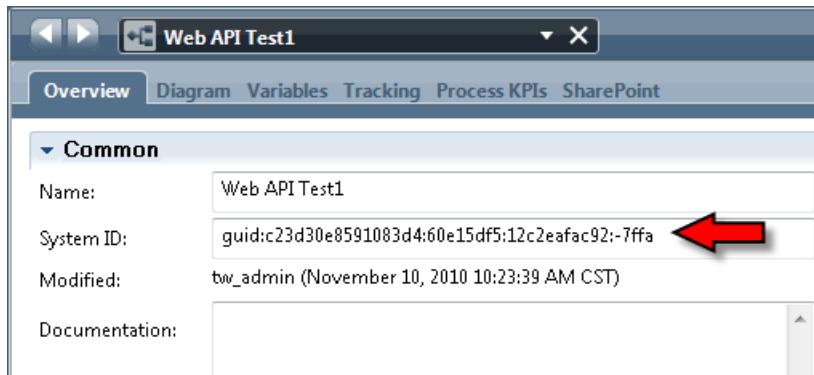
- TechNote – [How to use soapUI to test WebAPI in WLE](#)

Starting a Process

We can start a new instance of a process using the `startProcess` operation. This has two input parameters:

Parameter Name	Data Type
process	Process
inputs	Variable[]

The process is retrieved by executing a `getProcessBySystemId()` which takes the Application Acronym and the `processSystemId` as parameters. The `processSystemId` is shown in the Overview section of a BPD only when the Advanced Features mode of PD is enabled.



The following code fragment will invoke a process that does not expect any parameters:

```
String acronymName = "WEBAPI1";
String processSystemId = "guid:c23d30e8591083d4:60e15df5:12c2eafac92:-7ffa";
com.lombardisoftware.webapi.Process p = webAPI.getProcessBySystemId(acronymName,
processSystemId);

webAPI.startProcess(p, null);
```

The acronym name and the process system id would be changed to suit the process that is actually being invoked. If the process being invoked has input parameters then these can be supplied by the caller. Here is an example of passing in a simple string parameter:

```
ObjectFactory objectFactory = new ObjectFactory();

String acronymName = "WEBAPI1";
String processSystemId = "guid:c23d30e8591083d4:60e15df5:12c2eafac92:-7ffa";
com.lombardissoftware.webapi.Process p = webAPI.getProcessBySystemId(acronymName,
processSystemId);

Variable v1 = objectFactory.createVariable();
v1.setName(objectFactory.createVariableName("inp1"));
v1.setValue(objectFactory.createVariableValue("Hello World"));
ArrayOfVariable arrayOfVariable = objectFactory.createArrayOfVariable();
arrayOfVariable.getVariable().add(v1);

webAPI.startProcess(p, arrayOfVariable );
```

If the expected parameters are a complex data type, things become a little more challenging. A working recipe is shown next:

```
ObjectFactory objectFactory = new ObjectFactory();

String acronymName = "WEBAPI1";
String processSystemId = "guid:c23d30e8591083d4:60e15df5:12c2eafac92:-7ffa";
com.lombardissoftware.webapi.Process p = webAPI.getProcessBySystemId(acronymName,
processSystemId);

com.kolban.data.ObjectFactory of2 = new com.kolban.data.ObjectFactory();
MyDataType mdt = of2.createMyDataType();
mdt.setA(of2.createMyDataTypeA("A"));
mdt.setB(of2.createMyDataTypeB("B"));
mdt.setC(of2.createMyDataTypeC("C"));
ComplexValue complexValue = new ComplexValue();
complexValue.setAny(mdt);

Variable v1 = objectFactory.createVariable();
v1.setName(objectFactory.createVariableName("inp1"));
v1.setValue(objectFactory.createVariableValue(complexValue));
ArrayOfVariable arrayOfVariable = objectFactory.createArrayOfVariable();
arrayOfVariable.getVariable().add(v1);

webAPI.startProcess(p, arrayOfVariable );
```

To understand this code, realize that the target process was defined to have an input variable of a custom type called MyDataType. An XSD representing this type was exported from PD and saved in a file. An import statement was added to the WSDL so that the JAX-WS environment added the XSD to the list of Java classes to generate. The input parameter was then built and populated and finally encapsulated in a "ComplexValue" object before being added to the parameters sent to the target server.

Taking a trace of the SOAP message, the following was found:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header>
<p:ClientInfo xmlns:p="http://webapi.lombardissoftware.com">
<p:TimeZone>CST</p:TimeZone>
<p:Locale>EN</p:Locale>
<p:AcceptsTimeZone>false</p:AcceptsTimeZone>
</p:ClientInfo>
</soapenv:Header>
```

```

<soapenv:Body>
  <ns2:StartProcess xmlns="urn:data.kolban.com"
  xmlns:ns2="http://webapi.lombardisoftware.com">
    <ns2:Process>
      <ns2:Reference>/25.19641cab-e08b-457a-b21f-a4448c86488f</ns2:Reference>
      <ns2:SnapshotId>2064.4cba9ee4-5502-455a-afb7-
38f71370dee9</ns2:SnapshotId>
      <ns2:Name>Web API Test1</ns2:Name>
      <ns2:SystemId>guid:c23d30e8591083d4:60e15df5:12c2eafac92:-
7ffa</ns2:SystemId>
    </ns2:Process>
    <ns2:inputs>
      <ns2:Variable>
        <ns2:Name>inp1</ns2:Name>
        <ns2:Value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns2:ComplexValue">
          <MyDataType>
            <a>A</a>
            <b>B</b>
            <c>C</c>
          </MyDataType>
        </ns2:Value>
      </ns2:Variable>
    </ns2:inputs>
  </ns2:StartProcess>
</soapenv:Body>
</soapenv:Envelope>

```

Notes on Object Factory

In the preceding code, there was referenced to an apparently mysterious class called ObjectFactory. What is this, where did it come from and how is it used? When a WebService is exposed or called, that WebService may expose parameters of complex data types. These data types are described in XML Schema Definition (XSD). When we work in a Java environment, these XSDs are not of much value to us. What we would rather work with are *JavaBeans* which are Java classes that expose a complex data type as a Java Object with associated getter and setter methods to access the data. When using JAX-WS to work with Web Services, each time a WSDL is processed with the JAX-WS tools and new complex data types are found, JavaBeans are generated and a factory object is created to create instances of those JavaBeans. The factory object is called ObjectFactory. For more details on this concept, google "JAX-WS ObjectFactory".

Executing a search

The Web Api provides the ability to execute a search. The operation for this is called `executeSearch` and has the following parameters:

Parameter Name	Data Type
search	Search
maxRows (optional)	integer
beginIndex (optional)	integer

The key to executing a search is to correctly populate the Search object. This is a rich structure and should be read carefully to ensure that it is fully understood. The return from a call to `executeSearch` is a `SearchResults` Object.

The way to think about a search is that when the search is executed, a matrix of data is returned.

The matrix is composed of rows where each row is broken into a series of columns.

	Column 1	Column 2	Column 3	Column 4	Column 5
Row 1					
Row 2					
Row n					

When a search is executed, we explicitly name the columns of data that we want returned. If a column is not included in the list of columns to be returned, it is not returned. The reason behind this is that data that is not needed by the caller is simply not returned back to the caller and hence reduces the amount of work and data that needs to be performed and transmitted.

The rows correspond to instances of data. The search conditions are applied to each row of data and only those rows that pass the condition filter are returned. Combining this was the columns, for each row that passes the search condition, the columns to be returned are extracted from the rows and only these values are returned.

The `SearchResult` object contains two important parts. One part is a description of which columns have been returned. This should match the columns requested to be returned. The other part is an array of rows where each row has passed the filter condition.

Getting the details of a task.

We can retrieve the details of a Task using the `getTask` operation. This operation takes one parameter which is the id of the task to be retrieved. A Task object is returned.

Web API Data types

The Web API has a large number of data types (data structures) associated with it. Making Web Service calls takes input parameters and responses from Web Service calls also returns output parameters. The types of these parameters are described in the WSDL describing the Web API. The following are descriptions of many of the data types used in the Web API.

CustomProperty

name	String – The name of the Custom Property
value	String – The value of the Custom Property

ExternalActivity

customProperties	An array of CustomProperty
inputParameters	An array of Parameters
name	String – The name of the External Activity
outputParameters	An array of Parameters

reference	String
snapshotId	String
systemId	String

ExternalActivityAttachment

data	ExternalActivityData
externalActivity	ExternalActivity

ExternalActivityData

variables	An array of Variable
-----------	----------------------

Parameter

name	String – The name of the parameter
type	QName – The data type of the parameter

Process

name	String
reference	String
snapshotId	String
systemId	String

Search

The Search data type contains:

columns	an array of SearchColumn. These are the columns that are returned in the search.
conditions	an array of SearchCondition
orderBy	an array of SearchOrdering
organizedByType	String. Can be one of: • "Task" • "ProcessInstance"

A Search object describes the parameters of a search to be executed against the IBPM server. It contains the following key parts:

- The list of columns of data to be returned from the search. This describes the subset of possible columns returned when a search is executed. An array of SearchColumn objects are passed in. Each entry in the array corresponds to column data being returned from the search.
- The conditions used to include a result in the output following a search. This is used to filter

the results based on data. The parameter is an array of `SearchCondition` objects where each search condition object contributes a predicate in the search.

SearchColumn

The `SearchColumn` data type contains:

displayName	String
name	String
type	String. One of: • "BusinessData" • "Process" • "ProcessInstance" • "Task"

The `SearchColumn` is used to define which of the possible columns of data are to be returned. A class called `SearchableTaskColumn` is defined that contains pre-made definitions for tasks.

Activity	
AssignedToRole	
AssignedToUser	
ClosedBy	
ClosedDate	
DueDate	
Id	
Priority	
ReadDate	
ReceivedDate	
ReceivedFrom	
SentDate	
Status	
Subject	

Similar structures are available for the `ProcessColumn` and `ProcessInstanceColumn`.

SearchColumnMetaData

The `SearchColumnMetaData` is used to describe the nature of a column of data returned.

displayName	<TBD> Unknown. Currently returns null
name	The name of the column
type	one of: • Task
usableInSearchCondition	A boolean that indicates if this column can be used in a search.
valueType	The type of data that this column represents. Examples are: • {http://www.w3.org/2001/XMLSchema}string • {http://www.w3.org/2001/XMLSchema}long

- | |
|--|
| <ul style="list-style-type: none"> • <code>{http://www.w3.org/2001/XMLSchema}dateTime</code> • <code>{http://www.w3.org/2001/XMLSchema}double</code> |
|--|

SearchCondition

The SearchCondition data type contains:

column	A SearchColumn
operator	A string. Can be one off: <ul style="list-style-type: none"> • "CONTAINS" • "EQUALS" • "GREATER_THAN" • "LESS_THAN" • "NOT_EQUALS" • "STARTS_WITH"
value	It is suspected that the value is a function of the data type of the column. In the generated class it is defined as a JAXBELEMENT.

SearchMetaData

There are two methods that return an object of this type. These are `getSearchMetaData()` and `getSearchMetaDataForProcess()`. The power of these methods is to query the IBPM runtime to determine which columns can be searched in a search call.

columns	An array of SearchColumnMetaData.
process	Process

SearchResults

The SearchResults data type is returned from a call to `executeSearch`. This data type contains:

columns	an array of SearchColumnMetaData
rows	an array of SearchResultRow
totalRowCount	An int value containing the number of rows returned.

SearchResultRow

The SearchResultRow data type represents a single row of returned Search results"

values	An array of values associated with this row
--------	---

Task

The Task data type contains:

actionPermissions	
-------------------	--

activityName	
assignedToRole	
assignedToUser	
attachedExternalActivity	
attachedInfoPathForm	
businessData	
closedBy	
closedDate	
dueDate	The date the task is due.
id	The id of the task.
narrative	
participant	
participantDisplayName	
priority	The priority of the task.
priorityRanking	
processInstance	The details of the process that started this task.
readDate	
receivedDate	
sentDate	
serverAddress	
service	
sharepointDiscussionUrl	
status	
subject	

TaskStatus

The TaskStatus can be:

- "Actioned"
- "Alert"
- "Answered_Help_Request"
- "Closed"
- "Comment"
- "Deleted"
- "Forwarded"
- "Help_Request"
- "Ignored_Help_Request"
- "New"

- "New_or_Received"
- "Received"
- "Replied"
- "Sent"
- "Special"

User

The User represents a user definition as known to IBPM. Functions that refer to users include:

- getAllUsers()
- getCurrentUser()
- getUserByName()

id	long
name	String
roleMemberships	Array of Role

Variable

name	String
value	Object

The variable types map as follows:

XSD simple type	WLE simple type
string	String
normalizedString	String
token	String
language	String
Name	String
NMTOKEN	String
NMTOKENS	String
NCName	String
ID	String
IDREF	String
ENTITY	String

IDREFS	String
ENTITIES	String
base64Binary	String
hexBinary	String
anyURI	String
QName	String
NOTATION	String
gYearMonth	String
gYear	String
gMonthDay	String
gDay	String
gMonth	String
decimal	Decimal
double	Decimal
float	Decimal
int	Integer
integer	Integer
short	Integer
long	Integer
byte	Integer
nonNegativeInteger	Integer
nonPositiveInteger	Integer
negativeInteger	Integer
positiveInteger	Integer
unsignedLong	Integer
unsignedInt	Integer
unsignedShort	Integer
unsignedByte	Integer
date	Date
dateTime	Date
time	Time

boolean	Boolean

WLE simple variable type	XSD simple type
String	string
Integer	int
Decimal	double
Date	dateTime
Time	dateTime
Selection	string
Boolean	boolean

Variable

name	String
value	Object

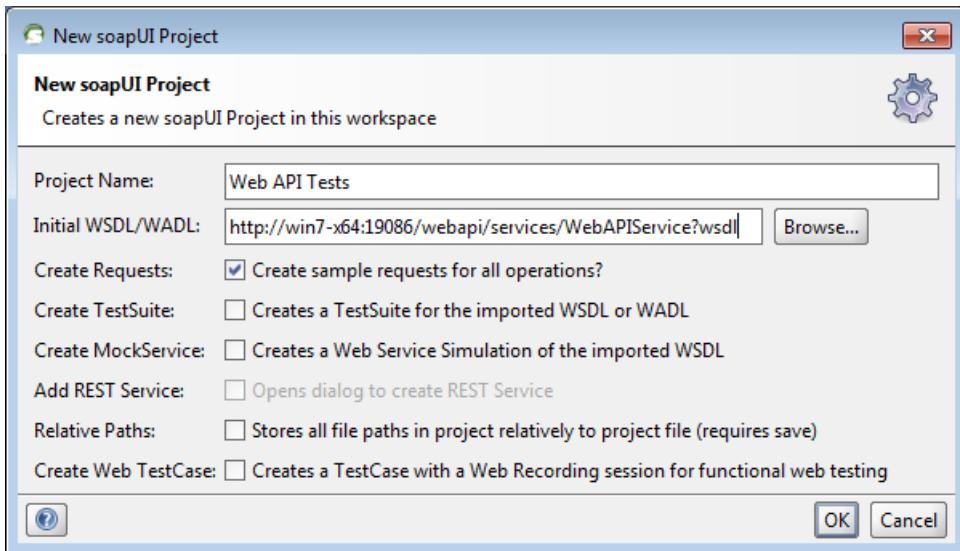
Exposed item types

- Process
- Scoreboard
- Report
- Service

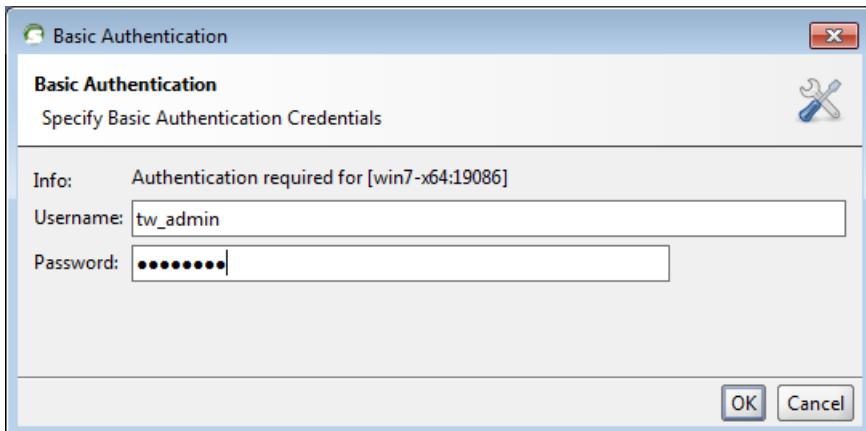
Testing Web API services with soapUI

Some of the Web API operations can be quite complex in nature. For example, a search request has a lot of parameters and options. Before committing to coding a solution, it can be beneficial to make the service calls directly in a test environment to validate that what you think will be returned works and is as expected. Using soapUI to invoke Web API operations can be a good solution.

Create a new soapUI project and provide the URL for where the Web API WSDL can be obtained:



You may be prompted for authentication to access the WSDL:



The project will be populated with all the operations discovered by retrieving the WSDL. We will now demonstrate an example execution of an operation. Let us choose a simple one ...
GetCurrentUser.

Open the Request for the GetCurrentUser operation. The ClientInfo header fields must be populated. In addition, authentication information must be provided:

Request 1

http://win7-x64:19086/webapi/serv

Raw XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <web:ClientInfo>
      <web:TimeZone>CST</web:TimeZone>
      <web:Locale>EN</web:Locale>
      <web:AcceptsTimeZone>false</web:AcceptsTimeZone>
    </web:ClientInfo>
  </soapenv:Header>
  <soapenv:Body>
    <web:GetCurrentUser/>
  </soapenv:Body>
</soapenv:Envelope>
```

Username: tw_admin

Password: tw_admin

Domain:

Outgoing WSS:

Incoming WSS:

Head... Attach... ... W... JMS... JMS Pro...

response time: 4ms (0 bytes)

Once executed, the response XML can be seen:

The screenshot shows the SOAPUI interface with two requests. Request 1 (left) is a SOAP envelope with a header containing a 'ClientInfo' block. The 'AcceptsTimeZone' element is highlighted in yellow and has a value of 'false'. Request 2 (right) is a response from the 'GetCurrentUser' operation, listing various user roles with their IDs and names. The roles listed are:

- Role 2: Name=Debug, DisplayName=Debug
- Role 3: Name=tw_admins, DisplayName=tw_admins
- Role 4: Name=tw_authors, DisplayName=tw_authors
- Role 15: Name=tw_process_owners, DisplayName=tw_process_owners
- Role 16: Name=tw_allusers, DisplayName=tw_allusers
- Role 1002: Name=All Users_T_da7e4d23-78cb-4483, DisplayName=All Users

See also:

- [TechNote: How to use SOAPUI to test WebAPI operations in Teamworks and WebSphere Lombardi Edition](#)

Building a Java Client

Once the WSDL has been retrieved, we can use the WID Web Service generate client to generate a Java Client that can invoke the WSDL. Unfortunately, WID 7 seems to have a problem with the WSDL format. The WSDL contains documentation tags of the format:

```
<a:since ...>
```

These cause the Java Client generation to fail. I used a grep tool to remove these lines with the following command:

```
grep -v a:since originalWSDL.wsdl > newWSDL.wsdl
```

To test the Web Service interface from the JAX-WS generated code, the following is a suggested sample:

```
WebAPIService webAPIService = new WebAPIService();
WebAPI webAPI = webAPIService.getWebAPISoap();
```

```

BindingProvider bp = (BindingProvider) webAPI;
bp.getRequestContext().put(BindingProvider.USERNAME_PROPERTY, "admin");
bp.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY, "admin");

webAPI.testConnection();

System.out.println("Connection worked!");

```

When using JAX-WS, for testing, the SOAP request can be redirected through a TCP/IP Monitor with the following:

```

BindingProvider bp = (BindingProvider) webAPI;
bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
"http://win7-x64:9089/webapi/services/WebAPIService");

```

where the TCP/IP monitor is listening on port 9089 and redirecting to the target port.

A Web service request to IBPM requires some soap headers to be added, an example fragment of code that achieves this is shown here:

```

Map <QName, List<String>> headersMap = new HashMap<QName, List<String>>();
QName myQName = new QName("http://webapi.lombardisoftware.com", "myHeader");
String mySoapHeader =
    "<p:ClientInfo xmlns:p='http://webapi.lombardisoftware.com'>" +
    "<p:TimeZone>CST</p:TimeZone>" +
    "<p:Locale>EN</p:Locale>" +
    "<p:AcceptsTimeZone>false</p:AcceptsTimeZone>" +
    "</p:ClientInfo>";
List<String> mySOAPHeaders = new ArrayList<String>();
mySOAPHeaders.add(mySoapHeader);
headersMap.put(myQName, mySOAPHeaders);

BindingProvider bp = (BindingProvider) webAPI;
bp.getRequestContext().put("jaxws.binding.soap.headers.outbound", headersMap);

```

here is a complete sample illustrating a search:

```

WebAPIService webAPIService = new WebAPIService();
WebAPI webAPI = webAPIService.getWebAPISoap();

Map <QName, List<String>> headersMap = new HashMap<QName, List<String>>();
QName myQName = new QName("http://webapi.lombardisoftware.com", "myHeader");
String mySoapHeader =
    "<p:ClientInfo xmlns:p='http://webapi.lombardisoftware.com'>" +
    "<p:TimeZone>CST</p:TimeZone>" +
    "<p:Locale>EN</p:Locale>" +
    "<p:AcceptsTimeZone>false</p:AcceptsTimeZone>" +
    "</p:ClientInfo>";

List<String> mySOAPHeaders = new ArrayList<String>();
mySOAPHeaders.add(mySoapHeader);
headersMap.put(myQName, mySOAPHeaders);

BindingProvider bp = (BindingProvider) webAPI;

```

```

bp.getRequestContext().put(BindingProvider.USERNAME_PROPERTY, "kolban");
bp.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY, "password");

bp.getRequestContext().put("jaxws.binding.soap.headers.outbound", headersMap);
ObjectFactory objectFactory = new ObjectFactory();

Search search = objectFactory.createSearch();
search.setOrganizedByType("Task");

SearchColumn searchColumn = objectFactory.createSearchColumn();
searchColumn.setType("Task");
searchColumn.setName("Subject");
ArrayOfSearchColumn arrayOfSearchColumn =
objectFactory.createArrayOfSearchColumn();
arrayOfSearchColumn.getSearchColumn().add(searchColumn);

searchColumn = objectFactory.createSearchColumn();
searchColumn.setType("Task");
searchColumn.setName("Id");
arrayOfSearchColumn.getSearchColumn().add(searchColumn);
search.setColumns(arrayOfSearchColumn);

SearchCondition searchCondition = new SearchCondition();
searchColumn = objectFactory.createSearchColumn();
searchColumn.setName("Status");
searchColumn.setType("Task");
searchCondition.setColumn(searchColumn);

searchCondition.setOperator("EQUALS");
searchCondition.setValue(objectFactory.createSearchConditionValue(TaskStatus.NEW
_OR RECEIVED));
ArrayOfSearchCondition arrayOfSearchCondition =
objectFactory.createArrayOfSearchCondition();
arrayOfSearchCondition.getSearchCondition().add(searchCondition);

search.setConditions(objectFactory.createSearchConditions(arrayOfSearchCondition
));

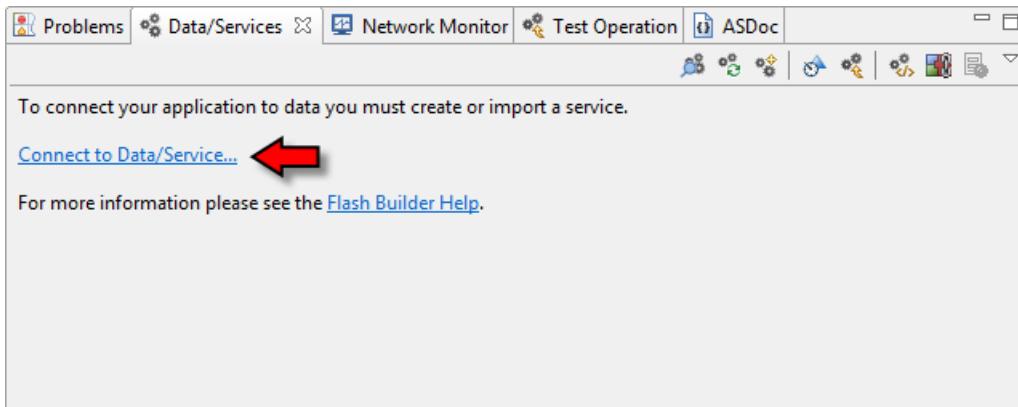
SearchResults searchResults = webAPI.executeSearch(search, null, null);

```

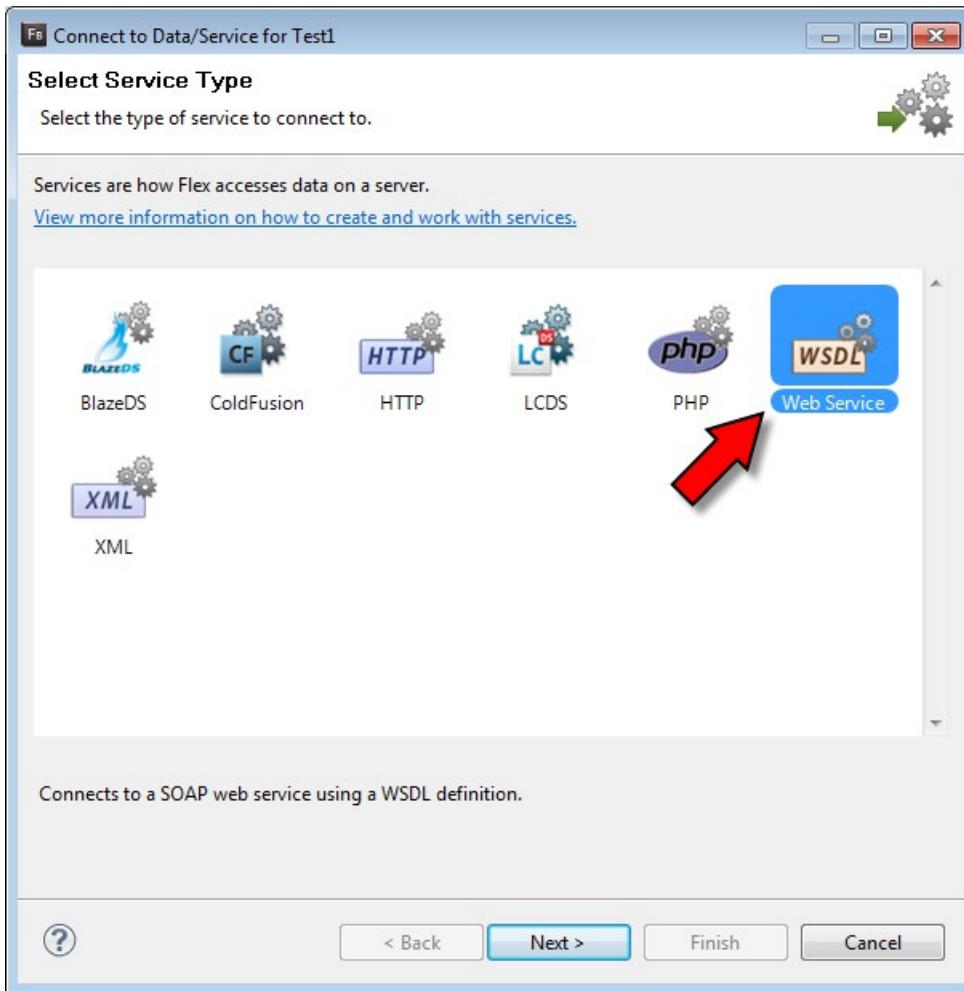
Building a Flex Client

Adobe Flex provides the ability to invoke Web Services from within a Flash application. The development environment for constructing Flex Applications is the Adobe Flash Builder (latest is v4). This IDE includes wizards that will build Web Service client code that can easily be called.

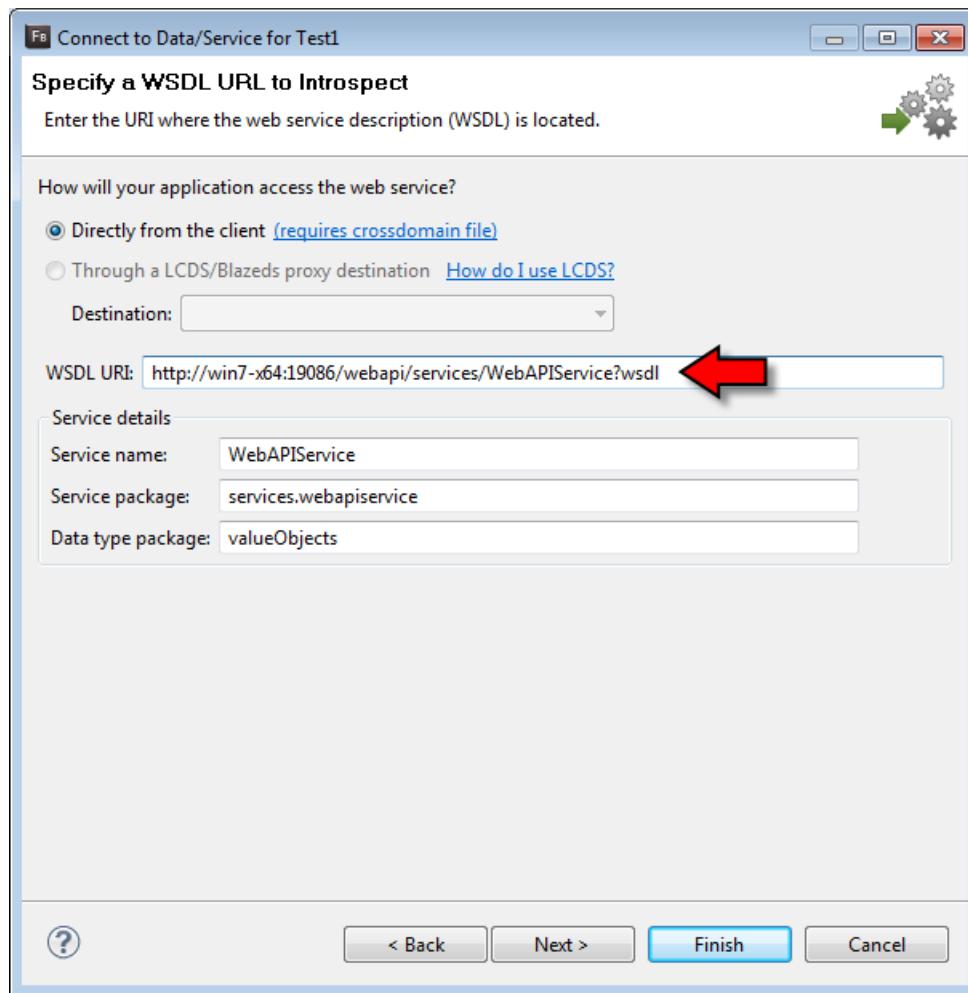
First we assume you have created a Flex Project which needs Web Service API access to IBPM. In the Data/Services tab click the Connect to Data/Service... link to start the wizard:



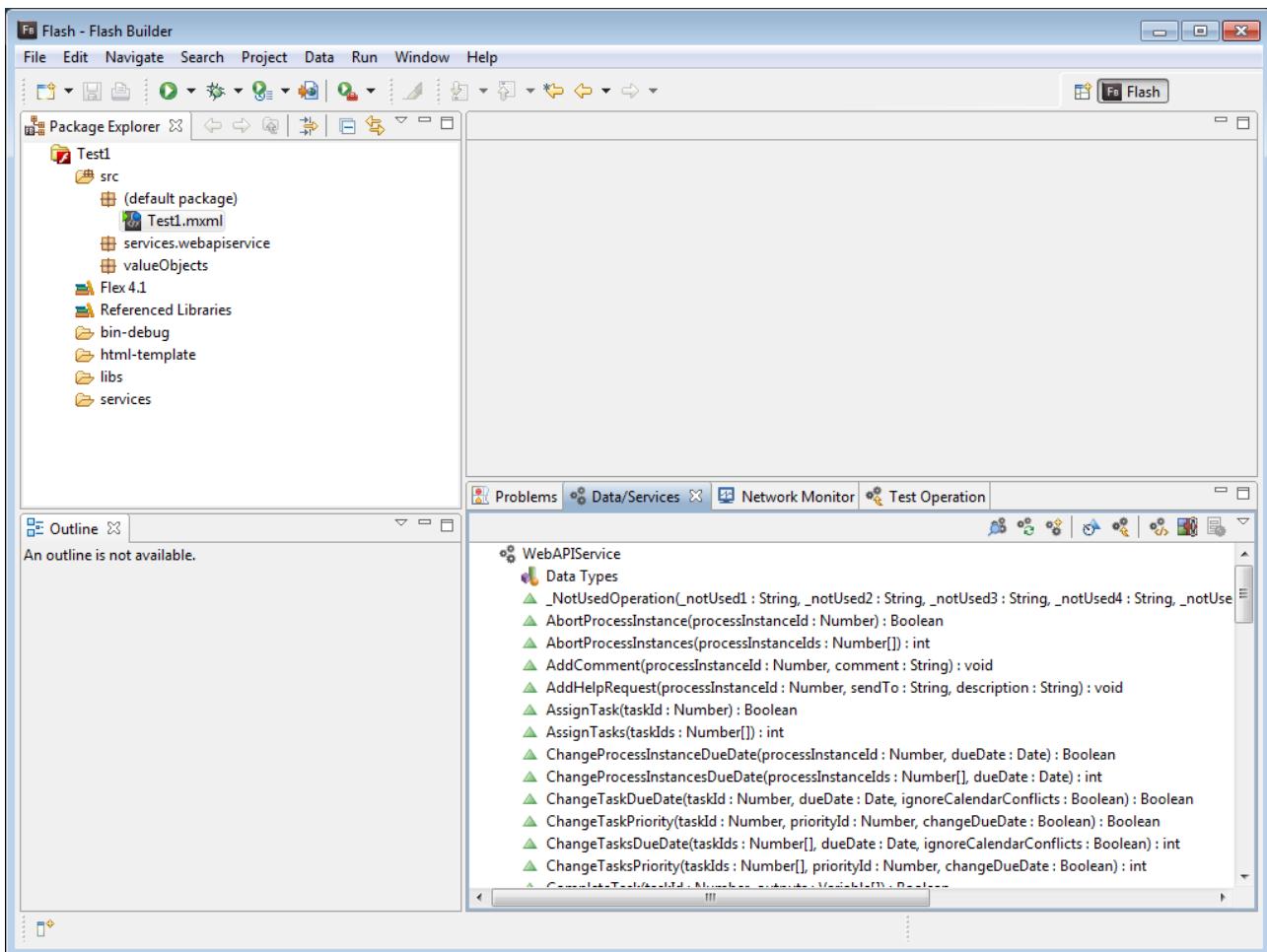
Next select Web Service as the type of service to build:



Next enter the URL from which the WSDL for the IBPM Web Service can be obtained:

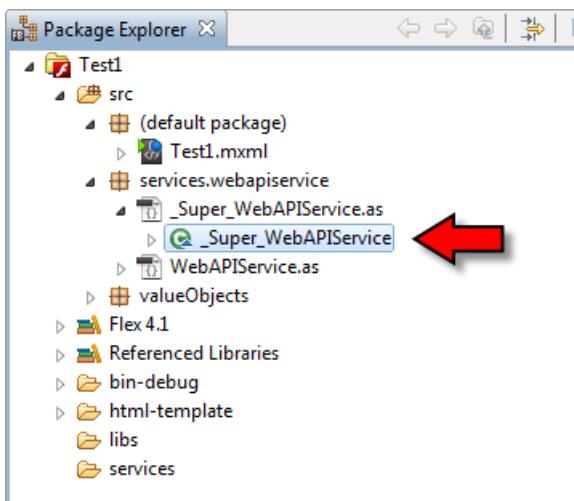


Once completed, new artifacts will have been generated:



The Web Service code generated does **not** know about the mandatory `ClientInfo` SOAP Header required to successfully make Web Service calls to IBPM. The `ClientInfo` structure has to be added manually. Fortunately, this is not a complex task.

Locate the generated source file called `_SuperWebAPIService`. This can be found in the `services.webapiservice` folder.



Open this file in the Action Script editor. Find the constructor function also called `_Super_WebAPIService`.

Add the following code fragment before the end of the function:

```
var q1:QName = new QName("http://webapi.lombardisoftware.com", "ClientInfo");
var header1:SOAPHeader = new SOAPHeader(q1,
    {"TimeZone": "CST",
     "Locale": "EN",
     "AcceptsTimeZone": false});
_serviceControl.addHeader(header1);
```

Here is an example:

```
324         _serviceControl.service = "WebAPIService";
325         _serviceControl.port = "WebAPISoap";
326
327
328     // Add
329     var q1:QName = new QName("http://webapi.lombardisoftware.com", "ClientInfo");
330     var header1:SOAPHeader = new SOAPHeader(q1,
331         {"TimeZone": "CST",
332          "Locale": "EN",
333          "AcceptsTimeZone": false});
334     _serviceControl.addHeader(header1);
335     // Add
336
337     wsdl = "http://win7-x64:19086/webapi/services/WebAPIService?wsdl";
338     model_internal::loadWSDLIfNecessary();
339
340
341     model_internal::initialize();
342 }
--
```

Save the changes and you have completed the task. If you should ever have to regenerate the generated code, you must repeat these steps but fortunately, that should not have to be performed unless you are upgrading from one release of IBPM to another and need Web Service operations found in the new release that weren't present in the older release.

We have now completed the preparation of the environment and can leverage the Web Service calls. Here is an example of a button press callback that executes a search:

```
protected function button1_clickHandler(event:MouseEvent):void
{
    var search:Search;
    search = new Search();
    search.OrganizedByType = "Task";
    var searchColumn:SearchColumn;
    var searchColumns:ArrayCollection = new ArrayCollection();
    var searchCondition:SearchCondition = new SearchCondition();
    var searchConditions:ArrayCollection = new ArrayCollection();

    searchColumn = new SearchColumn();
    searchColumn.Type = "Task";
    searchColumn.Name = "Subject";
    searchColumns.addItem(searchColumn);

    searchColumn = new SearchColumn();
    searchColumn.Type = "Task";
    searchColumn.Name = "Id";
    searchColumns.addItem(searchColumn);

    search.Columns = searchColumns;
    search.Conditions = searchConditions;

    executeSearch(search, 1000, 0);
}
```

Using a technique like this would allow us to build our own Flex based portal that could execute in the context of a Flash Player running in a browser or in a stand alone application running in Adobe AIR.

System Data Toolkit

The System Data Toolkit is a toolkit supplied by default with an IBPM installation. The artifacts contained within are assured to be present in all IBPM environments. This toolkit is automatically associated with all Process Applications. Importantly, since these services are provided with the product, they are fully supported by IBM.

IBM Supplied Business Process Definitions

Business Process Definitions are BPDs that are included in the System Data Toolkit.

Send SLA Violation Email

IBM Supplied General System Services

General System Services are services that are supplied with the System Data Toolkit. They can be referenced anywhere that a General System Service is needed.

Default BPD Event

Default System Service

Extract XML Validation Results

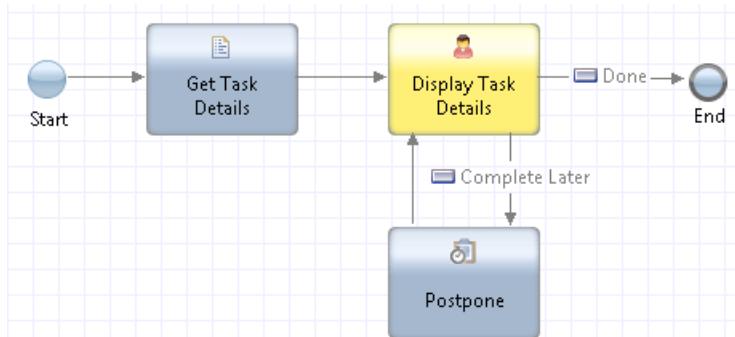
IBM Supplied Historical Analysis Scenarios

IBM Supplied Human Services

Human Services are services that are supplied with the System Data Toolkit that can be used in solutions anywhere a Human Service is needed.

Default Human Service

When an activity is added to a BPD, it is initially associated with a Human Service called Default Human Service found in the System Data Toolkit. The details of this task look as follows:



When shown to the user, it looks like:

Default Service

Default Activity

Activity Information	Task Information
Name: HT1	Subject: Step: HT1
Instance Name: Human Task Timeout:253	Assigned To: tw_admin
Process: Human Task Timeout	taskId: 2078.303
	Instance Id: 2072.253

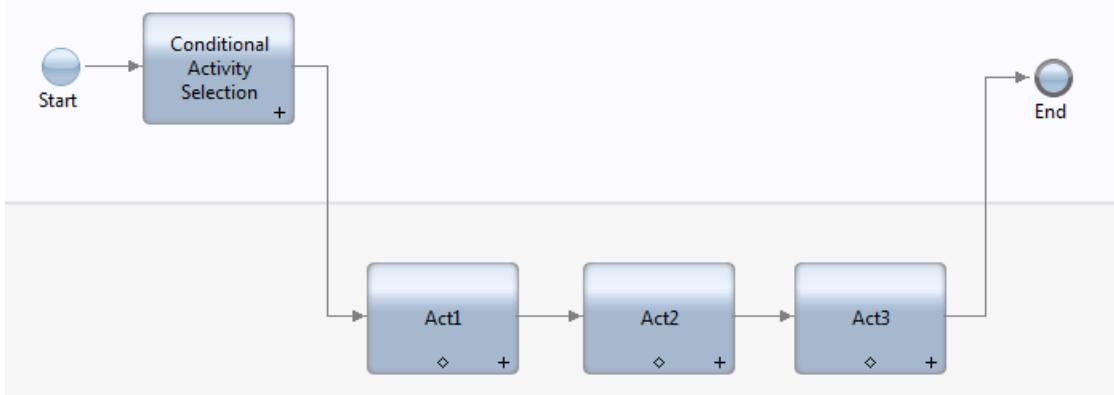
Done
Complete Later

The Coach shows some of the primary details of the Task/Activity that is currently running.

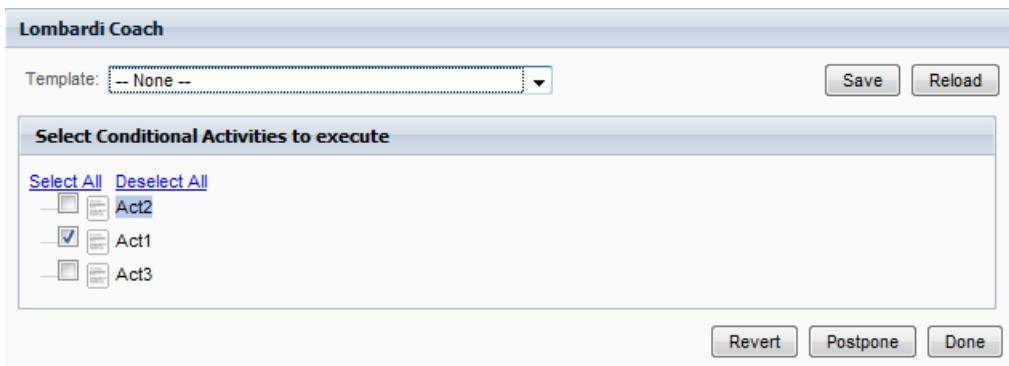
Isw Conditional Activity Selection Coach

The purpose of this Human Service is to present the end user with a tree structured interface that describes which activities can be skipped and which must be executed. In IBPM, activities can be flagged as conditional which means that they need not execute. This Human Service interrogates the process definition and determines the list of activities which are flagged as conditional. For each of these activities a check box marker is show beside the name of the activity. The user can then select which activities to execute and which to skip.

For example, look at the following BPD:



Here we see three conditional activities. The first Activity in the BPD as a whole is an invocation of the Isw Conditional Activity Selection Coach Human Service. The visual appearance of this will look as follows:



The Templates allow us to save selections of which activities are selected and which aren't so that we can re-use those templates in the future.

Inputs

Name	Data Type	Suggested value
bpdGUID	String	tw.system.currentProcess.guid
conditionalActivities	List of ConditionalActivity	tw.system.currentProcess.conditionalActivities
selectedConditionalActivities	List of String	tw.system.currentProcessInstance.selectedConditionalActivities
templateID	String	(default)
isReadOnly	Boolean	(default)

Outputs

Name	Data Type	Suggested value
selectedConditionalActivities	List of String	tw.system.currentProcessInstance.selectedConditionalActivities
templateID	String	(None)

See also:

- Conditional Activities

Fire Default BPD Event

IBM Supplied Integration Services

Call SPA BAPI-RFC

Call WebService via SOAP

Email Get System Default Properties

Email Send Teamworks Email

Read E-mail via IMAP

Read E-mail via POP

Read from HTTP

Read Text File

This service reads text from a named file. The parameters are:

- ↳ Input
 - ↳ filePath (String)
- ↳ Output
 - ↳ fileContents (String)

Parameter	Description
filePath	The full path to the file who's contents are to be read and returned.
fileContents	The data returned from reading the file.

See also:

- Write Text File

Send E-mail via SMTP

This service is described in detail in this book at the following section: eMail.

SLA Send Alert Email

SQL*

This is a set of database interaction services which can be used to retrieve, insert, update or delete data from a database. For usage, see Database Integration.

Transform XML Document using XSL Document

Transform XML Document using XSL File

Transform XML file using XSL file

Update ALL SLA Statuses

This service has no input or output parameters.

Update SLA Status

Validate XML File

Validate XML from URL

Validate XML String

Write Text File

This service writes data to a file on the local file system on which Process Server is running. The file can either be overwritten or appended to. The parameters are:

- ↳ Input
 - ↳ filePath (String)
 - ↳ fileContents (String)
 - ↳ appendToFile (Boolean)
- ↳ Output
 - ↳ success (Boolean)

Parameter	Description
-----------	-------------

filePath	The full path to the text file that will be created or appended. Take care to use forward slashes on windows.
fileContents	The data that is to be written into the file.
appendToFile	A value of true means that the file is appended. A value of false means that the file is created or replaced.
success	A boolean indicator saying whether the file IO succeeded or failed.

See also:

- [Read Text File](#)

IBM Supplied KPIs

IBM Supplied Layouts

[Report Filter – Simple Number Replacement](#)

[Report Filter – Time Grouping Range](#)

[Report Filter – Hidden User Filter](#)

IBM Supplied Participant Groups

IBM Supplied Server Files

Integration.jar

This is the implementation file for much of the toolkits functions written in Java.

IBM Supplied Tracking Groups

IBM Supplied Undercover Agents

IBM Supplied User Attribute Definitions

[Alert on Assign and Run](#)

[Image](#)

[Locale](#)

[Primary Role](#)

[Task Email Address](#)

[Task Notification](#)

IBM Supplied Variable Types

Cookbook Scenarios

In this section of the book we will illustrate some cookbook scenarios. These are not meant to be full applications but rather to illustrate how a technique or concept can be achieved.

Scenario - Asynchronously starting one process from another

In this scenario we wish to have one process asynchronously start another process. Consider a process called "Process A". During its execution, we determine that we wish to start an instance of "Process B" but have "Process A" continue in parallel.

We will use the concept of a UCA to achieve this.

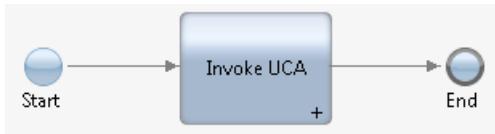
Process B looks as follows:



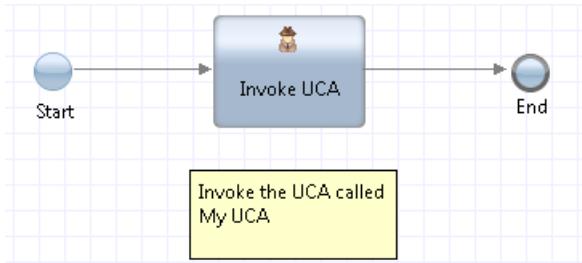
The interesting part of the process is that it is started by a "Start Message Event" step. This means that event has to arrive from an external source in order for the process to begin.

The configuration of this Step has as name a UCA which we called "My UCA".

Now let us look at the details of "Process A".

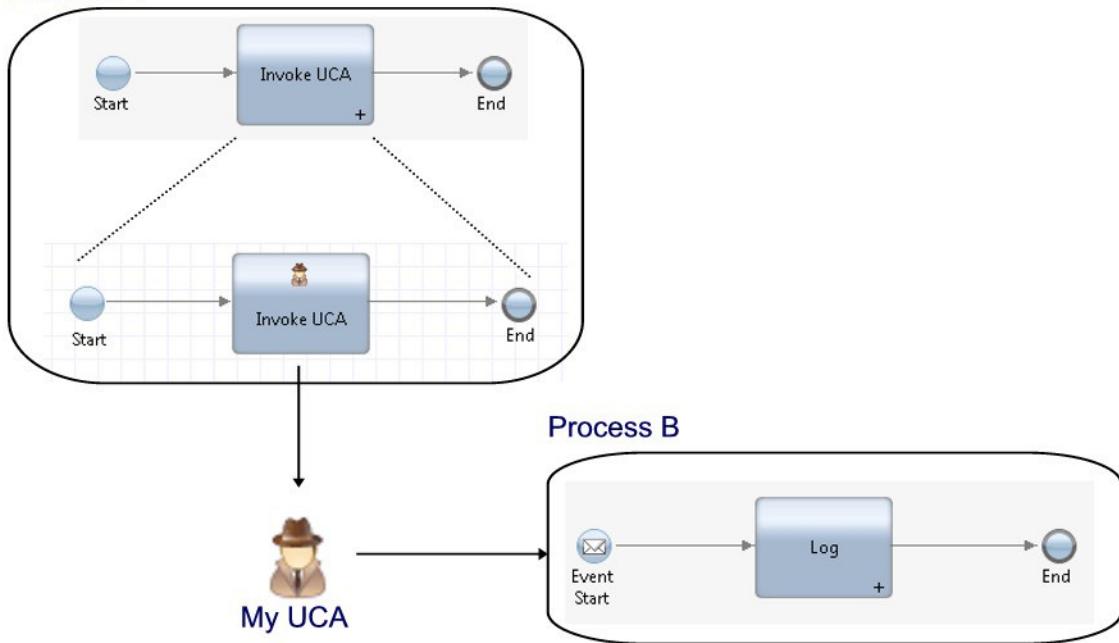


This contains an activity that is implemented as a General System Service.



The primary step in this service is an Invoke UCA step. This step names "My UCA" as the UCA instance to be invoked.

Process A



An alternative to this technique is to use the JavaScript API.

Scenario – Making a process extensible by others

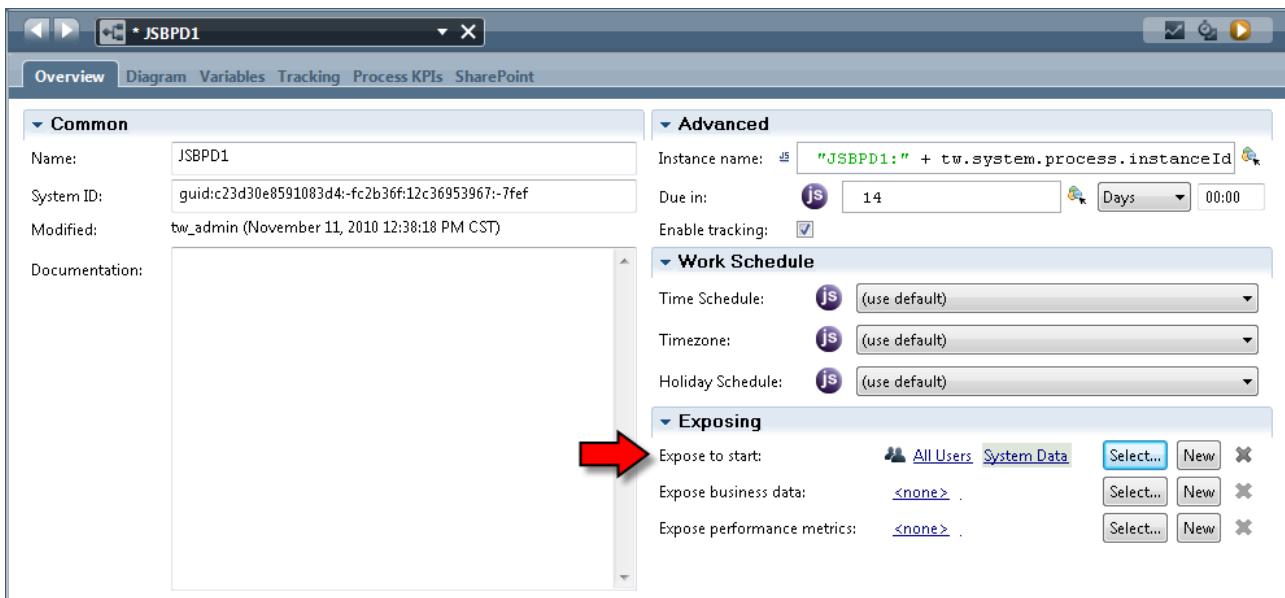
A customer (lets call them XYZ Corp) of IBPM wanted to build business processes that they sell to their own customers and they wanted to allow their customers to make minor customizations. They faced two problems. The first was that they didn't want the end customers to break the integrity of the business processes they were selling. The second problem was that the original processes were to be updated over time with new revisions and they didn't want to "obliterate" the customizations made by the end customers if the BPDs were replaced with new copies.

One way to achieve this task is to have the core processes make calls to nested BPDs. These nested BPDs become "exit points" in the core process. If an end customer wants to customize the core process, they would customize the sub-processes invoked by the core BPDs. If these sub-BPDs are kept in a separate toolkit from the core Process Application then a replace of the Process Application with a new version will result in the original Toolkits BPDs being called and that is where the customizations would take place. This solves both problems. The core processes become effectively read-only and the sub-processes called by the core become documented "exit points" in the core process. If the core processes are changed and the calls to the exit points left as they were then the sub processes will still be called at the correct times. When the new Process Application version is installed, as long as the Toolkit containing the sub processes is not itself replaced, the core process will continue to call the sub processes as they were before.

Scenario – Starting a process from an external browser

In previous releases of IBPM, there used to be a concept of *favorites* when it comes to starting processes. A process could be defined as a favorite and then started from a favorite list in Process Portal. This concept is now all but gone from the product but part of the technology still remains.

In Process Portal there is a list of processes that can be started. This list is the set of processes that have had their "Expose to start" entry defined in the BPD editor in the Overview section:



The access to this list of startable processes within the Process Portal is from the New button.
When one right click on an entry in this list, the URL link location (where the web link would take you) is available to be copied:

See:

Task ID	Task Subject	Task Priority	Task Due Date	Run
PT1 BPD1:305	Obtain email account for	Normal	10/27/10 10:18 AM	
PT1 BPD1:306	Step: Task 1	High	11/05/10 12:01 PM	
PT1 BPD1:307	Step: Task 1	High	11/05/10 12:04 PM	
PT1 BPD1:308	Step: Task 1	High	11/05/10 12:50 PM	
PT1 BPD1:309	Step: Task 1	Low	11/05/10 5:28 PM	
PT1 BPD1:310	Task HS2	Normal	11/05/10 10:45 PM	
JSBPD1:360	Step: Task	Normal	11/11/10 1:38 PM	

If the result of this link location is pasted into notepad, an entry similar to the following will be found:

```
javascript:doFavorite(407);
```

Note the number in parenthesis. Let us call this the "favorite id".

If a web browser now visits:

```
http://server:port/teamworks/process.lsw?
zObject=Favorite&zActivity=Execute&zPost=1&zFavId=favoriteId
```

Where:

- server – hostname or IP address of the IBPM server
- port – Port number on which IBPM is listening

- `favoriteId` – the numeric Favorite ID found by the recipe above

An instance of the process will be started and the first coach (if configured) will be displayed.

It is not known if this recipe is formally defined as part of the architecture of IBPM. No reference to this technique has been found in the product manuals and may not be supported today or work in the future.

Additional Toolkits

The architecture of IBPM lends itself very well to the re-use of logic and code. Through the concept of Toolkits, function can be added to a project very easily. As such, there are many Toolkits out there that can be freely used. However, without documentation of what these toolkits can do for you, they are useless. As the author works with IBPM, he finds commonly used functions and has built his own set of toolkits. These are freely available with complete source code where applicable.

Kolban JavaScript Toolkit

This is the repository of JavaScript functions the author finds useful.

function: kolban.date.add

We add a time delta onto an input TWDate and return a new TWDate. If the time delta is negative, then we subtract time.

Examples:

```
// Subtract one day  
var newDate = kolban.date.add(oldDate, -1, java.util.Calendar.DATE);
```

originalDate:TWDate	An instance of TWDate that is the base date that will have time added to it
delta:Integer	The amount of time to be added to the date. May be negative to subtract time
type:Integer	The java.util.Calendar constant for the type of time to be added. This is likely to be one off: <ul style="list-style-type: none">• java.util.Calendar.YEAR• java.util.Calendar.MONTH• java.util.Calendar.DATE - Date of month• java.util.Calendar.HOUR• java.util.Calendar.MINUTE• java.util.Calendar.SECOND

returns A new TWDate that represents the new time.

function: kolban.task.getTaskByActivity

Return the task as a TWTTask object of the task associated with the named activity. If there is no task associated with that activity, null is returned. If there are multiple tasks associated with the activity, one is returned but which one is not defined.

Examples:

```
var myTask = kolban.task.getTaskByActivity("MyActivity");
```

activityName:String	The name of a BPD activity for which the task is to be retrieved.
---------------------	---

Returns the TWTTask object associated with the activity. If no task is associated with the activity, null is returned.

function: kolban.debug.documentElementToString

Return a string representation of the passed in TWDocument

Examples:

```
var myString = kolban.debug.documentElementToString(myDocument);  
log.info("Doc is: " + myString);
```

document:TWDocument	The document to be returned as a string.
---------------------	--

Returns a string representation of the TWDocument.

Bluemworks Live

Bluemworks Live is a process discovery and modeling tool that lives on the web. It is a Software As A Service (SaaS) package. This means that users don't install anything specific on their desktops, instead they navigate through a Web Browser to the Bluemworks Live web site and the functions run inside the browser.

The home web page for Bluemworks Live is:

<https://www.bluemworkslive.com/sLogin.html?redir=%2Fscr%2Fhome>

You **must** have an account there in order to use it.

On a historical note, Bluemworks Live used to be called Blueprint

The home page provides a login screen against which you login.

The screenshot shows the Bluemworks Live login interface. At the top, a dark blue header bar displays the "BluemworksLive" logo. Below this is a white login form. On the left side of the form, the text "Log in to Bluemworks Live" is centered. To the right of this text are two input fields: one for "Email Address" containing "kolban@us.ibm.com" and another for "Password" showing five masked dots. Below these fields is a checkbox labeled "Remember my email address on this computer." To the right of the password field is a blue "Log In" button. To the far right of the login area, a sidebar titled "Latest News..." lists four recent updates, each preceded by a small circular icon. The news items include: "Welcome to Bluemworks Live! A new destination for doing your job.", "Bluemworks live is now available for existing Blueprint customers we encourage you to check out [Announcements](#)", "Bluemworks Live is now available! Learn more [here](#)", and "Join Phil Gilbert on 11/23 at 12 ET as he introduces Bluemworks Live on the [Zero to Process in 90 seconds webinar](#)". At the bottom of the sidebar, a link "View more..." is visible. Below the login form, three links are provided: "Don't have an account? [Sign up now >>](#)", "Trouble logging in? [Find help here >>](#)", and "New to Bluemworks Live? [Learn More >>](#)". The footer of the page features the IBM logo and a copyright notice: "Copyright © 2010 IBM Corporation. All Rights Reserved. [Privacy](#) | [Security](#)".

After logging in, you will be taken into the primary tool itself:

The screenshot shows the BlueworksLive interface. At the top, there's a navigation bar with links for Work, Community, Library, and a search bar. Below the navigation, there are tabs for Tasks assigned to me, Work I've launched, and Items I'm following. A sidebar on the right contains buttons for Blueprint a Process and Automate a Process. The main content area displays a list of spaces, each with a thumbnail, name, dropdown menu, star rating, and last modified date. The spaces listed are: Conference room scheduling project, Education Classes, Onboarding, Payroll Processing, and Sandbox. At the bottom, there are links for IBM, Terms of Use, and Invite New Users.

Neil Kolban - Help Logout

search

Tasks assigned to me Work I've launched Items I'm following

Processes Spaces Activities Values Work

Conference room scheduling project ★
Last modified by Neil Kolban on 2010/11/18 10:08:43
Recent Changes

Education Classes ★
Last modified by Neil Kolban on 2010/10/06 14:39:53

Onboarding ★
Last modified by Neil Kolban on 2010/11/03 17:38:58

Payroll Processing ★
Last modified by Neil Kolban on 2010/11/15 10:31:35

Sandbox ★
Last modified by Neil Kolban on 2010/11/15 11:34:29

ACTIONS

Filter DWTest Pruebas 1

IBM Terms of Use Invite New Users

The high level purpose of Blueworks Live is to allow staff members to capture their existing processes and potential changes to those processes. Where IBPM is concerned with capturing processes with the intent of final execution, Blueworks Live focuses on the discussion and discovery aspects of process based solutions.

Creating a Space

Within Blueworks Live, a *Space* is a related set of artifacts associated with a particular process or business goal. From the home page of your Blueworks Live account, you can create a new space:

BlueworksLive Work Community Library Neil Kolban - Help Logout search spaces

Spaces Processes Blogs Templates **Create New Space**

Active Spaces (265) Active ▾

Academic Internship NEW Last modified by Martin Triska on 2010/08/31 05:08:25	Accounting project NEW Last modified by Marek Zajac on 2010/10/21 01:35:21
AccServices NEW Last modified by Bruno Rodrigues on 2010/07/26 09:29:10	ACM insurance NEW ☆ Last modified by Petr Kocura on 2010/11/11 05:50:33 Recent Changes
Acme Pharmaceutical NEW Last modified by Dave Marquard on 2010/02/24 13:32:29	AIM NEW Last modified by Dwaipayan on 2010/09/14 04:35:16
Alan's Demo Project NEW Last modified by Alan Lim on 2010/06/02 21:04:53	Andrej-Procesi NEW Last modified by Andrej Porenta on 2010/09/17 06:27:50
AOC_BPM NEW Last modified by Yew Cheng Cai on 2010/10/12 00:32:10	Ashutosh Test NEW Last modified by ASHUTOSH JOSHI on 2010/06/14 00:21:03
August 6 2010 NEW Last modified by Amit on 2010/08/06 08:49:24	Avery MiniJam Project NEW Last modified by Dave Wakeman on 2010/07/28 13:40:48

IBM Terms of Use Invite New Users

Creating a Process

A process can be created within a space.

BlueworksLive Work Community Library Neil Kolban - Help Logout search processes

Spaces **Processes** Blogs Templates **Blueprint a Process** **Automate a Process**

Active Processes (769) Active ▾

02 Marketing NEW 1 Last modified by Federico Senese on 2010/07/27 16:43:32	03 Komunikace se zájemci NEW Last modified by Blueprint Support on 2010/07/15 08:48:31
04_Realizace býr.ríz. 2.kolo NEW Last modified by Blueprint Support on 2010/07/15 08:48:31	1.5 Projektevaluation NEW
2.0 Exemptions, Extensions and Missin... NEW Last modified by Matt Khodadad on 2010/09/26 17:08:51	4.3.3 Order Materials and Services NEW Last modified by Raymond Novak on 2010/11/18 05:54:37
48 Hour Day Process NEW Last modified by Richard Kezys on 2010/06/28 18:10:03	48 Hour Day Process Proposed NEW Last modified by Richard Kezys on 2010/06/28 18:10:03
[HR] Hiring - Onboarding NEW Last modified by Lillian Wang on 2010/11/09 21:13:38	[HR] Hiring - Onboarding NEW 7 Last modified by Preston Romney on 2010/10/26 11:23:39
[HR] Hiring - Onboarding ACME NEW 5 Last modified by Marek Zajac on 2010/10/21 01:29:02	[HR] Hiring - Onboarding ACME NEW 5 Last modified by Nidhi Nijhawan on 2010/10/22 06:11:31

IBM Terms of Use Invite New Users

In the dialog, select a name for the process to be created in the project and then select the space into which the process will be added. An option allows us to create a new space if a suitable one does not already exist:

Blueprint a Process

Select a name for your new process. We suggest a name that describes the overall process or the types of activities that it will contain.

Process Name

My Process

By default, you will automatically follow all processes you create. You can change this in your user settings panel.

Next, select a space to place this process in.

Space Name

Create a new space...

My new Space name

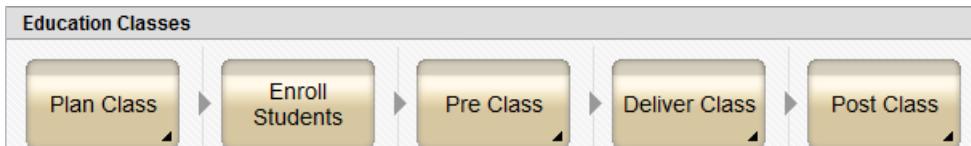
Select Create to start blueprinting your process.

Create

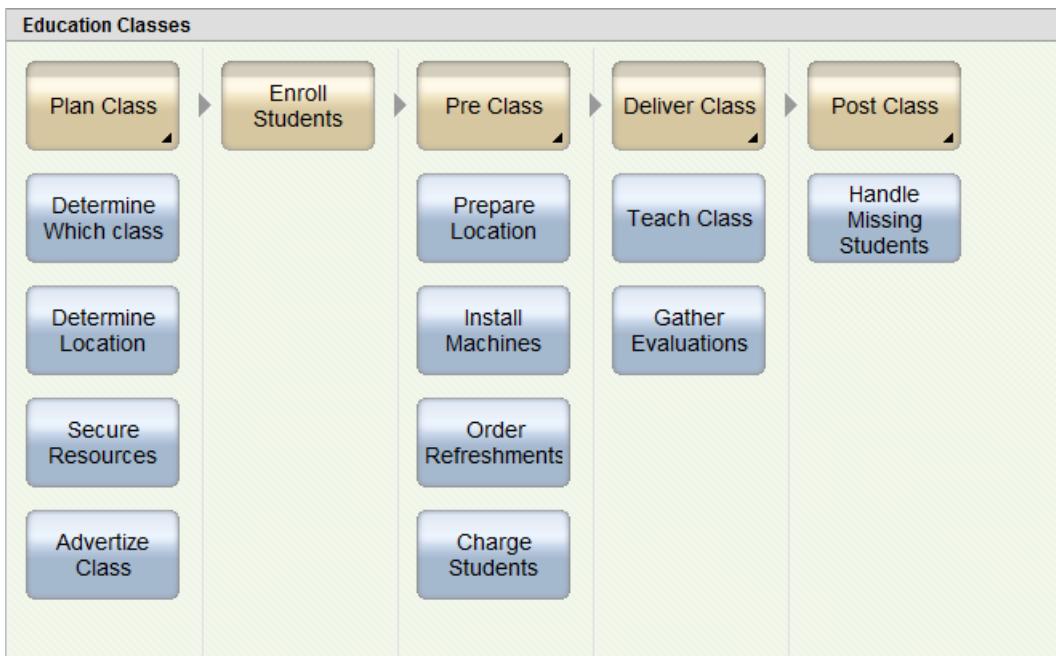
There is a proverb/saying that says "Unless you know where your destination is, you will never get there". This is especially true when building BPM based solutions. Whether you are implementing a Business Process for the first time or revising a currently existing process, you will want to be able to capture and discuss all the steps in that process before committing to implementation. Blaworks live, at its simplest level, provides a drawing canvas in which two different but related styles of process diagram can be drawn. The first is called a "Discovery Map".

Discovery Map

A Discovery Map contains a set of "Milestones" that are reached during the execution of a process. Capturing these milestones is the highest level in beginning to build out a process. There will probably be (on average) six to ten milestones in the diagram. More than that probably means that the process has been drilled down too low to start with.



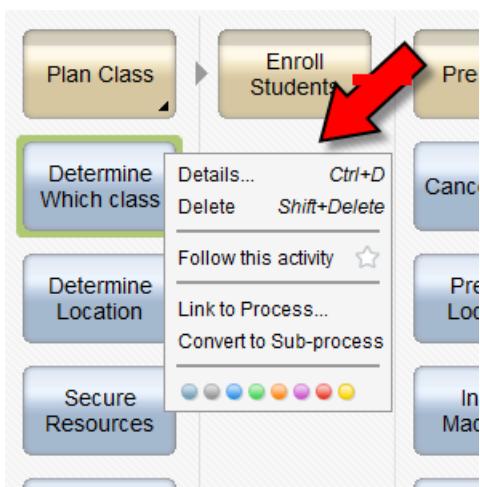
The milestones are deliberately light in any kind of refinement. After capturing the top level milestones, we can then look further to think about major activities that need to be accomplished to achieve those goals. We can then add these activities beneath each milestone.



There is no right or wrong answer for this. The Discovery Map is put together by conducting team and individual interviews. At this stage, the map is still extremely high level but as we can see, things are starting to take shape.

Each of the activities in the map can be supplied with additional attributes. How much or how little time you spend on these attributes is up to you. Remember, this diagram is not prescriptive, it is a powerful aid to help you discuss and capture the process.

Right clicking on an activity brings up its context menu. From here, we can select the Details option.



The Details option brings up a dialog into which information (or details) about that activity step can be entered.

Determine Which class

Details Problems Documentation Attachments Comments

- Participant: Education Owner
- Business Owner(s):
- Expert(s):
- System(s):
- Cycle Time: Minutes
- Cost:

It contains five tabs called Details, Problems, Documentation, Attachments and Comments.

In the Details tab, we can enter various items about this step in the process. These include:

Participant	This is one of the core attributes of a step. It defines the role of the person who is responsible or eligible to perform this task. Later on we will see that the Participant value entered here becomes translated to a Swim Lane in a BPMN diagram.
Business Owner(s)	One or more entries that describe the Business Owner of this step. This could be a role or department and should probably not be someone's name. Ask yourself the question "Who cares about how well this step performs"?
Expert(s)	One or more entries that name experts who know about this step. These may be folks who perform this task today.
System(s)	If this step interacts with any IT systems, this is where this interaction can be documented. One or more entries may be supplied.
Cycle Time	This is an estimate of the duration this step will take.
Cost	This is the cost of executing this step.
Supplier(s)	
Inputs	
Outputs	
Customer(s)	
Risk	
Value Add	

In the Problems Tab, we can view and enter the problems that are to be discussed and solved:

The screenshot shows a software application window titled "Milestone 1". At the top, there are tabs: "Details", "Problems" (which is highlighted in blue), "Documentation", "Attachments", and "Comments". Below the tabs, under the "Problems" section, it says "Problem(s) ①". There is a small thumbnail image of a document icon. To the right of the thumbnail are buttons for "Severity" (set to "Low") and "Frequency" (set to "Low"). There are also "+" and "-" buttons for filtering.

In the documentation tab we can add documentation on this step or milestone.

The screenshot shows the same "Milestone 1" window, but the "Documentation" tab is now selected. A rich text editor toolbar is visible at the top of the main content area. The text area contains the following text: "This is a description of this milestone|".

In the Attachments tab we can view and add attachments associated with this step or milestone.

Milestone 1 

Details Problems Documentation **Attachments** Comments

Attachment(s) 
There are currently no files attached.

Add

The comments tab allows others to add comments to this step or milestone.

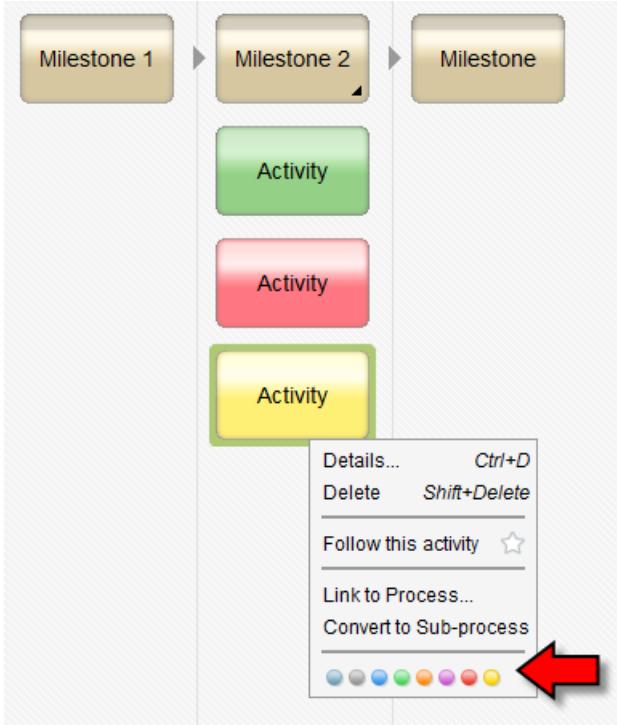
Milestone 1 

Details Problems Documentation Attachments **Comments**

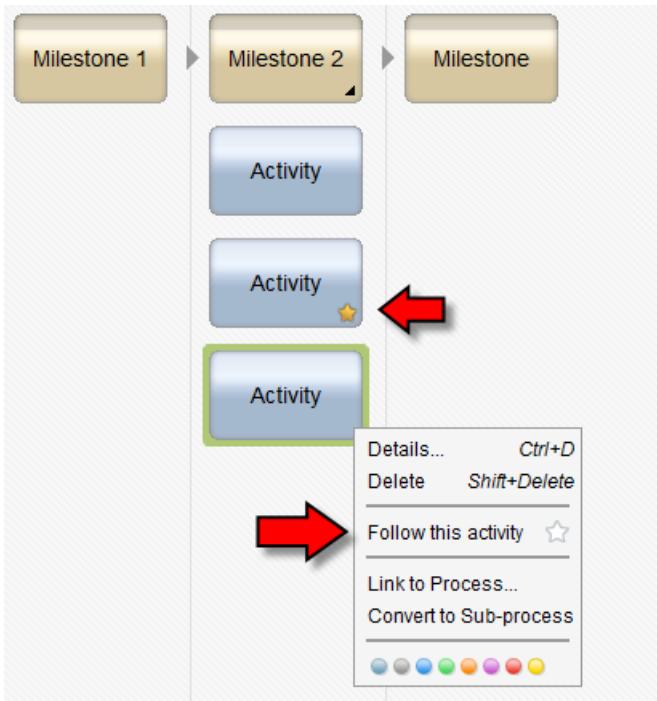
Add Comment

 Neil Kolban: This is a comment on this milestone.
2010/11/15 08:47:58 - [Reply](#) 

Within the Details pull-down, we can change the color coding of an activity. Changing the colors of activities has no semantic effect but can be used to quickly visualize different types of activities. For example, activities colored red could be expense generating while activities colored green could be used to indicate revenue generating.



We can flag activities as being followed which allows us to track changes to them in a special fashion. Activities flagged as followed have a star added to their visualization.



The followed activities then show up in the Following category under the followed Activities tab:

BlueworksLive Work Community Library Neil Kolban - Help Logout search

Tasks assigned to me Work I've launched ★ Items I'm following

Processes Spaces Activities Work **Activity 2 ★**

Activity in Sandbox Last modified by Neil Kolban on 2010/11/15 08:56:39

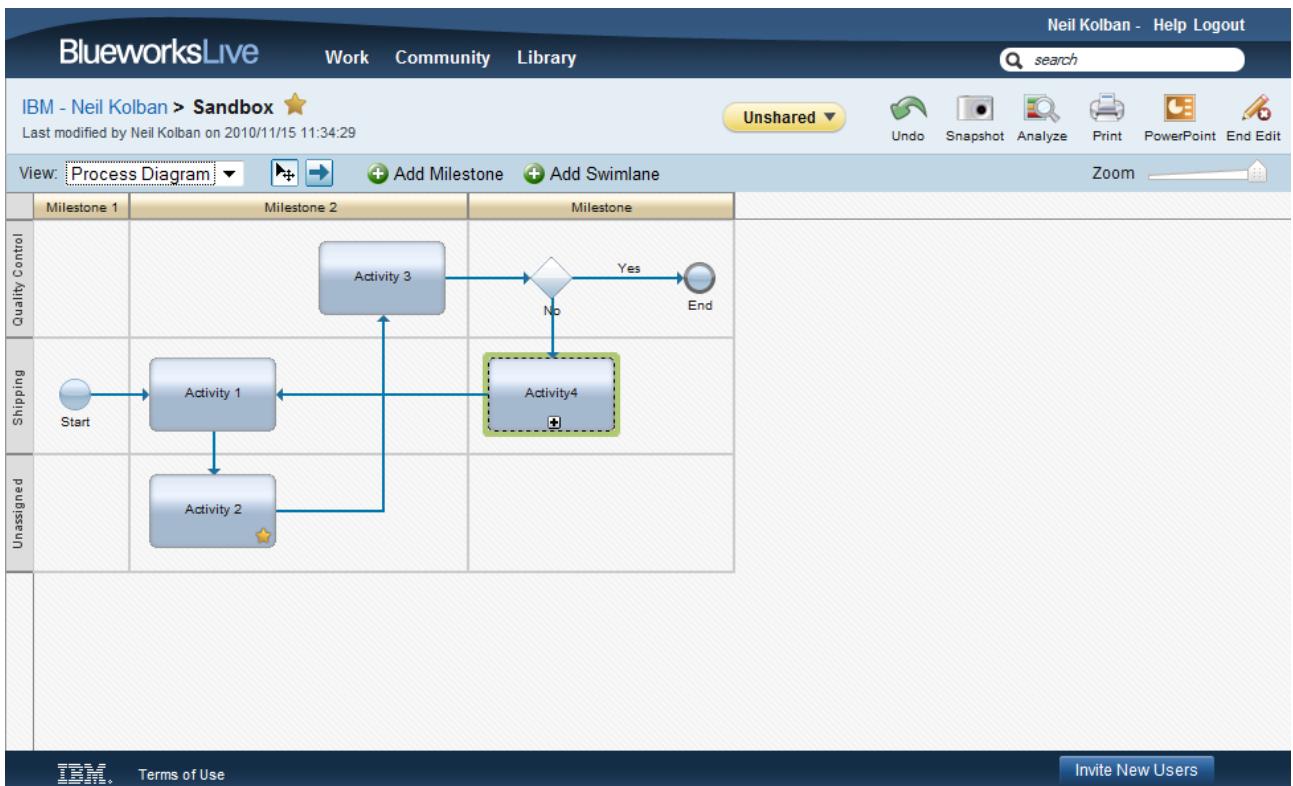
Blueprint a Process Automate a Process

ACTIONS Filter DWTest Pruebas 1

IBM Terms of Use Invite New Users

The Process Diagram

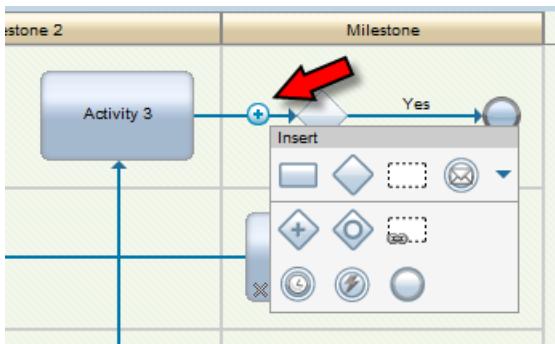
The Process Diagram is the partner to the Discovery Map. The Process Diagram shows a BPMN style diagram of the operation of the process.



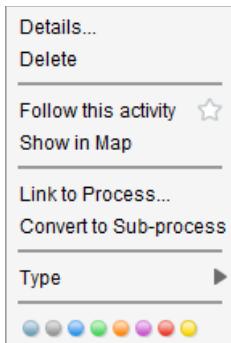
When a Process Diagram is created from a Discovery Map the milestones in the discovery map

become the milestones in the process and the activities in the Discovery Map become the activities in the Process Diagram. The swim lanes in the Process Diagram are taken from the Participants associated with the activities. Moving an activity from one swim lane to another changes that activity's Participant attribute.

Clicking on the wire link between activities produces a pop-up menu from which additional elements can be added:



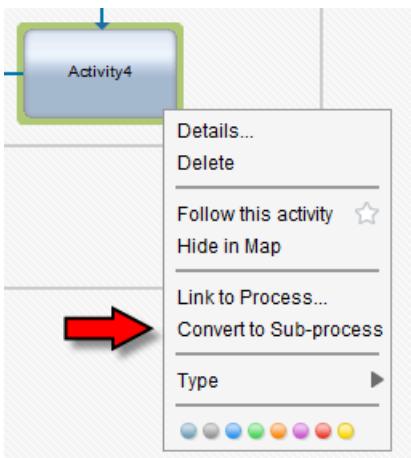
Clicking on an activity produces a details menu from which attributes can be changed. Many of these attributes are as previously described in the Discovery Map attributes.



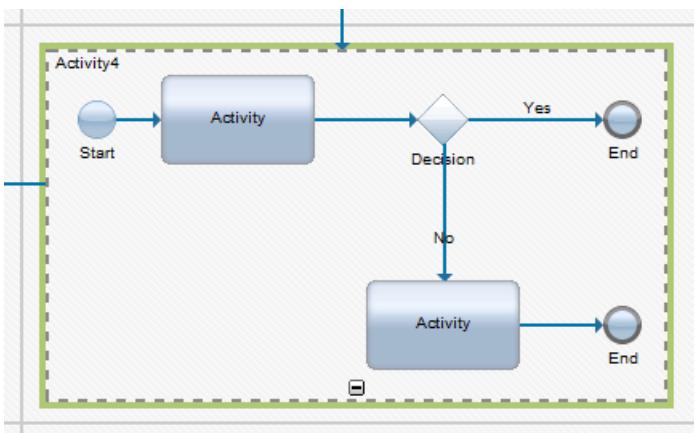
One particularly interesting attribute is the Show in Map attribute. By default, this attribute has a value of true meaning that the activity will also be seen in the Discovery Map. If changed to false, the activity will **not** be shown in the Discovery Map. This is particularly useful if the activity represents a technical aspect of the process as opposed to a business value step.

Working with Sub processes

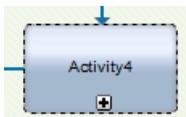
Selecting an activity and clicking Convert to Sub-process allows us to define that an activity is actually composed of a sequence of activities.



The sub process can then be embellished with additional decisions and activities.



The [-] icon at the bottom of the diagram allows the sub process to be collapsed back to its original container:



Notice that the container activity now has a [+] symbol indicating that it is a sub process that contains additional steps.

This sub process is currently not reusable. It is a sequence of steps contained solely within the containing process. A whole new process can be created from the sub process that can be reused in other processes. This creates a linked process.

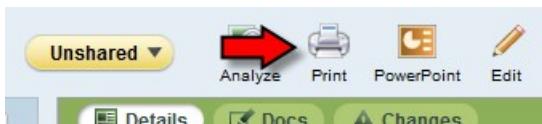
Editing mode vs Viewing mode

The Blueworks Live screen has two modes of view. Editing mode where the Discovery Map and Process Diagram can be changed and viewing Mode where the Discovery Map and Process Diagram are viewed with associated details.

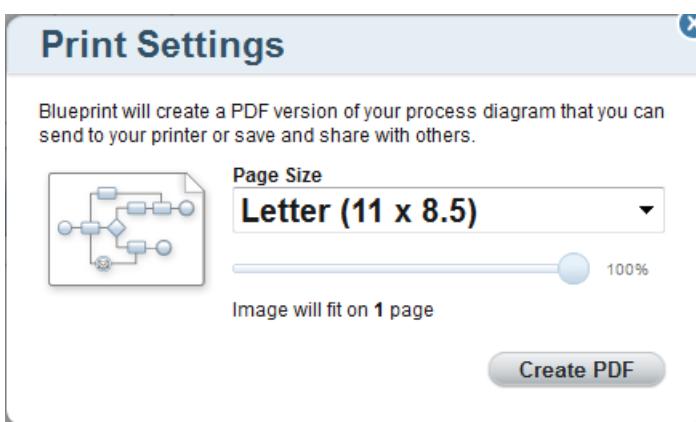
Here is an image of a Discovery Map in viewing mode. Notice that there is an area on the right in which the selected activity or milestone's details are displayed. The Edit button can be pressed to place the tool into editing mode.

Sharing the solution

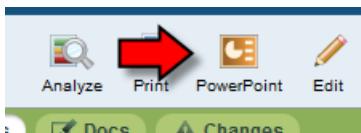
Both Discovery Maps and Process Diagrams can be printed by pressing the Print Button.



A dialog appears allowing the page size to be selected. At the conclusion a Create PDF button can be clicked which will render the image as a new PDF document.



A Microsoft Power Point presentation can also be generated by clicking the PowerPoint button.



A dialog appears asking where the Power Point should be saved.

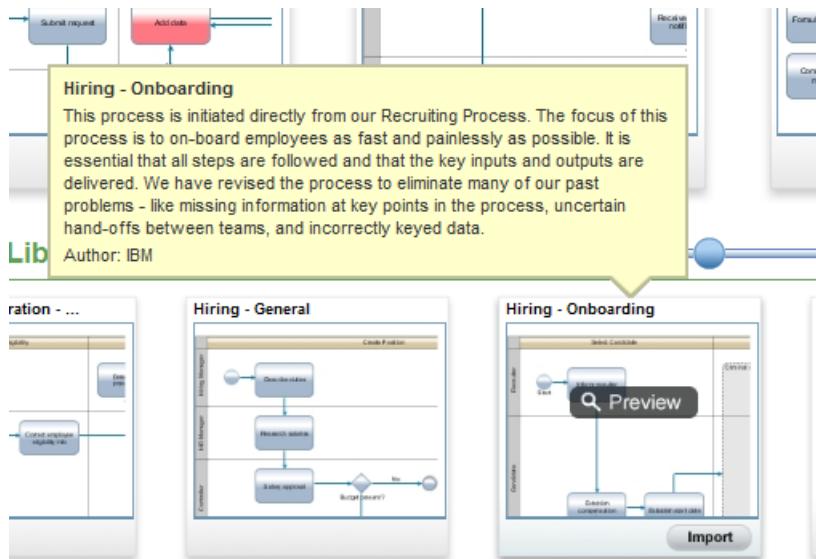
Creating a Process from a Template Library

Blueprint provides a set of process templates that can be used as starting points for your own processes. Selecting the Template Library link shows us a list of categorized process templates that can be selected.

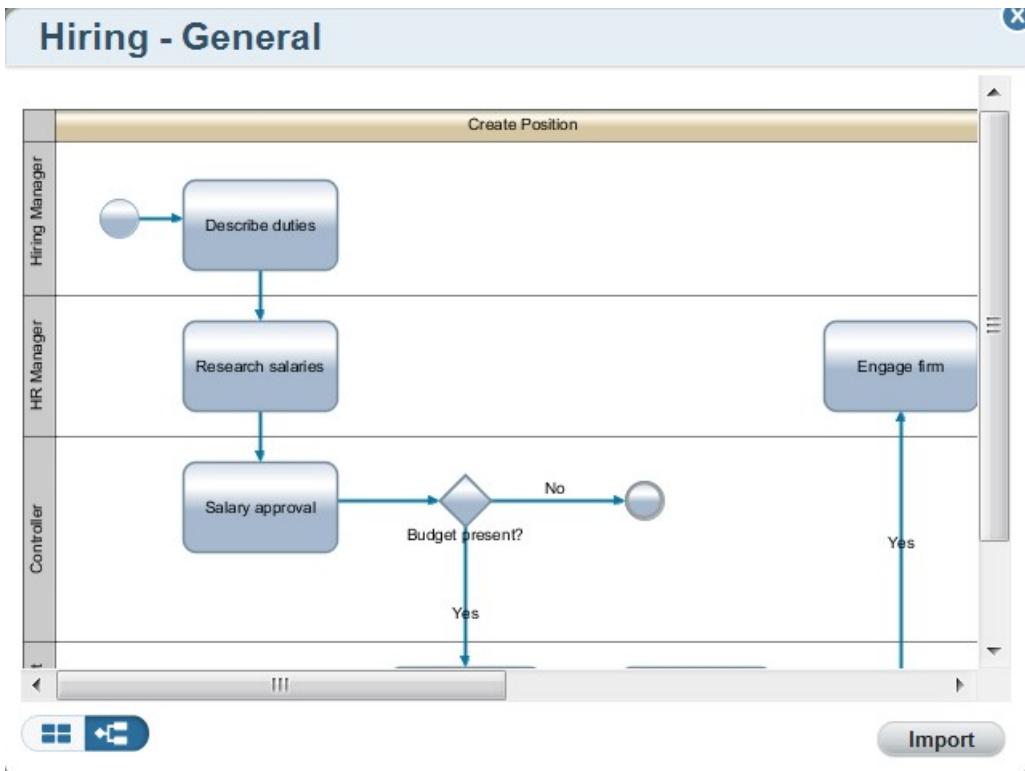
The screenshot shows the Blueprint software interface. At the top, there is a navigation bar with tabs: Work, Community, Library (highlighted with a red arrow), and Help Logout. Below the navigation bar, there are more tabs: Spaces, Processes, Blogs, and Templates (highlighted with a red arrow). A search bar labeled "search templates" is located at the top right. The main content area displays several process templates under the heading "Featured (10 Processes)". Each template is represented by a small diagram and a title: Accounting Services, Claims Payment, Develop business strategy, and Hiring - Onboarding. Below this, there is a larger section titled "Template Library (99 Processes)" containing four more templates: Accounting Services, APQC Classification - Manage Information, APQC Classification - Operate, and Benchmark performance. A "Preview" button is visible over the "Develop business strategy" template, and an "Import" button is located at the bottom right of the library section. At the bottom of the screen, there are links for "IBM Terms of Use" and "Invite New Users".

When a process template has been found which looks like a suitable candidate for forming the basis of your own process, a copy of the process can be imported into your own project.

Hovering the mouse over a process brings up a caption area in which a summary of the process can be found.



Clicking on the process picture brings up its preview/details. Both the Discovery Map and the Process Diagram can be viewed before importing.



When changes are made to a process or project, these changes are recorded in the News Feed. Think of this as a list of all changes that can be scanned like the headlines of a newspaper.

PRIVATE ACTIVITY STREAM

- I'm Following ★
- POPULAR TAGS: apqc, customer success, blueprinting, insurance, brms, getting started, telecom, bpm, banking, soa, btm, enterprise architecture, healthcare, tutorial, stephen white
- Date, User, Process, Space, Expand All, Collapse All, New Post
- Today**

 - David Whitehouse shared DWTest process in DavidWh space 5:39 PM
 - created DWTest process in new DavidWh space 5:35 PM
 - Angelo Sousa (TSI) imported Develop business strategy process template into TSI space 4:51 PM
 - created Teste process in TSI space 4:49 PM
 - created Teste process in new TSI space 4:41 PM
 - Genaro Nieto shared Pruebas 1 process in Genaro's Space space 4:03 PM
 - made 1 change to Pruebas 1 process in Genaro's Space space 4:03 PM
 - Vinicius Braganca created Teste process in Projetos Sao Paulo space 3:15 PM
 - Genaro Nieto created Pruebas 1 process in new Genaro's Space space 2:59 PM

- Yesterday**

PUBLIC BPM STREAM

- Today
- skemsley RED: OMG LOL FTW #moviereview 6:50 PM
- InnoSpotting Congrats to @omodica, @alexhoffat, @uihero, @jbasrai, @philgilbertsr and the rest of the non tweetering team at #bwlive! Great job! 6:38 PM
- BlueworksUpdate Welcome to Bluworks Live! A new destination for doing your job. #news 6:18 PM
- Bluworks Live Product Management posted a blog 6:08 PM
- Welcome to IBM Bluworks Live! We're very happy to announce the release of IBM Bluworks Live. This release includes some exciting new

Clicking on the News Feed link shows a summary of the changes made over the last period of time. These changes can be filtered by date, user who made the change, process in which the change was

made or project.

Automating a Process

From within the space where you want to create your automated process, click on the Automate a Process button:

The screenshot shows the BlueworksLive interface. At the top, there is a navigation bar with links for Work, Community, Library, and a search bar. Below the navigation bar, the page title is "IBM - Neil Kolban - Blueworks Live". Underneath the title, it says "last modified by Neil Kolban on 11/20/10 7:34PM". There are three tabs: Overview, Library (which is selected), and Users. The main content area is titled "Active Processes (0)" and contains the message "There are no active processes". To the right of this area are three buttons: "Blueprint a Process", "Automate a Process", and "Import from Visio". A large red arrow points to the "Automate a Process" button.

The screenshot shows the "Automate a Process" configuration dialog box. The title bar says "Automate a Process". The main text area says "Automation allows you to create a simple process app that can be launched by users to perform work items." Below this is a "Process App Name" input field containing "My Automated App". A note below the input field says "By default, you will automatically follow all processes you create. You can change this in your user settings panel." The next section, "Next, choose the type of process automation you want to create", has a dropdown menu set to "Simple Workflow". To the right of the dropdown is a small diagram of four circles connected by arrows forming a cycle. A tooltip for this option says "Allows you to define a sequence of tasks and approval steps that can be assigned to a set of users." At the bottom, there is a "Create" button.

TBD Describe the different templates

BlueworksLive Work Community Library Neil Kolban - Help Logout

IBM - Neil Kolban - Blueworks Live > My Automated App ★
Last modified by Neil Kolban on 2010/11/20 19:37:04

Configure Work Unshared ▾ Undo

Configure the Process App
Configure the process app information and details below. Once configured, the app can be shared as an action on the work page. Users can choose this action to launch the process app to create, assign and manage tasks.

This is the link a user will select to launch the process app. Select a name that accurately describes the process.

Action Label: My Automated App

Launch Instructions:

Work Subject:

Details:

Attachment(s): Add
There are currently no files attached.

Workflow Tasks

Order	Task description	Task assigned to	Approval step
1			<input type="checkbox"/> <input type="button" value="+"/> <input type="button" value="-"/>

Share with Participants

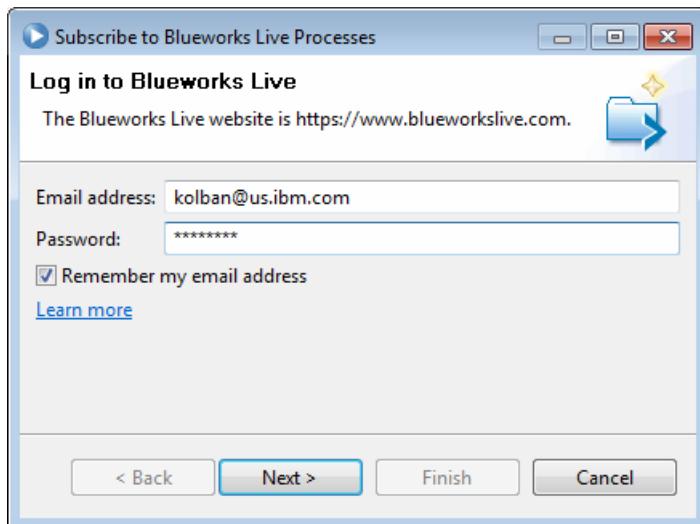
IBM Terms of Use Invite New Users

Accessing processes from Blueworks in Process Designer

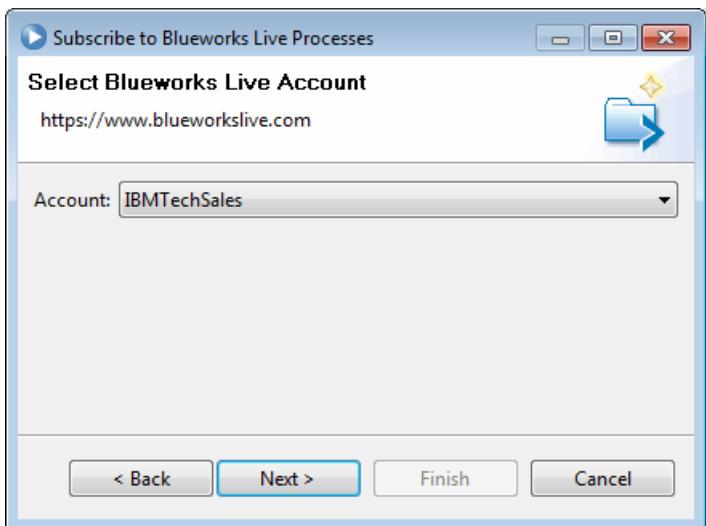
Processes captured and modeled in Blueworks can be directly imported into Process Designer. On the main Library area, there is an entry called BLUEWORKS LIVE PROCESSES. Clicking the plus button allows us to subscribe to a Blueworks process.



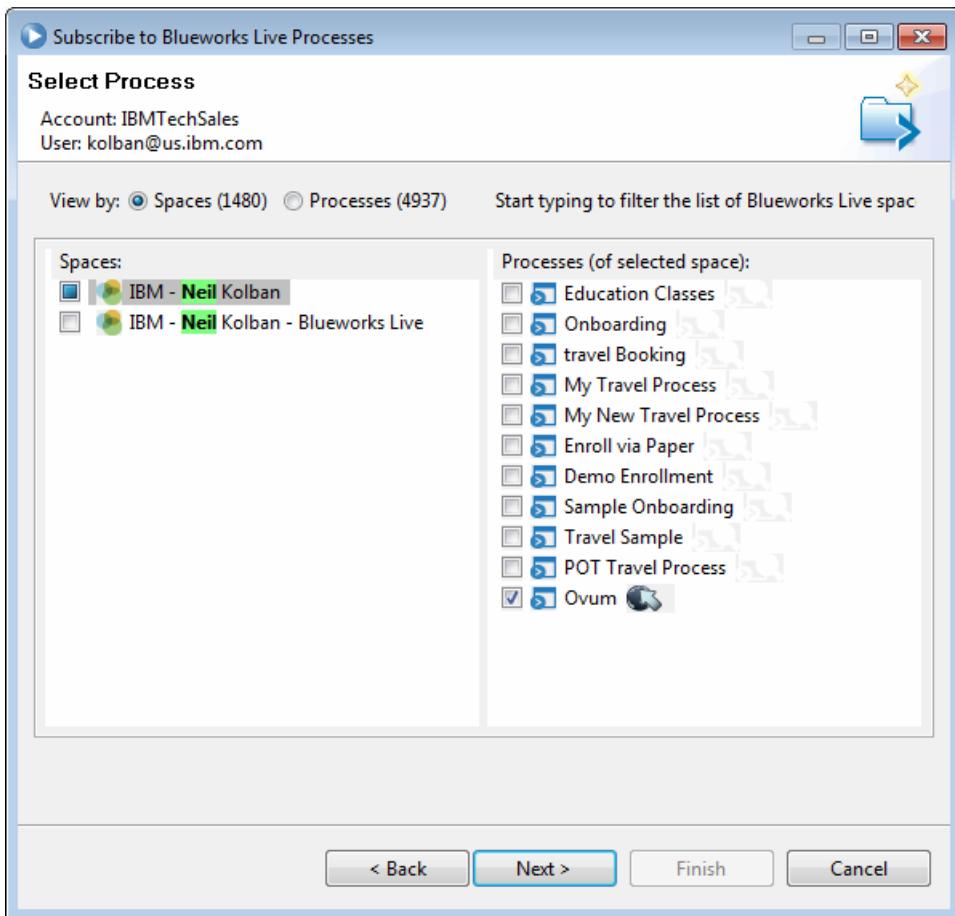
The first time this is navigated, you must enter account information to allow you access to the Blueworks server.



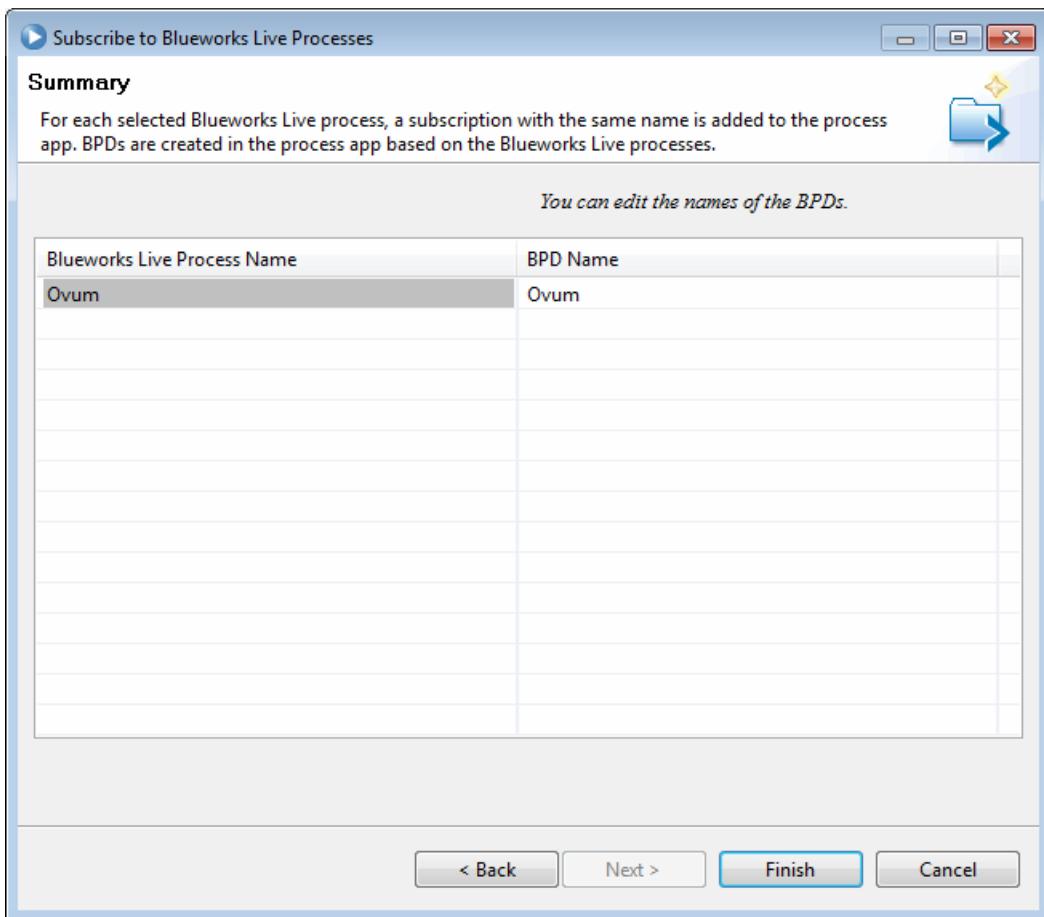
Once connection with Blueworks has been established, the account for the userid is allowed to be selected. It is possible a single userid may have multiple accounts:



a list of the projects and processes associated with the account are displayed. From this list, one or more projects can be selected for subscription.



A confirmation summary window is shown in which you can review the projects that are about to be subscribed to.



After a subscription has been made, you will now find a new set of artifacts in your Process Application that have come from the Blueworks maintained process.

TOOLKITS

- System Data (8.5.0)
- Coaches (8.5.0)
- Dashboards (8.5.0)

BLUEWORKS LIVE PROCESSES

- Ovum** (highlighted with a red arrow)

SMART FOLDERS

- Favorites
- Changed today
- Changed this week
- Validation errors
- Public

Ovum

- Type: Ovum
- Blueworks Live Subscriptions: 1
- Business Process Definitions: 1
- Timing Intervals: 3
 - Ovum_Milestone (Blueworks Live)
 - Ovum_Milestone_1 (Blueworks Live)
 - Ovum_Milestone_2 (Blueworks Live)
- Tracking Groups: 1
 - Ovum (Blueworks Live)

IBM Business Process Manager – Advanced

The edition of IBM Business Process Manager (IBPM) known as IBM Business Process Manager Advanced could trivially be the subject its own book (and more likely many of them). Here we will cover the core concepts and patterns of this edition of the product. First, we will discuss the heritage of this area which will give us a good understanding of why things are the way they are.

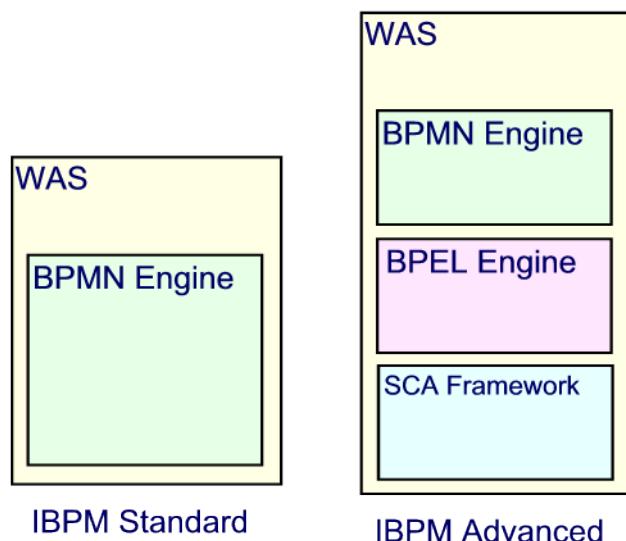
In 2005, IBM released a powerful product to their customers called WebSphere Process Server (WPS). WPS was a BPM runtime based on the industry standard language called BPEL. Beyond simply choreographing work flows, WPS provided a first class environment for achieving integration with external systems. IBM saw the coming together of technology that could be called "Enterprise Application Integration" with business process management. Without the ability to integrate a consumer's existing IT solutions, it was arguable that BPM would not succeed. To achieve this goal, the WPS environment provided an integration framework that was called Service Component Architecture (SCA) that provided a solid grounding for building Service Oriented Architecture (SOA) solutions. All of this will be covered in much more detail later.

After the procurement of the company and products called Lombardi, IBM effectively had two BPM solutions available. These were WebSphere Lombardi Edition and WebSphere Process Server. With the latest release of IBPM, these two, distinct entities have been carefully integrated together to provide the best of all possible worlds. The joining of WLE and WPS provides a super-set of capabilities of both historic products in one run-time environment. Building a solution that involves capabilities of what was once WPS and what was once WLE now becomes a very achievable proposition.

However ... there is a catch.

IBM has chosen *not* to distribute both the heritage WLE and WPS functions with the standard IBPM edition. Instead, for customers that need just WLE functions, the product is sold as IBM Business Process Manager Standard Edition. For customers that can utilize both WLE and WPS functions, there is a second version of the product which is called IBM Business Process Manager Advanced Edition. In this section of the book, I will assume that you now understand that there are two marketed versions of IBM BPM ... standard and advanced and that when I say IBPM from here onwards, I am referring to the edition called IBM Business Process Manager Advanced.

The following diagram illustrates the high level schematic of what is contained in each edition.



The breadth of additional functions available in the IBPM Advanced edition is huge. There are capabilities found in there to solve every conceivable integration and SOA issue. However, with richness comes a degree of complexity. Fortunately, the amount of knowledge needed to build a solution using these functions is easily manageable as long as one realizes that every solution doesn't need to use every capability. By way of an analogy, here is a picture of my TV remote control.



99.9% of the time, I use it to switch on and off the TV, turn up/down the volume and change the channel. In principle, this means that I need only three buttons yet when I look at the device, there are buttons everywhere. Some have labels, some have colors, some have multiple functions. Fortunately, I don't need to learn all the buttons just to watch TV. Once I know what I want to achieve I can then find the appropriate button because I trust that it is there somewhere. The same is true for IBM BPM Advanced. I believe it is important to learn what functions are present, when and where they should be used. The mechanical acts of pressing the correct buttons when needed will simply come by referring to manuals and other documentation.

Information Sources

Right up front it is important to realize that the functions that make IBPM Advanced different from IBPM Standard are primarily what used to be known as WebSphere Process Server (WPS). Since this product has been around for nearly seven years, there is already a wealth of available literature. However, when you read it, you will find that it refers to WPS by name and *not* IBPM. It is expected new materials will refer to the function by the new name but until then, the existing materials are *very* much applicable. As such, perform the mental transformation yourself and realize that when you read WPS you should think IBPM Advanced and when you read WebSphere Integration Developer (WID) you should translate that to Integration Designer (ID).

Redbooks

Redbooks are IBM free books. These are commonly written by IBMers (and occasionally customers) and provide real-world and hands-on guidance to products. There are many existing redbooks that are applicable to our story and what follows is a useful list of the more pertinent ones:

- REDP-4730-00 - [WebSphere Process Server Staff Resolution Using Virtual Groups](#) – 2011-02-23

- REDP-4724-00 - [WebSphere Process Server Versioning: From Design to Production](#) – 2011-01-21
- SG24-7885-00 - [Deploying WebSphere Business Process Management V7 in Secured Production Environments](#) – 2010-11-24
- SG24-7876-00 - [Version-to-Version Migration to IBM WebSphere Dynamic Process Edition V7](#) – 2010-07-16
- SG24-7854-00 - [WebSphere Business Process Management V7 Production Topologies](#) – 2010-05-11
- REDP-4664-00 - [WebSphere Business Process Management and WebSphere Enterprise Service Bus V7 Performance Tuning](#) – 2010-04-30
- REDP-4433-02 - [IBM Business Process Management Reviewer's Guide](#) – 2010-04-16
- REDP-4543-00 - [BPM Solution Implementation Guide](#) – 2009-10-30
- REDP-4551-00 - [WebSphere Business Process Management 6.2.0 Performance Tuning](#) – 2009-07-15
- SG24-7733-00 - [z/OS: WebSphere Business Process Management V6.2 Production Topologies](#) – 2009-10-15
- SG24-7732-00 - [WebSphere Business Process Management V6.2 Production Topologies](#) – 2009-06-25
- REDP-4495-00 - [Business Process Management Enabled by SOA](#) – 2009-03-26
- REDP-4391-00 - [An End-to-End SOA Integration Scenario using IBM WebSphere Process Server on z/OS](#) – 2009-03-05
- REDP-4466-00 - [IBM WebSphere Process Server Best Practices in Error Prevention Strategies and Solution Recovery](#) – 2008-12-29
- REDP-4432-00 - [Configuring IBM WebSphere Process Server V6.1 with an Oracle Database](#) – 2008-07-17
- SG24-7643-00 - [Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 3: Run time](#) – 2008-07-17
- SG24-7642-00 - [Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 2: Scenario](#) – 2008-07-11
- SG24-7608-00 - [Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 1: Development](#) – 2008-06-30
- SG24-7477-00 - [Human-Centric Business Process Management with WebSphere Process Server V6](#) – 2007-10-15
- REDP-4196-00 - [z/OS Technical Overview: WebSphere Process Server and WebSphere Enterprise Service Bus](#) – 2006-11-19
- REDP-4041-00 - [Technical Overview of WebSphere Process Server and WebSphere Integration Developer](#) - 2005-12-06

DeveloperWorks

WebSphere Process Server articles ... [search query](#).

- [Best practices and tuning for large objects in WebSphere Enterprise Service Bus](#) - 2011-10-03
- [Aggregation design patterns and performance considerations in WebSphere Enterprise Service Bus V7.5](#) - 2011-11-16
- [Configuring WS-Security for JAX-WS web services in WebSphere Process Server V7](#) - 2011-09-14
- [Tracing, logging, and error handling in mediation modules using IBM Integration Designer, WebSphere ESB, WebSphere Process Server, and Business Process Manager Advanced Edition, Part 2](#) - 2011-08-24
- [Troubleshooting WebSphere Process Server projects, Part 1: Integrating LDAP systems for authentication and authorization](#) - 2011-07-20
- [Developing a custom tree iWidget for WebSphere Business Space that retrieves data from a REST service using Apache Wink](#) - 2011-07-13
- [Just-In-Time Throttler and Dispatcher for WebSphere ESB](#) - 2011-06-15
- [Performance tuning the WebSphere JDBC Adapter inbound service](#) - 2011-06-08
- [Installing and clustering WebSphere Process Server V7 in non-GUI mode, Part 2: Using the command line to create a cluster](#) - 2011-06-08
- [The Enterprise Service Bus, re-examined](#) - 2011-06-01
- [Dynamically retrieve and map human task information with WebSphere BPM V6.2](#) - 2011-06-01
- [Decoupling business process and business data in WebSphere Process Server: A new twist to the MVC pattern](#) - 2011-06-01
- [Processing an array using an XML map in WebSphere Integration Developer V7](#) - 2011-05-25
- [The Enterprise Service Bus, re-examined](#) - 2011-05-18
- [Operating a WebSphere Process Server environment, Part 4: Archiving considerations for WebSphere Process Server](#) - 2011-05-11
- [Monitoring your business applications, Part 1: Products offering first-class integration with WebSphere Business Monitor](#) - 2011-05-04

- [Data4BPM, Part 2: BPEL4Data: Binding WS-BPEL to Business Entity Definition Language \(BEDL\)](#) - 2011-04-28
- [Setting up a global transaction in an SCA mediation module using WebSphere Adapters](#) – 2011-04-18
- [Using WS-I compliant referenced attachments in WebSphere ESB](#) - 2011-05-18
- [Message binding enhancements in WebSphere Process Server V7.0, Part 1: Developing integration components that interact with WebSphere MQ and other messaging providers](#) – 2011-04-13
- [Installing and clustering WebSphere Process Server V7.0 in non-GUI mode, Part 1: Silent installation and setup](#) – 2011-04-06
- [Enabling web services security between modules in WebSphere Process Server V7.0](#) - 2011-03-16
- [Managing failed flows using the Failed Event Manager API in WebSphere Process Server](#) – 2011-03-02
- [Migrating from WebSphere Process Server V6.0.2 to V7](#) – 2011-02-16
- [Invoking cross-cell EJBs using WebSphere Integration Developer V7 or WebSphere Enterprise Service Bus V7](#) – 2011-02-16
- [Using COBOL copybooks in WebSphere Integration Developer V7](#) – 2011-02-09
- [BPM Voices: Standards and why they matter for BPM](#) – 2011-01-19
- [Business process management adoption scenarios](#) – 2011-01-19
- [Recommendations for iterative development in WebSphere Business Modeler](#) – 2011-01-19
- [Configuring WS-Security for JAX-RPC web services in WebSphere Process Server V7](#) – 2011-01-12
- [Importing WS-BPEL 2.0 process definitions](#) – 2011-01-05
- [Comment lines: WebSphere Process Server and WebSphere Lombardi Edition provide different, flexible BPM options](#) – 2010-12-15
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 1: Generating the data model](#) – 2010-12-08
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 2: Generating the JPA entities](#) – 2010-12-08
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 3: Creating a stateless session EJB](#) - 2010-12-08
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 4: Creating an SCA client](#) – 2010-12-08
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 5: Creating a BPEL process](#) – 2010-12-08
- [Using the Java Persistence API 2.0 services with WebSphere Process Server V7, Part 6: Generating the user interface](#) – 2010-12-08
- [Ensuring transactional integrity using Web Services Atomic Transaction support in WebSphere ESB and WebSphere Application Server](#) – 2010-12-08
- [Implementing web services transactions in WebSphere Process Server V7](#) – 2010-11-24
- [Implementing tracing, logging, and error handling in mediation modules using WebSphere Integration Developer and WebSphere ESB V7, Part 1](#) – 2010-11-24
- [Multi-module monitoring with the V7 WebSphere BPM suite, Part 2: Use new plug-ins for WebSphere Integration Developer V7 to achieve end-to-end business process monitoring](#) – 2010-11-03
- [Building human tasks with the Virtual Member Manager and LDAP](#) – 2010-11-03
- [WebSphere Process Server database configuration made easy](#) – 2010-10-27
- [WebSphere Process Server relationship service: Efficiently manage static relationship instance data](#) – 2010-10-20
- [Multi-module monitoring with the V6.2 WebSphere BPM suite](#) – 2010-10-06
- [Multi-module monitoring with the V7 WebSphere BPM suite, Part 1: Use two new plug-ins for WebSphere Business Modeler V7 to achieve end-to-end business process monitoring](#) – 2010-10-06
- [Integrating WebSphere Process Server and SCA Feature Pack, Part 3: Interoperability across SCA web services bindings](#) – 2010-09-29
- [Implementing the cancellation function in WebSphere Process Server, Part 1: Adding the correlation set and event handler](#) - 2010-09-22
- [Implementing the cancellation function in WebSphere Process Server, Part 2: Adding the compensation and invoking the cancellation with the Business Process Choreographer EJB API](#) – 2010-09-22
- [Static and dynamic relationships in WebSphere Process Server and WebSphere ESB V7](#) – 2010-09-15
- [Performance tuning of throughput-based SOA solutions for WebSphere Process Server](#) – 2010-09-15
- [Using the Java Persistence API to initialize HTML-Dojo forms for WebSphere Business Space](#) – 2010-09-08
- [Implementing pagination query of business requests with custom properties in WebSphere Process Server](#) – 2010-08-25
- [Migrating process instances in WebSphere Process Server V7](#) – 2010-08-11
- [Best practices for BPM and SOA performance](#) – 2010-08-04

- [Using advanced human task patterns in WebSphere Process Server V7](#) – 2010-08-04
- [Using pureXML in SCA component development with WebSphere Integration Developer V7](#) – 2010-07-28
- [Migrating a business solution from WebSphere Business Integration Adapters to WebSphere Adapters V6.2](#) – 2010-07-28
- [Creating a custom Business Process Choreographer Explorer view of WebSphere Process Server](#) – 2010-07-21
- [Developing JAX-WS web service integration solutions using WebSphere Integration Developer V7](#) – 2010-07-21
- [Integrating WebSphere Process Server and SCA Feature Pack, Part 2: Interoperability across SCA JMS bindings](#) – 2010-07-21
- [Integrating WebSphere Process Server and SCA Feature Pack, Part 1: Interoperability across SCA bindings](#) – 2010-07-01
- [Concurrent human task assignment in WebSphere Process Server V7](#) – 2010-06-30
- [Using the store-and-forward feature in WebSphere Process Server V7.0](#) – 2010-06-16
- [Automatic synchronization of EIS data using the WebSphere Process Server relationship service](#) – 2010-06-09
- [Setting custom properties for new query requirements in WebSphere Process Server](#) – 2010-06-02
- [What's new in WebSphere Business Modeler and WebSphere Business Compass V7](#) – 2010-05-26
- [Using binary Jar files with WebSphere Integration Developer and WebSphere Process Server](#) – 2010-05-12
- [Dynamically select adapters based on context using WebSphere Business Services Fabric](#) – 2010-05-05
- [Troubleshooting WebSphere Process Server deadlocks and timeouts](#) – 2010-05-05
- [Interacting with WebSphere Process Server via the Web service API and JAXB marshalling](#) – 2010-04-28
- [Measuring performance of WebSphere Process Server applications using the Request Metrics tool](#) - 2010-04-21
- [Solution design in WebSphere Process Server and WebSphere ESB, Part 3: Process implementation types: Patterns based design for process-based solutions](#) – 2010-04-14
- [Building a RESTful integration for SAP using WebSphere Adapter and WebSphere Process Server, Part 1: Outbound](#) – 2010-04-07
- [Operating a WebSphere Process Server environment, Part 3: Setup, configuration, and maintenance of the WebSphere Process Server Business Process Choreographer database](#) - 2010-04-01
- [Develop a rich Web client for a business process application with WebSphere Integration Developer, WebSphere Process Server, and Adobe Flex](#) – 2010-03-24
- [Using Business Space to manage business process applications](#) – 2010-03-24
- [Migrating WebSphere InterChange Server artifacts to WebSphere Process Server V7.0, Part 1: Migrated BPEL improvements](#) – 2010-03-17
- [Monitoring business database changes with a WebSphere Adapter](#) – 2010-03-10
- [Migrating to WebSphere Process Server Version 7](#) - 2010-03-03
- [Using WebSphere MQ bindings in WebSphere ESB, Part 1: Manipulating MQ headers in WebSphere ESB using the XSL transformation primitive and WebSphere Integration Developer](#) – 2010-03-03
- [Customizing WebSphere Business Space V6.2 Human Management widgets](#) - 2010-02-24
- [Overview of ILOG JRules and WebSphere Process Server integration](#) – 2010-02-10
- [Feeding iPhone applications with WebSphere Process Server V6.2 services](#) - 2010-01-27
- [Mediation policy for target services: Constructing a dynamic mediation for WebSphere ESB V7 based on which target service has been selected](#) – 2010-01-27
- [Integrating Oracle E-Business Suite through the Oracle Open Interface and Concurrent Program using WebSphere Adapters](#) - 2010-01-27
- [Managing WebSphere Adapters more effectively through wsadmin in WebSphere Process Server](#) - 2010-01-20
- [Installing and configuring the 6.1.2 business process management stack on z/OS](#) – 2010-01-18
- [Develop custom widgets for Business Space with Rational Application Developer or WebSphere Integration Developer](#) – 2009-12-20
- [Business process monitoring in WebSphere Process Server V6.2 using WebSphere Business Monitor V6.2, Part 1: Configuring the integration](#) – 2009-12-16
- [Operating a WebSphere Process Server environment, Part 1: Overview](#) – 2009-12-16
- [Operating a WebSphere Process Server environment, Part 2: Options for maintaining an optimal Business Process Choreographer database size](#) - 2009-12-16
- [Developing custom widgets for Business Space using Dojo, Part 1: Generating Dojo markup using a generic markup handler](#) – 2009-12-09
- [Ensuring high availability and performance of events delivery with WebSphere Process Server and WebSphere JDBC Adapter](#) - 2009-12-02
- [Using DynaCache to improve the performance of your WebSphere Process Server or WebSphere ESB solution](#) – 2009-12-02

- [Integrating a Dojo client with an SCA application via SCA HTTP binding](#) – 2009-11-25
- [Migrating to WebSphere Process Server V6.2](#) - 2009-11-11
- [Event sequencing using WebSphere Process Server](#) - 2009-10-28
- [Managing WebSphere Adapters more effectively through wsadmin in WebSphere Process Server](#) – 2009-10-14
- [Build and deploy a business process model using WebSphere Business Process Modeler Advanced and Lotus Forms, Part 3: Implement a mediation flow](#) – 2009-10-14
- [Using the MQ binding plug-in for WebSphere Integration Developer 6.0.2](#) - 2009-10-07
- [WebSphere Process Server throughput management, Part 2](#) – 2009-11-30
- [Using the Query Table Builder in WebSphere Process Server V6.2](#) - 2009-11-23
- [Integrating WebSphere Business Events V6.2 with WebSphere ESB and WebSphere Process Server](#) – 2009-11-16
- [Improve performance and usability by implementing TopCount functionality for WebSphere Business Monitor dimensions with DB2 Alphablox](#) – 2009-11-02
- [Building mediation flows in WebSphere Integration Developer for deployment on WebSphere ESB or WebSphere Process Server](#) – 2009-08-19
- [Building mediation flows in WebSphere Integration Developer for deployment on WebSphere ESB or WebSphere Process Server](#) – 2009-08-19
- [Solution design in WebSphere Process Server and WebSphere ESB: Part 2](#) – 2009-08-12
- [Validating business objects in Websphere Process Server](#) – 2009-07-29
- [Implementing your WebSphere Process Server 6.1 business module the RESTful way](#) – 2009-07-15
- [Restoring the WebSphere Process Server deployment manager](#) – 2009-07-01
- [WebSphere Process Server throughput management, Part 1](#) – 2009-07-01
- [Exploring WebSphere Process Server transactionality](#) - 2009-06-17
- [Getting more out of the WebSphere SAP Adapter](#) – 2009-06-03
- [Configuring WebSphere Business Integration Adapters for TCP/IP V1.0.4 on WebSphere Process Server V6.2](#) – 2009-05-20
- [Solution design in WebSphere Process Server: Part 1](#) – 2009-05-13
- [Extending the power of the WebSphere Process Server business rules component](#) – 2009-05-06
- [Asynchronous processing in WebSphere Process Server](#) – 2009-04-29
- [Business integration using WebSphere Partner Gateway, Websphere Process Server, and Web services](#) – 2009-04-22
- [Build and deploy a business process model using WebSphere Business Process Modeler Advanced and Lotus Forms, Part 2: Implement and test the BPEL process using WebSphere Integration Developer](#) – 2009-04-15
- [Customizing HTML-Dojo forms for Business Space powered by WebSphere](#) - 2009-04-09
- [Building a process task portlet application using the Portlet Generator in WebSphere Integration Developer 6.2](#) – 2009-04-08
- [Developing a PHP client using the REST API in WebSphere Process Server 6.2](#) – 2009-04-01
- [Using SCA HTTP binding in typical real-life scenarios with WebSphere Integration Developer](#) - 2009-04-01
- [Using WebSphere Process Server business calendars in business processes](#) – 2009-03-25
- [Working with WebSphere Process Server Relationship Service APIs: Part 1: Pre-populating relationship instance tables](#) – 2009-03-05
- [Securing JMS connections to WebSphere Enterprise Service Bus V6.1 or V6.2](#) – 2009-02-18
- [Using WebSphere Process Server operational architecture to design your applications: Part 2: Implementation: SCA runtime, Business Process Choreographer, and supporting services](#) – 2009-02-11
- [What's new in WebSphere Business Monitor V6.2](#) – 2009-01-28
- [Comment lines: Bhargav Perepa: Business calendars and timetables can be fun](#) – 2009-01-28
- [Using WebSphere MQ bindings in WebSphere ESB, Part 3: Using custom WebSphere MQ headers with WebSphere ESB](#) – 2009-01-28
- [Expanding clustered topologies for WebSphere Process Server and WebSphere Enterprise Service Bus](#) – 2009-01-07
- [Repairing processes with WebSphere Process Server](#) – 2008-12-17
- [Build and deploy a business process model using WebSphere Business Process Modeler Advanced and Lotus Forms, Part 1: Create and export the model](#) – 2008-12-10
- [Test-driven development in an SOA environment: Part 2: Continuous integration with WebSphere Process Server](#) – 2008-12-10

- [Customize your BPM user interfaces with business spaces](#) – 2008-12-04
- [What's new in WebSphere Process Server V6.2](#) – 2008-12-04
- [What's new in WebSphere Integration Developer V6.2](#) – 2008-12-04
- [Managing WebSphere Process Server Relationship Service database objects](#) – 2008-11-12
- [Analyzing Java virtual machine performance in WebSphere Process Server](#) – 2008-11-05
- [Endurance testing with WebSphere Process Server V6.1](#) – 2008-11-05
- [WebSphere Process Server invocation styles](#) – 2008-10-29
- [Error handling in WebSphere Process Server, Part 1: Developing an error handling strategy](#) – 2008-10-29
- [WebSphere Process Server business rules lifecycle](#) – 2008-10-22
- [Asynchronous replication of WebSphere Process Server and WebSphere Enterprise Service Bus for disaster recovery environments](#) – 2008-11-24
- [WebSphere Process Server operational architecture: Part 1: Base architecture and infrastructure components](#) – 2008-11-17
- [Use mediation flows to integrate WebSphere Service Registry and Repository with WebSphere Process Server](#) – 2008-09-10
- [Versioning business processes and human tasks in WebSphere Process Server](#) – 2008-08-13
- [Advanced techniques and patterns for business process client development](#) – 2008-08-06
- [IBM SOA Foundation product integration: Leveraging Information as a Service in your WebSphere-based SOA solution](#) – 2008-07-30
- [What's new in WebSphere Integration Developer V6.1.2](#) – 2008-07-23
- [What's new in WebSphere Process Server V6.1.2](#) – 2008-07-23
- [Web services with SOAP over JMS in IBM WebSphere Process Server or IBM WebSphere Enterprise Service Bus, Part 2: Using the IBM WebSphere MQ JMS provider](#) – 2008-07-17
- [Using WebSphere Transformation Extender V8.2 with WebSphere Process Server and WebSphere ESB V6.1, Part 1: Generating the WTX maps for use with WebSphere Process Server and WebSphere ESB](#) – 2008-07-09
- [Using WebSphere Transformation Extender V8.2 with WebSphere Process Server and WebSphere ESB V6.1, Part 2: Utilizing the WTX Data Binding with the Flat File Adapter](#) – 2008-07-09
- [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 5: Deploying the user interfaces](#) – 2008-06-18
- [IBM SOA Foundation product integration: Managing your WebSphere-based SOA solution](#) – 2008-06-18
- [Getting started with Lotus Forms in the WebSphere business process management suite](#) – 2008-06-18
- [SOA integration: Decouple service consumers from service providers over an ESB](#) – 2008-06-16
- [Use ARM to monitor SCA invocations in IBM WebSphere Process Server V6.1, Part 2: Understand SCA invocation patterns and debug asynchronous scenarios](#) – 2008-06-05
- [Use ARM to monitor SCA invocations in IBM WebSphere Process Server V6.1, Part 1: Debug SCA invocations using IBM Tivoli Composite Application Manager for Response Time Tracking](#) – 2008-05-29
- [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 4: Developing a task list application](#) – 2008-05-28
- [Web services with SOAP over JMS in IBM WebSphere Process Server or IBM WebSphere Enterprise Service Bus, Part 1: Using the SIBus JMS provider](#) – 2008-05-22
- [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 3: Developing user interfaces for originating and participating tasks](#) – 2008-04-30
- [Get started with WebSphere business process management V6.1 software](#) – 2008-04-24
- [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 1: Solution overview](#) – 2008-04-16
- [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 2: Developing a human-centric business process](#) – 2008-04-16
- [Developing integration solutions with WebSphere Process Server relationships](#) – 2008-04-03
- [Business Process Management with SOA, Part 3: Combining process execution, content management and user interactions](#) – 2008-04-02
- [Building SOA composite business services, Part 12: Combine document-centric workflows in IBM FileNet with business state machines in IBM WebSphere Process Server](#) – 2008-03-27
- [WebSphere Process Server and Lotus Forms integration](#) – 2008-03-26
- [Selecting a human task with custom properties using WebSphere Process Server](#) – 2008-03-26

- [Configuring SCA HTTP binding to enable real-life scenarios](#) – 2008-03-26
- [Building clustered topologies in WebSphere Process Server V6.1](#) – 2008-03-19
- [Modeling human tasks with WebSphere Business Modeler](#) – 2008-03-12
- [BPEL or ESB: Which should you use?](#) - 2008-03-12
- [Keeping your data in sync with WebSphere Process Server relationships](#) – 2008-03-12
- [Versioning and dynamicity: Part 2: Applying versioning and process dynamicity using WebSphere Process Server V6.1](#) – 2008-03-05
- [Map Web services with WebSphere Integration Developer](#) - 2008-02-20
- [End-to-end BPEL business activity monitoring with IBM SOA tools: Part 4: Deploying the BPEL workflow application](#) – 2008-02-20
- [Handling unmodeled faults within WebSphere Process Server V6.1](#) – 2008-02-06
- [Versioning and dynamicity: Part 1: Creating multiple versions of a business process with WebSphere Process Server](#) – 2008-02-06
- [Modeling business processes in WebSphere Business Modeler for BPEL transformation](#) – 2008-01-23
- [Authorization and staff resolution in Business Process Choreographer: Part 4: Staff resolution specifications and reference guide](#) – 2008-01-23
- [Recovering from failed asynchronous SCA service invocations on WebSphere Process Server](#) – 2008-01-16
- [Using the WebSphere DataPower SOA Appliance with WebSphere Process Server](#) – 2008-01-16
- [Overview of bidirectional script support in IBM WebSphere Integration Developer](#) – 2008-01-01
- [Authorization and staff resolution in Business Process Choreographer: Part 3: Customization options for staff resolution](#) – 2007-12-12
- [Incorporating database access in business processes using WebSphere Integration Developer 6.0.2](#) – 2007-12-05
- [What's new in WebSphere Business Modeler V6.1](#) – 2007-12-05
- [What's new in WebSphere Process Server V6.1](#) – 2007-12-05
- [Authorization and staff resolution in Business Process Choreographer: Part 2: Understanding the programming model for staff resolution](#) – 2007-11-07
- [Developing Eclipse Rich Client Platform applications for WebSphere: Part 5: Process human tasks in WebSphere Process Server](#) – 2007-11-07
- [Creating custom data bindings in WebSphere Process Server without coding](#) – 2007-10-31
- [Hello World: Monitor a simple business process using WebSphere Business Monitor V6.0.2](#) – 2007-10-31
- [Business process standards, Part 2: How the standards are used in WebSphere products](#) – 2007-10-24
- [Merging partial result sets from multiple sources of DB2 for Linux, UNIX, and Windows and WebSphere Process Server Business Process Choreographer](#) – 2007-10-24
- [Business process standards, Part 1: An introduction](#) – 2007-10-17
- [Transactionally integrate CICS COBOL applications into WebSphere Process Server SOA solutions](#) – 2007-10-17
- [Using WebSphere Integration Developer and WebSphere Flat File Adapter with custom data bindings](#) – 2007-10-10
- [Running migrated WebSphere InterChange Server projects in WebSphere Process Server with global and J2EE security enabled](#) – 2007-10-10
- [Business logic versus connectivity logic: Using WebSphere Process Server and WebSphere ESB together](#) – 2007-10-03
- [End-to-end BPEL business activity monitoring with IBM SOA tools: Part 2: Developing a BPEL workflow application in WebSphere Integration Developer](#) – 2007-10-3
- [Authorization and staff resolution in Business Process Choreographer: Part 1: Understanding the concepts and components of staff resolution](#) – 2007-09-26
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 3, Migrating a business process to WebSphere Process Server](#) – 2007-09-12
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 4, Benefits of migrating WebSphere Commerce using WebSphere Business Modeler and Integration Developer](#) – 2007-09-12
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 3, Migrating a business process to WebSphere Process Server](#) – 2007-09-17
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 5, Tying it all together with the Esperanto methodology](#) – 2007-09-12
- [Using WebSphere Business Services Fabric with SEEC Web service components](#) - 2007-09-05
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 2, Getting WebSphere Commerce and WebSphere Process Server talking](#) – 2007-09-01
- [Migrating WebSphere Commerce to the SOA foundation tooling: Part 1, Introduction to series](#) – 2007-09-01

- [Integrate metadata between WebSphere Service Registry and Repository and WebSphere Business Services Fabric](#) – 2007-08-29
- [Handling exceptions across sequential invocations of SCA business process components in WebSphere Process Server](#) – 2007-08-29
- [Dynamic configuration of the Human Task Manager in WebSphere Process Server](#) – 2007-08-22
- [WebSphere Process Server Business State Machines, Part 3: Adding more Business State Machine capabilities to your vending machine](#) – 2007-08-08
- [Building a human task-centric business process with WebSphere Process Server, Part 4: Portlet development](#) – 2007-07-30
- [Differences between WebSphere InterChange Server and WebSphere Process Server support for event sequencing](#) – 2007-07-25
- [Endurance testing with WebSphere Process Server V6.0.2](#) – 2007-07-18
- [Configuring WebSphere Partner Gateway V6.1 and WebSphere Process Server V6.0.2, Part 3: Communication configuration](#) – 2007-06-27
- [Integrate a sales order validation process between SAP and Siebel using WebSphere Process Server and WebSphere Adapters](#) – 2007-06-27
- [Integrating business processes with TIBCO Enterprise Message Service and WebSphere Process Server](#) – 2007-06-20
- [Migrating WebSphere Business Integration Adapter solutions to WebSphere Process Server V6](#) – 2007-06-20
- [End-to-end BPEL business activity monitoring with IBM SOA tools: Part 1: Modeling for process execution with WebSphere Business Modeler](#) – 2007-06-13
- [Building a human task-centric business process with WebSphere Process Server, Part 3: Adding portlets](#) – 2007-06-06
- [Configuring WebSphere Partner Gateway V6.1 and WebSphere Process Server V6.0.2, Part 2: JMS integration case studies](#) – 2007-05-30
- [Configuring WebSphere Partner Gateway V6.1 and WebSphere Process Server V6.0.2, Part 1: JMS integration overview](#) – 2007-05-23
- [Leveraging WebSphere Portal V6 programming model: Part 3. Introducing Composite Application Workflow in WebSphere Portal](#) – 2007-05-16
- [Exception handling in WebSphere Process Server and WebSphere Enterprise Service Bus](#) – 2007-05-16
- [BPEL fault handling in WebSphere Integration Developer and WebSphere Process Server](#) – 2007-04-25
- [Clustering WebSphere Process Server V6.0.2, Part 2: Install and configure WebSphere Process Server clusters](#) – 2007-04-18
- [Clustering WebSphere Process Server V6.0.2, Part 1: Understanding the topology](#) – 2007-04-18
- [IBM WebSphere Developer Technical Journal: Developing adaptive composite business services using WebSphere Business Services Fabric, Part 1](#) – 2007-04-04
- [Using WebSphere MQ binding with a client channel definition table](#) – 2007-04-04
- [Getting connected with WebSphere Integration Developer adapters : Part 1, An introduction to connecting with adapters](#) - 2007-04-04
- [Installing a WebSphere Process Server 6.0.2 clustered environment](#) – 2007-03-28
- [Designing and versioning compatible Web services](#) – 2007-03-28
- [Building a human task-centric business process with WebSphere Process Server, Part 2: Customizing JSPs](#) – 2007-03-21
- [Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 2: Understanding the WebSphere Process Server SCA components](#) – 2007-03-14
- [How to use additional JMS providers with WebSphere Process Server and WebSphere Enterprise Service Bus Version 6.02](#) – 2007-02-28
- [Connecting Eclipse Rich Client Platform Applications to WebSphere Application Server V6 Service Integration Bus](#) – 2007-02-21
- [WebSphere Process Server Business State Machines, Part 2: Enhancing your vending machine](#) – 2007-02-21
- [Using SOA with WebSphere process integration products, Part 7: Configure WebSphere security using a file-based custom user registry](#) – 2007-02-20
- [Building a human task-centric business process with WebSphere Process Server, Part 1: Using Business Process Choreographer Explorer](#) – 2007-02-14
- [WebSphere Process Server made easy, Part 3: Application deployment and server resource configuration](#) – 2007-02-06
- [Considering WebSphere on System z for your SOA environment](#) – 2007-01-31
- [IBM WebSphere Developer Technical Journal: Reliable and repeatable unit testing for Service Component Architecture modules, Part 3](#) – 2007-01-24
- [WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 2: My first WebSphere Process Server cluster](#) – 2007-01-17
- [Tip: Canceling a human task](#) – 2007-01-17
- [What's new in WebSphere Process Server and WebSphere Integration Developer Version 6.0.2?](#) - 2006-12-20
- [Migrating WebSphere InterChange Server artifacts to WebSphere Process Server artifacts, Part 1: Migrating collaboration templates to BPEL](#) –

2006-12-13

- [Developing business processes that use WebSphere Portal V6 and WebSphere Process Server V6](#) – 2006-12-06
- [IBM WebSphere Developer Technical Journal : A guided tour of WebSphere Integration Developer -- Part 8](#) – 2006-12-06
- [WebSphere Process Server made easy, Part 1: Architecture](#) – 2006-12-05
- [Using WebSphere Business Modeler to build business integration modules in WebSphere Integration Developer](#) – 2006-11-29
- [Using SOA with WebSphere Process Integration products: Part 6: Business measures and monitoring](#) – 2006-11-15
- [WebSphere Process Server and WebSphere Enterprise Service Bus deployment patterns, Part 1: Selecting your deployment pattern](#) – 2006-11-08
- [Interactions with WebSphere Process Server and WebSphere ESB using Representational State Transfer](#) – 2006-11-08
- [IBM WebSphere Developer Technical Journal : A guided tour of WebSphere Integration Developer -- Part 7](#) - 2006-10-25
- [Using SOA with WebSphere Process Integration products: Part 5: Sequential versus concurrent processes](#) – 2006-10-18
- [Creating and deploying business rules](#) – 2006-10-11
- [WebSphere Process Server Business State Machines concepts and capabilities, Part 1: Exploring basic concepts](#) – 2006-10-11
- [IBM WebSphere Developer Technical Journal : A guided tour of WebSphere Integration Developer -- Part 6](#) - 2006-09-20
- [IBM WebSphere Developer Technical Journal: Reliable and repeatable unit testing for Service Component Architecture modules -- Part 2](#) – 2006-09-20
- [Invoke WebSphere business processes from a database](#) – 2006-09-13
- [Integrate WebSphere Business Integration Adapters with WebSphere Process Server, Part 3: One-way asynchronous request scenario](#) – 2006-08-23
- [IBM WebSphere Developer Technical Journal: Reliable and repeatable unit testing for Service Component Architecture modules -- Part 1](#) – 2006-08-23
- [Web services client programming for WebSphere Process Server](#) – 2006-08-21
- [Publish/subscribe interactions in WebSphere Process Server and WebSphere ESB](#) – 2006-08-16
- [Using BPEL and EJBs with WebSphere Process Server and WebSphere Integration Developer](#) – 2006-08-09
- [Configuring dynamic authentication in WebSphere Adapter for SAP](#) – 2006-08-09
- [Using SOA with WebSphere process integration products, Part 4: Enabling the CBE and CEI infrastructure](#) – 2006-08-02
- [Implementing two-way interactions between a business state machine and a human task using WebSphere Process Server](#) – 2006-07-26
- [Use a generic data handler for inbound delimited flat files using WebSphere Process Server and WebSphere JCA adapter for flat files](#) – 2006-07-26
- [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6 -- Part 4](#) – 2006-07-26
- [Comment lines: Robert Peterson: Be more productive using WebSphere Integration Developer](#) – 2006-07-26
- [Developing business processes that use IBM WebSphere Portal and WebSphere Process Server](#) – 2006-06-12
- [Create a business process portlet application with WebSphere Process Server V6 using Business Process Choreographer APIs](#) – 2006-06-21
- [Implementing custom databinding and custom function selectors in IBM WebSphere Adapters](#) – 2006-06-14
- [Using SOA with WebSphere process integration products, Part 3: Business-driven development](#) – 2006-05-31
- [Websphere Process Server relationship service: Part 2, Dynamic relationships](#) – 2006-05-17
- [Integrating WebSphere Partner Gateway V6 and WebSphere Process Server V6 using JMS: Part 2: In-depth case studies](#) – 2006-05-10
- [Integrating WebSphere Partner Gateway V6 and WebSphere Process Server V6 using JMS: Part 3: Communication configuration](#) – 2006-05-10
- [Integrating WebSphere Partner Gateway V6 and WebSphere Process Server V6 using JMS: Part 1: Overview](#) – 2006-05-10
- [Integrate EJB services with WebSphere Process Server, Part 2: Advanced scenarios for integrating EJBs running on WebSphere Process Server and WebSphere Application Server](#) – 2006-05-03
- [Recommended reading list: Service-Oriented Architecture and WebSphere Process Server](#) – 2006-04-26
- [Integrate WebSphere Business Integration Adapters with WebSphere Process Server V6: Synchronous Request Response \(OutboundRequest\) scenario -- Part 2](#) - 2006-04-26

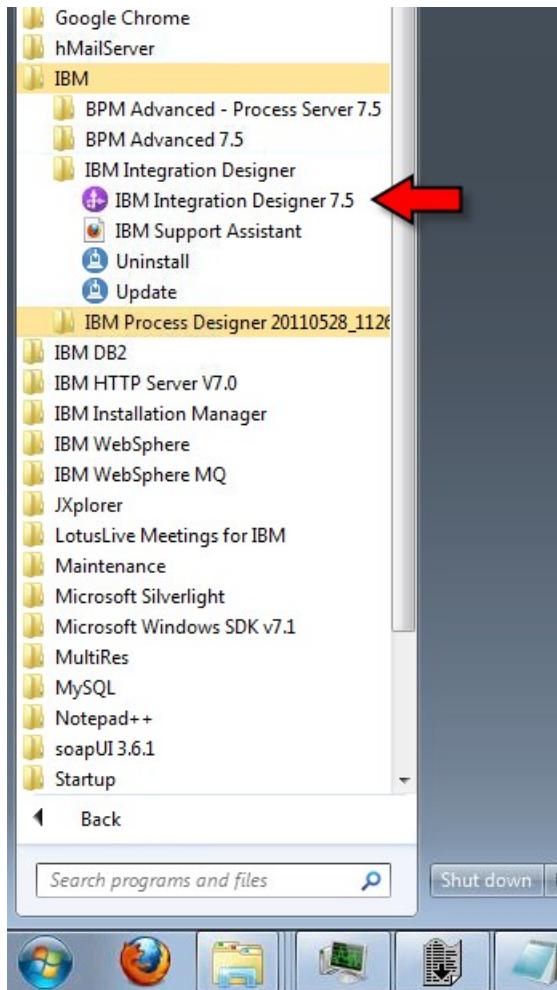
Other

A list of knowledge sources has been compiled by IBM and refreshed periodically. Much of its is already listed in the previous sections but it does make a good reference:

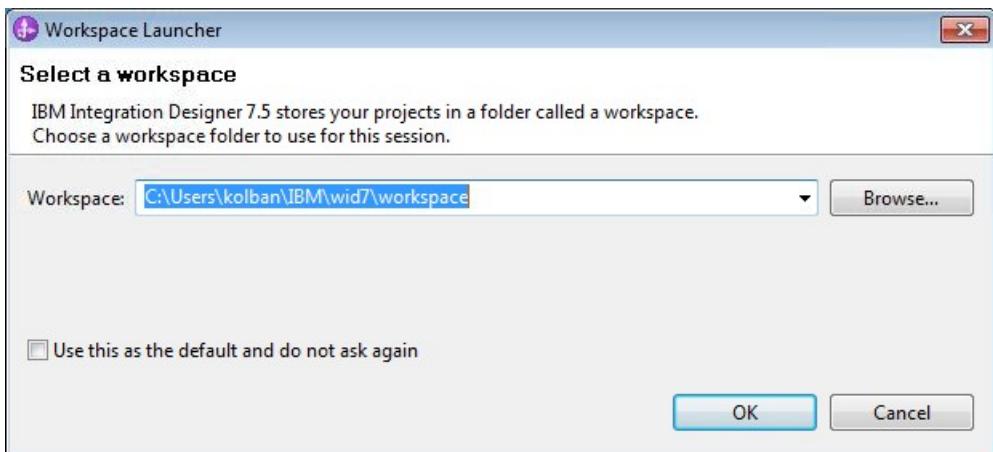
- [IBM Business Process Manager 7.5 and WebSphere Process Server 6.X and 7.0 Public Knowledge Compendium](#)

Integration Designer

The development environment for building SCA and BPEL based modules is called *Integration Designer*. It is a standard Eclipse based environment. Once installed, it can be started from the following menu item:

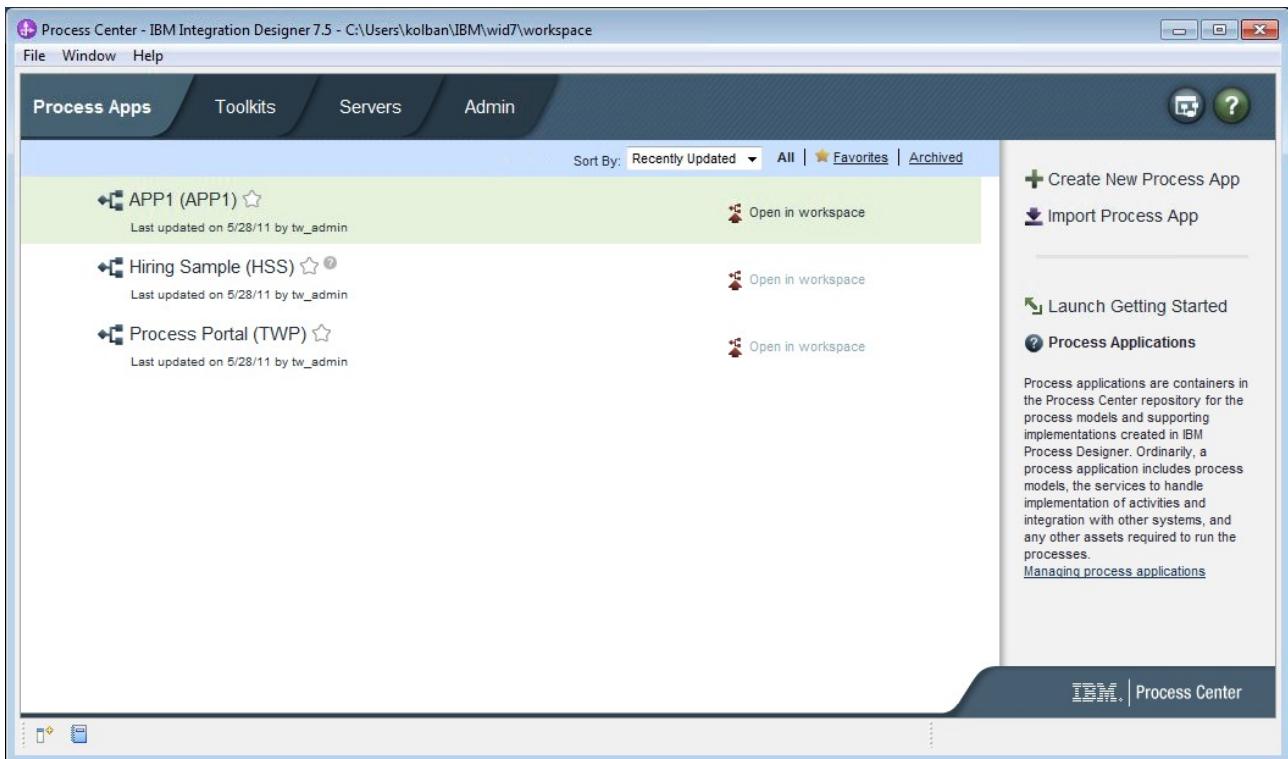


When Integration Designer (ID) starts, it asks for the location of a WorkSpace directory:

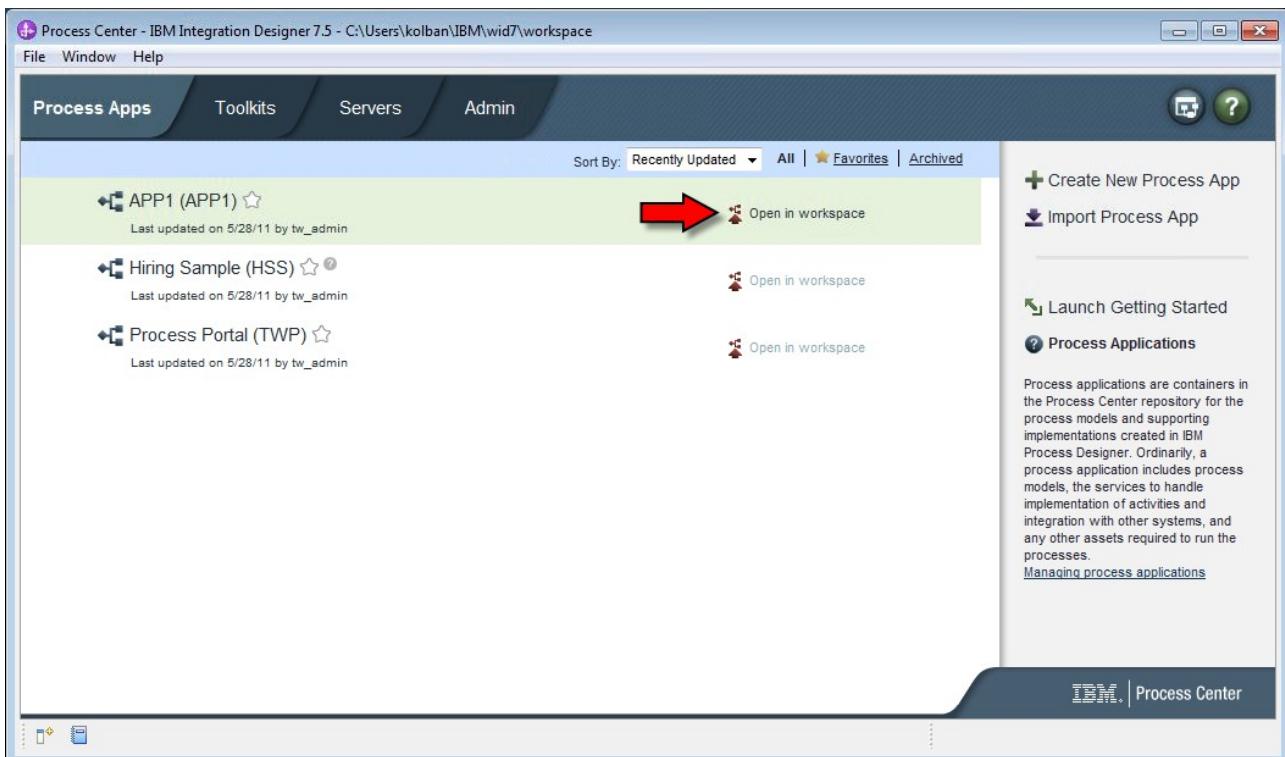


The WorkSpace directory is where the artifacts of a solution are stored locally on the file system. Any directory path can be supplied and it is not uncommon to create a new WorkSpace for each different project upon which you work.

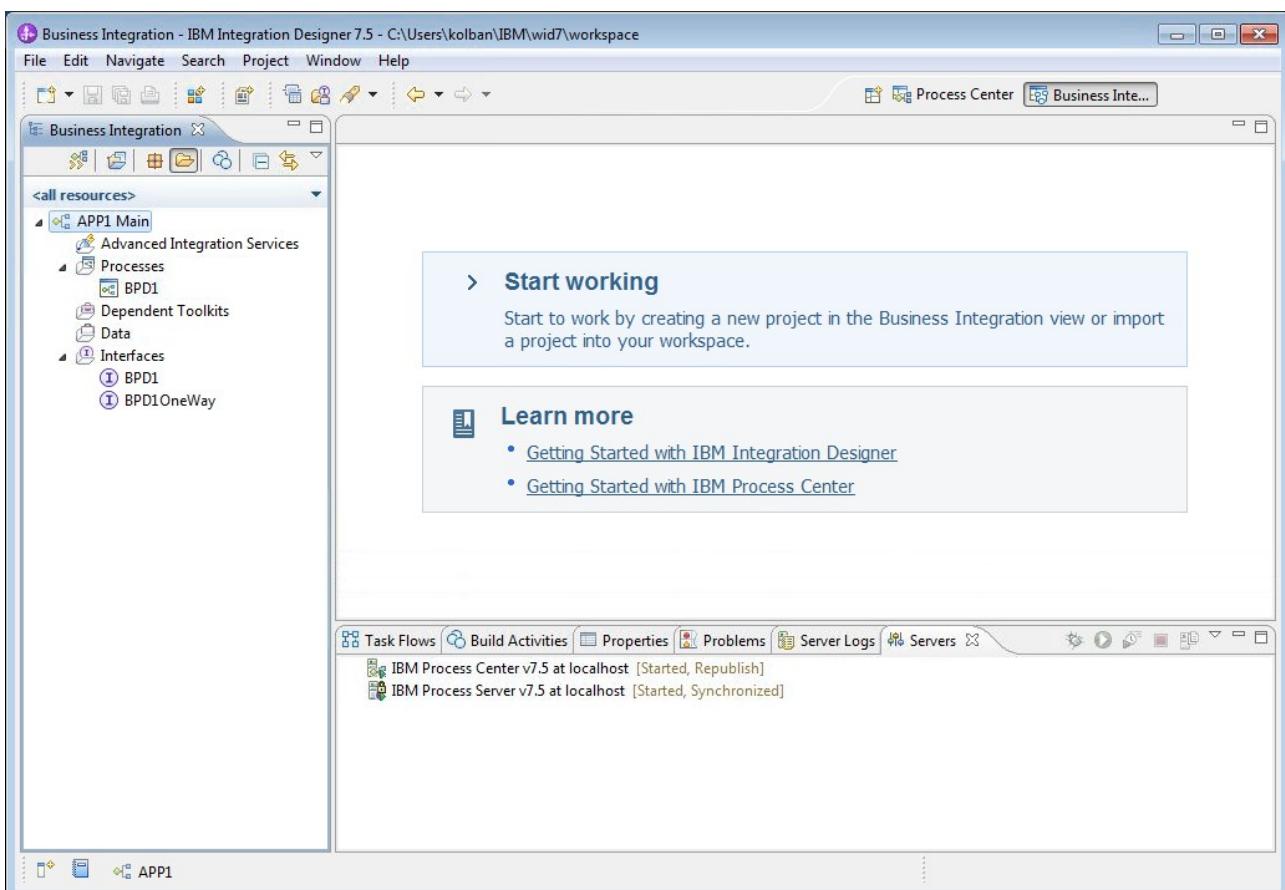
ID has an Eclipse perspective called Process Center. When selected, it connects to the Process Center repository and displays the process applications that can be found there.



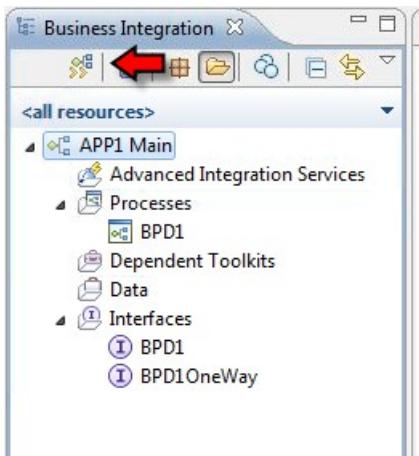
Selecting a Process Application and then "Open in workspace" causes the process application to be copied from the Process Center into the local file system workspace.



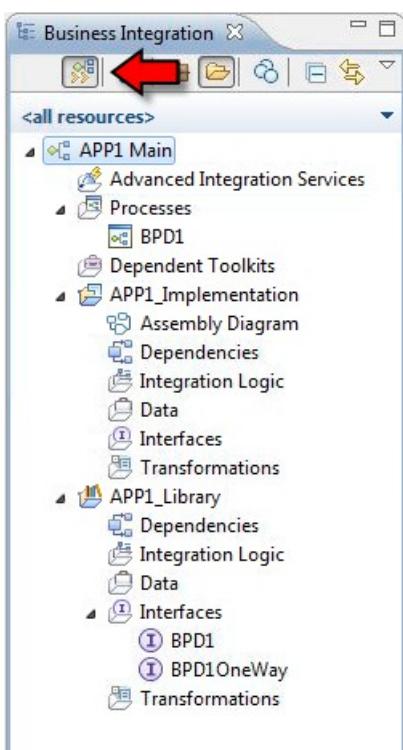
The ID Eclipse Perspective is switched to the Business Integration perspective which is where the majority of work is performed:



By default, a project is shown in its *simple* mode. This means that only BPMN and related artifacts are shown:



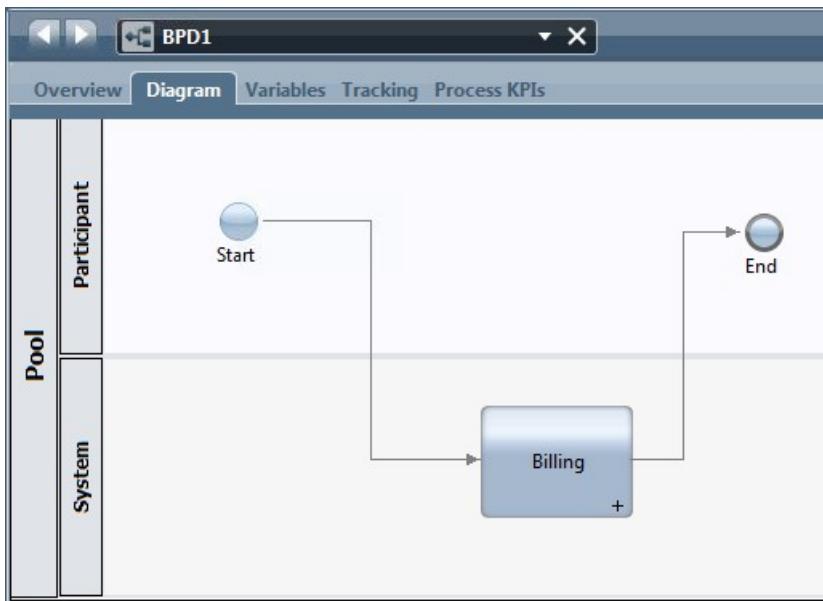
However, there is a toggle button that switches the project to view it in its *advanced* mode which then shows more details including the SCA and BPEL artifacts:



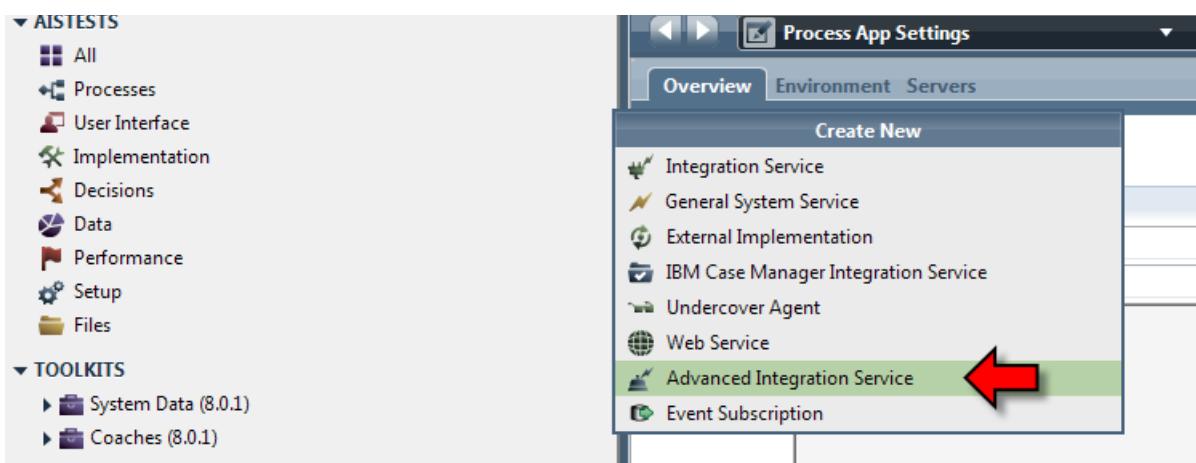
Inter-operating between a BPMN process and an SCA module

When we think of a BPMN process contained in a BPD which is itself contained within a Process Application, we realize that it can "call out" to a variety of back-end systems using its supplied Java implemented connectors. However, these are secondary capabilities in the overall story. The more robust and powerful integration engine is contained within an SCA module. As such, we want to be able to define a step within a BPD as being implemented *by* an SCA module.

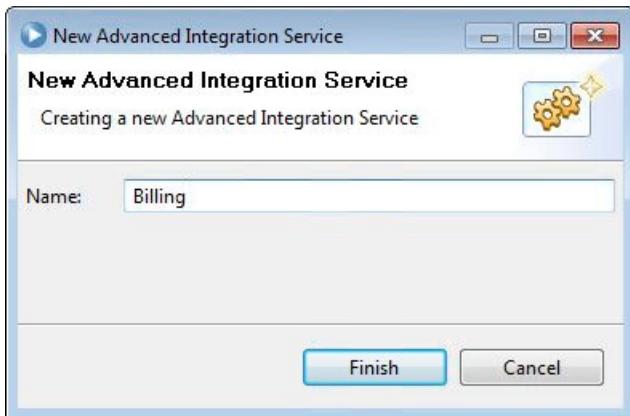
Let us look at a simple example.



Here is a BPD which has a step called Billing. We wish this to be implemented by a call to an SCA module. In PD, we create a new artifact of the type called the Advanced Integration Service (AIS).



A wizard is started which prompts us for a name for this new component:

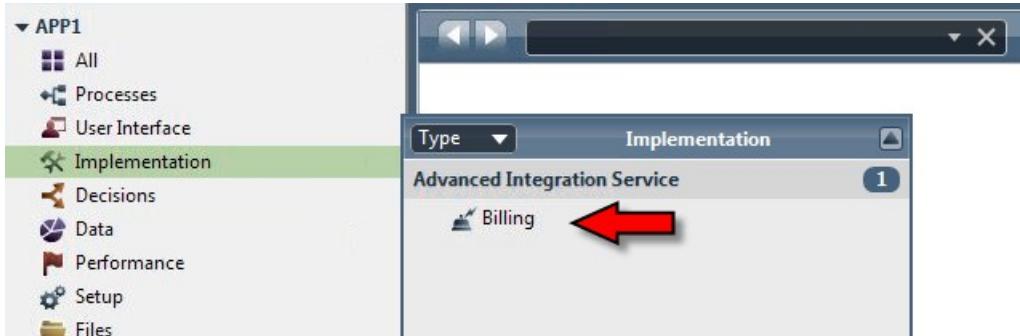


Once the name has been entered, we can then supply additional details for this AIS instance:

Common		Advanced Integration Service	
Name:	Billing	This service is implemented and deployed into the Process Center using IBM Integration Designer. The advanced integration service can be used like any other service.	
System ID:	guid:56d35d2f21c8d0e54743712:13b7cc71155:-7fdc	Module name:	
Modified:	admin (Dec 8, 2012 5:13:51 PM)	Export name:	
Documentation:	Click Edit to add or edit text.	Operation name:	
(Edit)		Can be used with service?: Yes	
		Open in Integration Designer	
Parameters		Parameter Details	
<input checked="" type="checkbox"/> Parameters <input checked="" type="checkbox"/> Input <input checked="" type="checkbox"/> name <input checked="" type="checkbox"/> amount <input checked="" type="checkbox"/> Output <input checked="" type="checkbox"/> confirmationId <input checked="" type="checkbox"/> Error	Add Input Add Output Add Error Remove Move Up Move Down	Name: confirmationId Documentation: Click Edit to add or edit text. <input checked="" type="checkbox"/> Is List: Parameter Type: <input checked="" type="checkbox"/> String System Data Select... New...	

Primarily, these are its expected input and output parameters.

Once the AIS panel has been completed, we find we have a new artifact available to us in the Process Application library under the Implementation category:

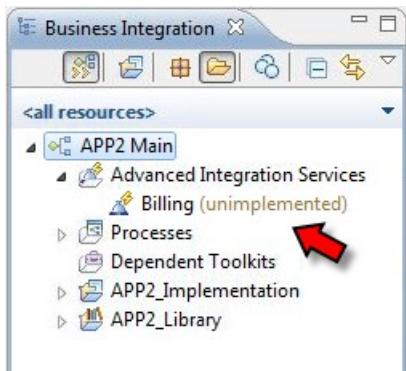


In the BPD, we can now associate the Billing AIS with the BPD activity we also called Billing:

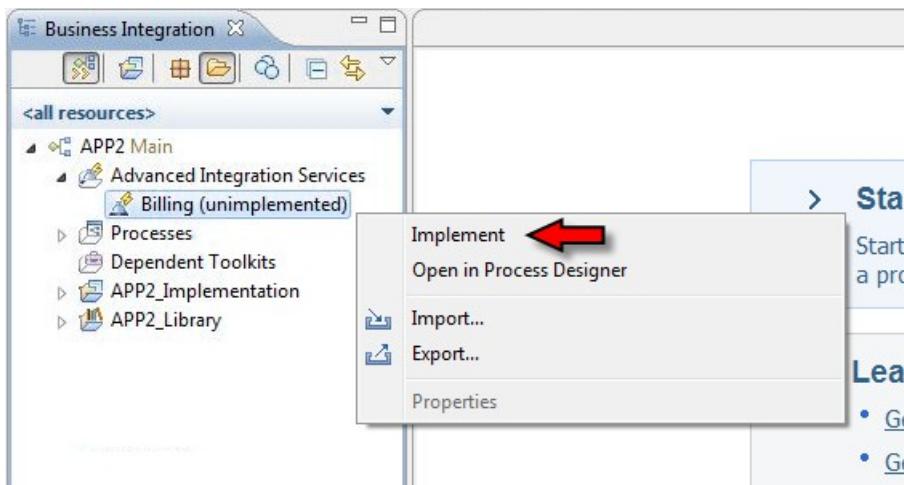


With all of these PD changes made and saved back to Process Center, we can now open an ID session.

In ID, we see the Billing AIS that was defined in PD as being un-implemented.



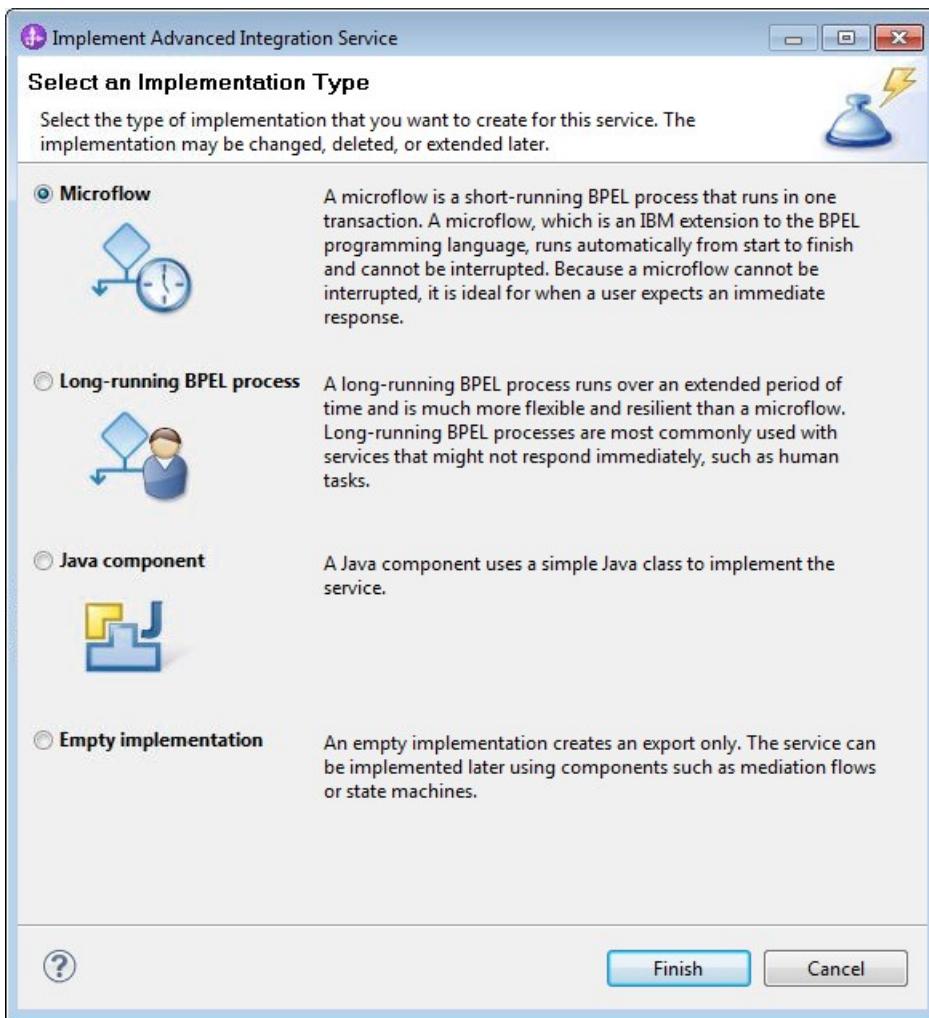
We can now right-click on this and select "Implement":



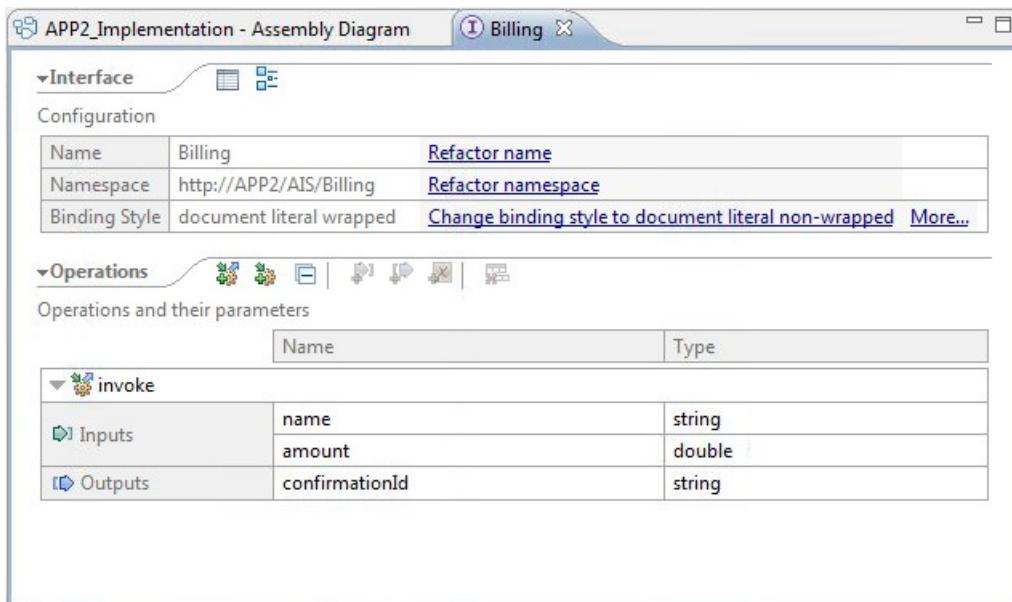
A wizard appears asking us for what we would like the base implementation to contain. The choices available to us include:

- Microflow – A BPEL process flagged as "short running"
- Long-running BPEL process – A BPEL process flagged as "long running"
- Java component – A Java component
- Empty implementation – No implementation suggestions made

Selecting "Empty implementation" is my recommended selection. This is the most flexible and allows us to take control from the outset for what we want to achieve.

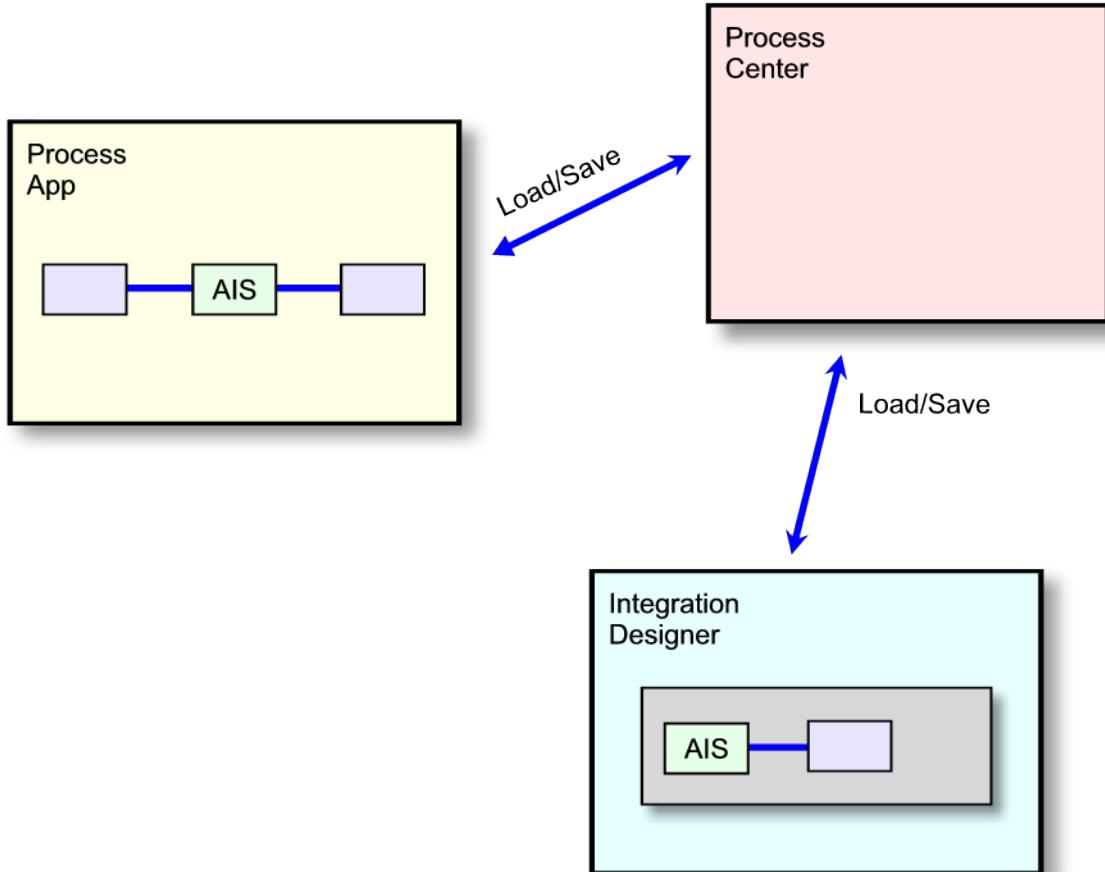


At the conclusion of these steps, an SCA interface is created for the Billing service:



and the Assembly Diagram is updated with an SCA Export for the Billing function. Once the implementation in the SCA Assembly has been built, we can publish the solution back to Process Center which will now include the SCA implementation. If we now run the Process App, we will see control being passed to the SCA module when the billing component is reached. This simple

example has achieved a *top-down* implementation of an AIS. We say top-down because the *nature* of the service was described in the high-level BPMN diagram in the BPD while its concrete (lower-level) implementation was handled in the ID. Schematically, the following shows what we have achieved:



A bottom-up solution can also be achieved by creating an interface and exposing that interface in an Assembly Diagram with an Export with SCA bindings. It appears that the name of the AIS created has the same name as the Interface. This doesn't feel right and is being investigated further.

When ID is started for the first time, we are prompted to define a connection to Process Center:

Process Center Login

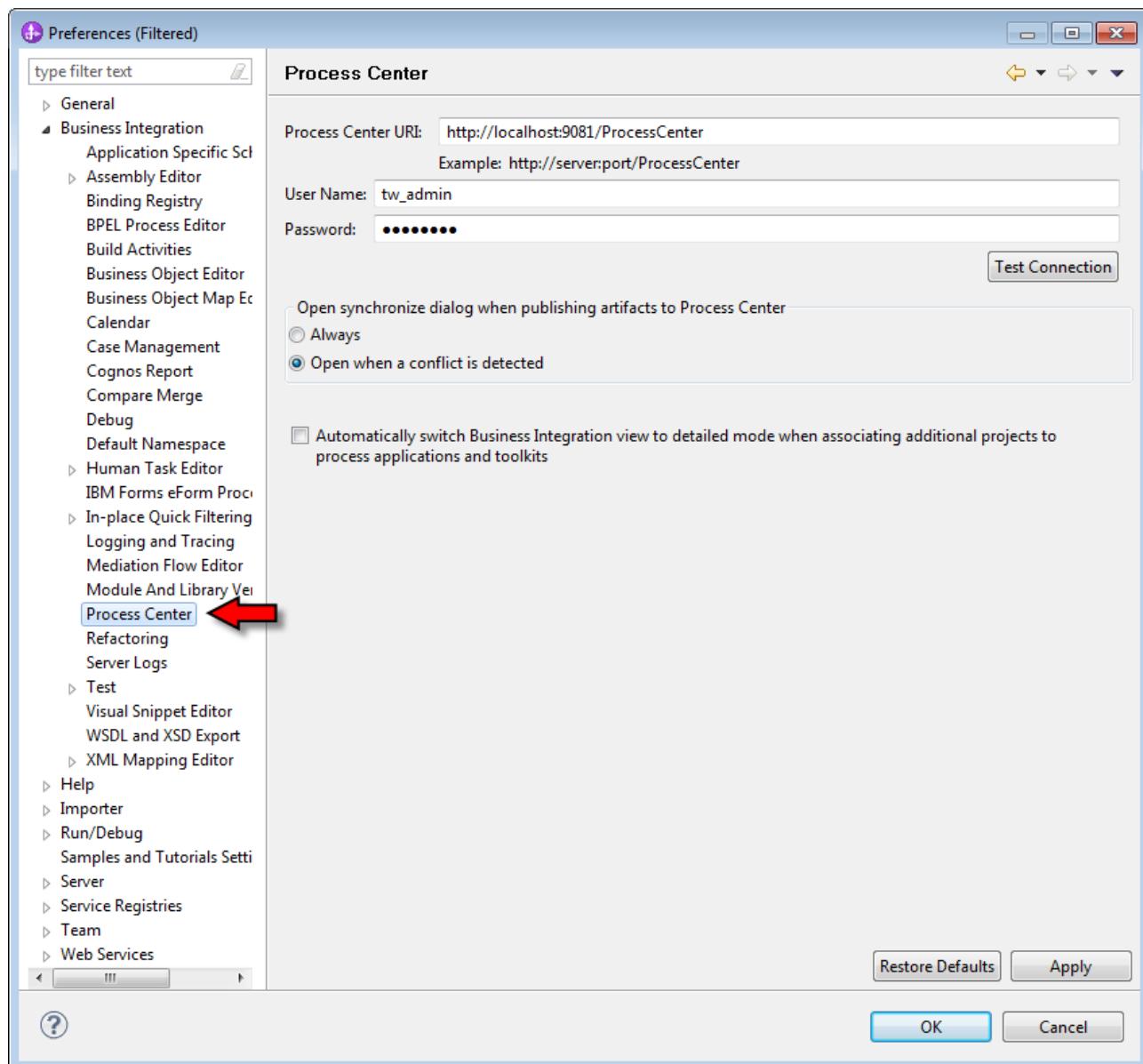
Log in to start working with Process Center or cancel to switch to the Business Integration perspective.

Process Center URI:
Example: <http://server:port/ProcessCenter>

User Name:
tw_admin

Password:

To see the currently selected Process Center URL open the ID preferences and navigate to Business Integration > Process Center. The URL and userid/password to access Process Center can be found there.



See also:

- Advanced Integration Service
- BPEL and AIS
- DeveloperWorks - [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 1: Introduction and setting up the databases](#) – 2013-03-06
- DeveloperWorks - [Developing a transactional Advanced Integration Service with IBM Business Process Manager, Part 2: Defining the business process](#) – 2013-03-13
- DeveloperWorks - [Linking business processes and enterprise services together using IBM Business Process Manager Advanced](#) – 2012-09-26
- DeveloperWorks - [The benefits of using IBM Business Process Manager Advanced](#) - 2012-06-20

Interface Types Exchange

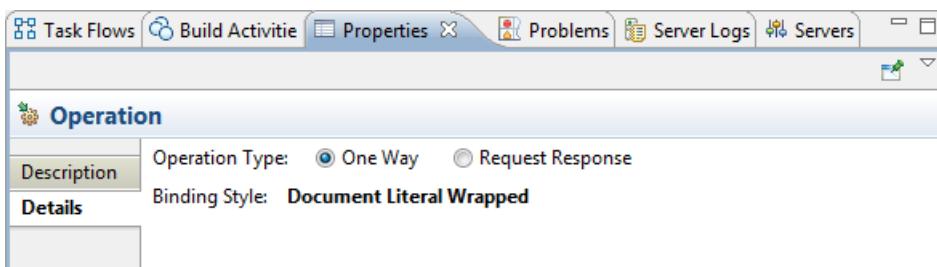
When we define a new AIS in PD what we are in fact doing is describing a new named interface that can be called from BPM that is implemented by SCA. The input, output and error definitions defined in PD show up as a new interface in ID. However, some thought is required on this.

- An SCA interface has a name associated with it. This is mapped from the name of the AIS defined in PD.
- An SCA interface has one or more operations associated with it. These are mapped to the single name called "invoke" as a single AIS is a single operation.
- An SCA interface has a name space associated with it. This is mapped to the form:
`http://<ProcessAppAcronym>/AIS/<AIS Name>`

In addition, an AIS that only has input parameters is mapped to a two-way operation in ID. For example:

Name	Type
invoke	
Inputs	p1
Outputs	string

If a one-way operation is required, ID provides the capability to re-factor the interface to be one-way. Clicking on the operation and looking at the Details tab, we can choose the operation type:



Data Types Exchange

In PD, data types are defined as Complex Data Types. For example.

The screenshot shows the 'Business Object' configuration screen for an 'Order' object. The top navigation bar includes icons for back, forward, and close, followed by the title 'Order'. A circular icon with numbers 1, 2, and 3 is in the top right corner.

Common tab (selected):

- Name: Order
- System ID: guid:56d35d2f221c8d0e:-27f5ee2:1309efc5:
- Modified: tw_admin (Jun 17, 2011 2:11:04 PM)
- Documentation: (empty text area)

Behavior tab:

- Definition Type: Complex Structure Type

Parameters tab:

- Defined parameters: name (String), type (String), amount (Integer).
- Action buttons: Add, Remove, Up, Down.

Parameter Properties tab:

- Name: (empty text field)
- Is List: (checkbox)
- Variable Type: <none> (dropdown with 'Select...' and 'New' buttons).
- Documentation: (empty text area)

Advanced Properties and **Advanced Parameter Properties** tabs are also visible at the bottom.

When an ID Workspace import is created, the corresponding Business Object for these data types is created.

The screenshot shows the 'Business object' configuration interface. It includes sections for 'Configuration' and 'Definition'.

Configuration section:

Name	Order	Refactor name
Namespace	http://MSG1	Refactor namespace

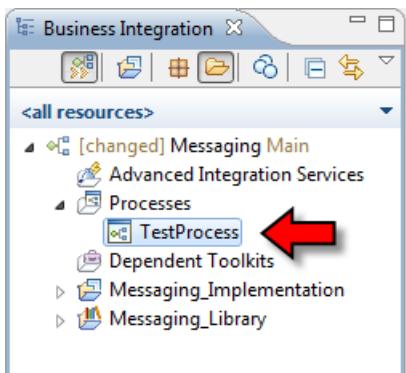
Definition section:

- Definition Type: Order (represented by a folder icon).
- Action buttons: +, Up, Down, X, List.
- Parameter list:
 - name string
 - type string
 - amount int

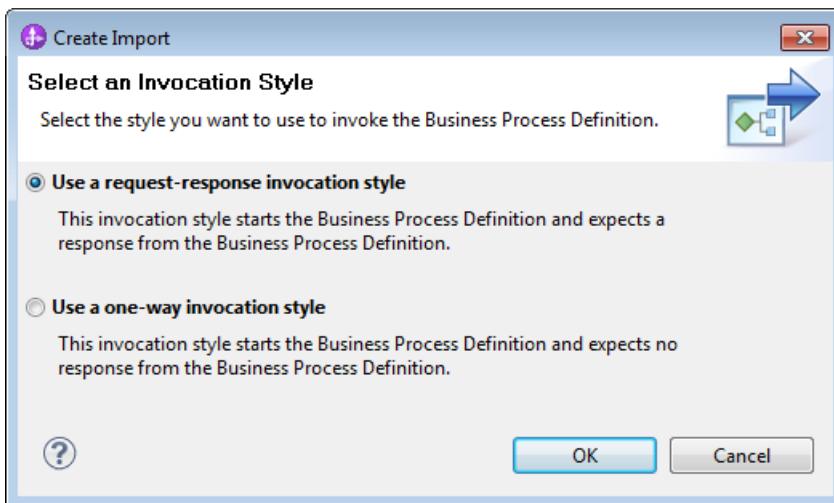
Note that the name of the Business Object is the same. The Namespace on the business object is, by default, the same as the Acronym value for the Process App that defines the data type.

BPDs exposed to an AIS

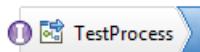
Each BPD process in a Process App appears in the Processes section of the Business Integration view of ID.



These processes can be dragged to an Assembly Diagram which will generate an SCA Import allowing the process to be invoked. For each process, **two** interfaces are created. One for a request/response call and one for a one-way call. When the process is dropped onto the assembly diagram, a dialog appears asking us which interface type should be used:



Once added to the diagram, the import looks as follows:

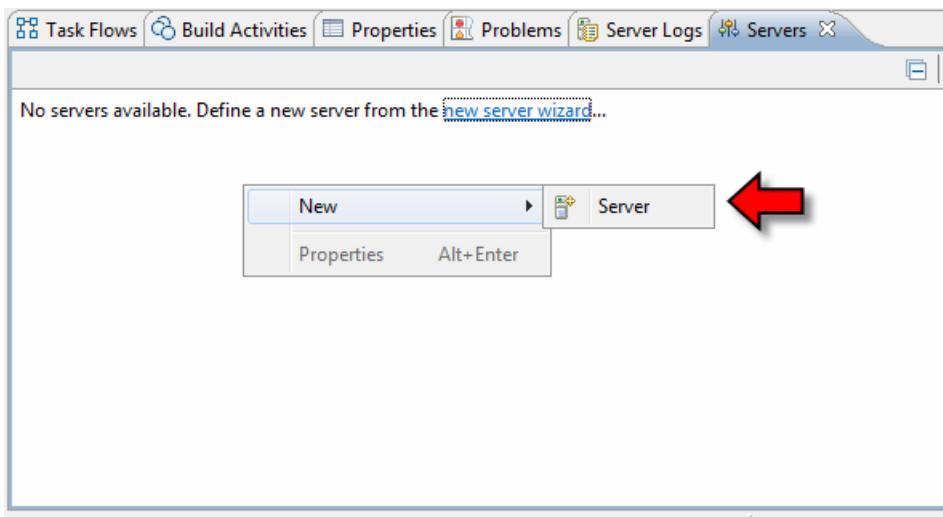


Invoking this import will result in an instance of the BPD being created and started. Since this otherwise behaves like a normal SCA import, we have all the power that SCA provides in terms of wiring and usage.

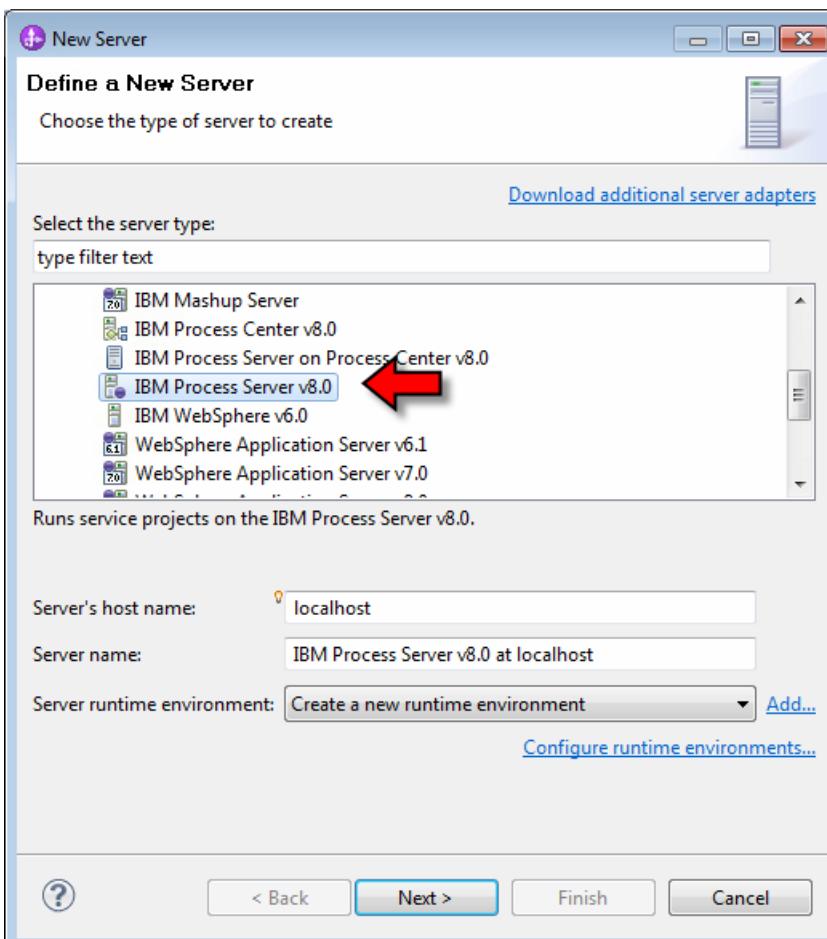
It should be noted that a BPD exposed to be called from an SCA assembly diagram is going to be invoked asynchronously. What this means is that any caller of this SCA Import that expects the provider to be synchronous will fail. This shows us (for example) if one adds an SCA Export with Web Services bindings (over HTTP) and wires that into the BPD SCA Import. A Web Service Export will always be invoked synchronously and will not be allowed to connect to an asynchronous provider.

Deploying for testing from ID

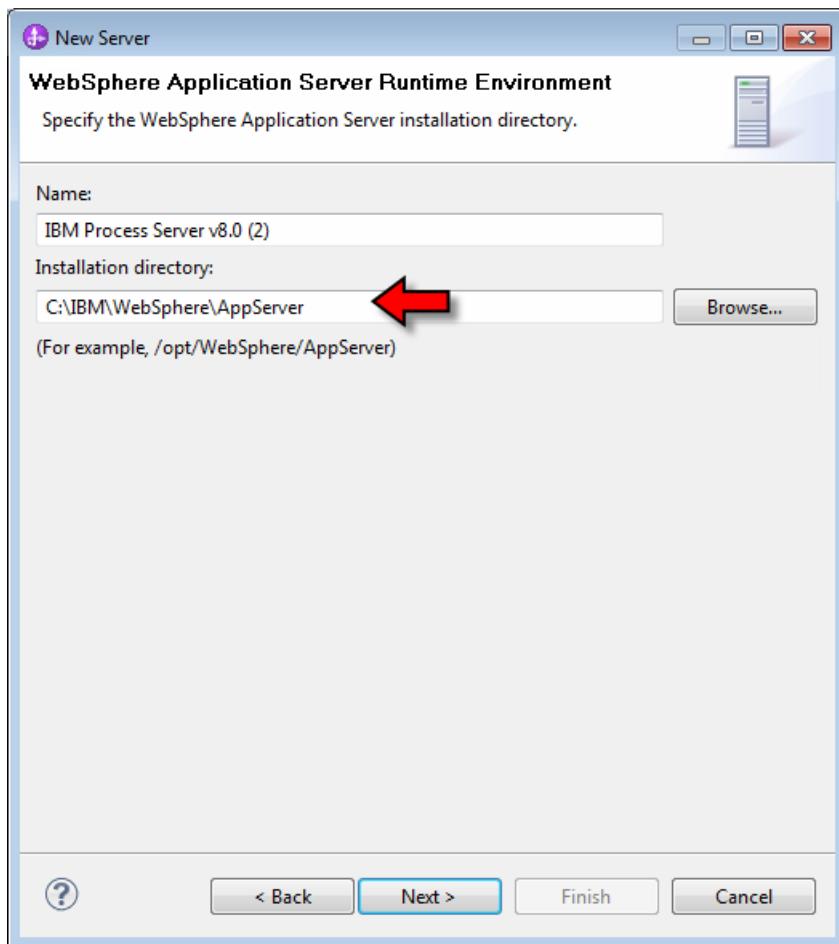
To deploy a module for testing, we must define a Process Server definition in the servers tab. Right click in the blank space for a context menu. Select New > Server.



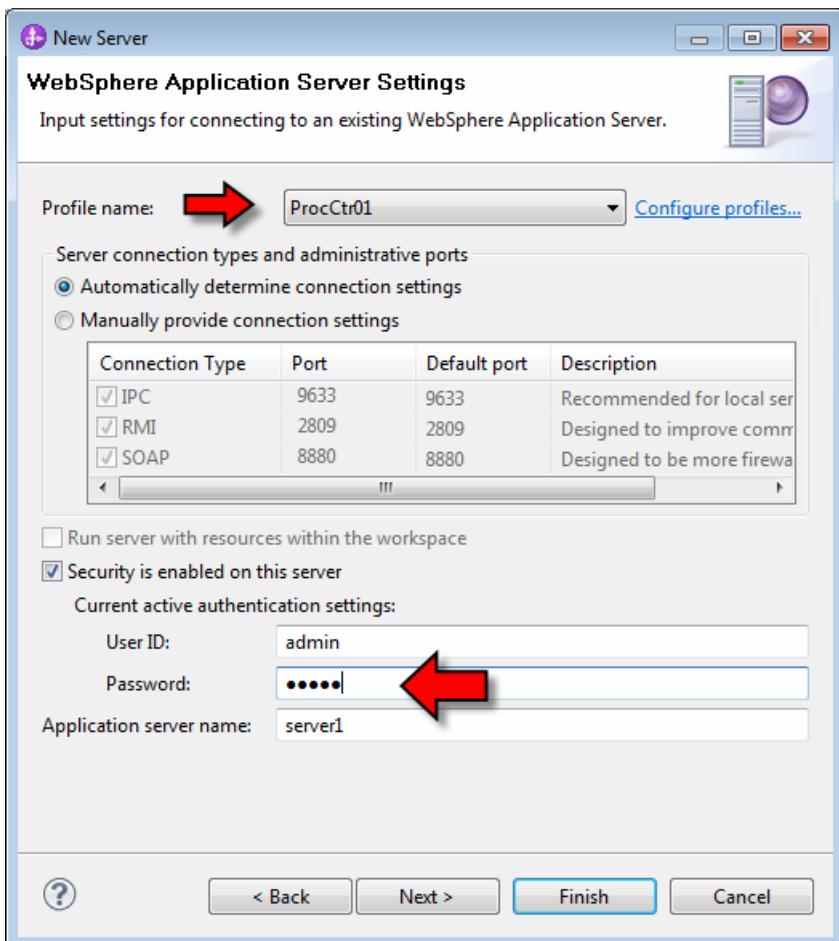
From the wizard, select "IBM Process Server v8.0". I recommend doing this irrespective of whether or not we are connecting to a Process Center or a Process Server. Both have worked just fine for me.



Next we must tell IID where the local installation image of Process Server can be found. We point to the AppServer root directory of the BPM install:



Finally we select which profile we will use and also enter the associated userid and password for connections.



Returning faults from an AIS

In principle a called AIS can fail for a variety of reasons such as a system error (a service it depends upon not being available) or a business error (insufficient funds to perform the debit). In these instances, what we would like to happen is for the AIS to raise a fault and for that fault to be captured by the calling BPMN process. Unfortunately, in 7.5, the InfoCenter documentation explicitly states that interfaces with faults are simply not supported. This appears to raise a dilemma. One possible solution is to add an extra parameter to the interface that will be used to hold a return code from the execution of the AIS. After the BPMN process invokes the AIS, it should then check the value of this response and handle appropriately. This is not as elegant as being able to use the BPMN Intermediate Exception mechanisms but it does appear to be a solution.

In v8 of the product, this issue is claimed resolved.

Manually un-deploying an SCA module

When we deploy a Process App to a Process Server, the SCA module associated with the project starts up. On occasion, we may wish to stop or un-deploy such a module. This can be achieved through the WAS admin console in the Applications > SCA modules area:

View: All tasks

- Welcome
- Guided Activities
- Servers
- Server Types
- Applications
 - New Application
 - Application Types
 - SCA modules
- Services
- Resources
- Security
- Environment
- Integration Applications
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

Cell=my4Node01Cell, Profile=ProcCtr01

SCA modules

This page shows all installed Service Component Architecture (SCA) modules and their associated applications. SCA modules encapsulate services, so you can make changes to services without affecting users of the service. To use the SCA module services you start the associated application.

Preferences

Start Stop Install Uninstall

Select Module Version Application Status

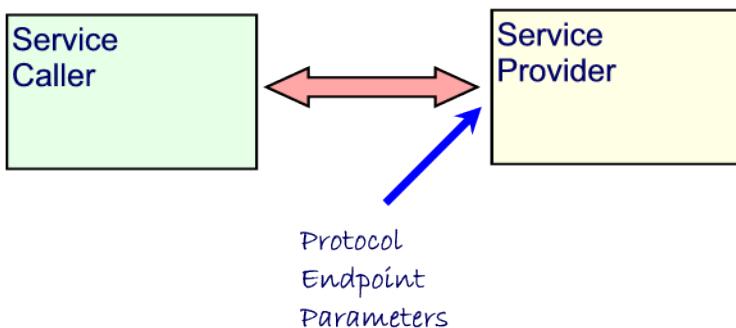
You can administer the following resources:

Module	Version	Application	Status
TestMod		AISTST1-Tip-TestModApp	
App01_Implementation	SN2	APP01-SN2-App01_ImplementationApp	
App01_Implementation		APP01-Tip-App01_ImplementationApp	
BFMIF_myNode01_server1		BPEContainer_myNode01_server1	
HTMIF_myNode01_server1		TaskContainer_myNode01_server1	
HTM_PredefinedTaskMsg_V7500_myNode01_server1		HTM_PredefinedTaskMsg_V7500_myNode01_server1	
HTM_PredefinedTasks_V7500_myNode01_server1		HTM_PredefinedTasks_V7500_myNode01_server1	
Messaging_Implementation		MSG2-Tip-Messaging_ImplementationApp	
Manufacturing		ORDERS-Tip-ManufacturingApp	
Manufacturing		STEW01-Tip-ManufacturingApp	

Total 10

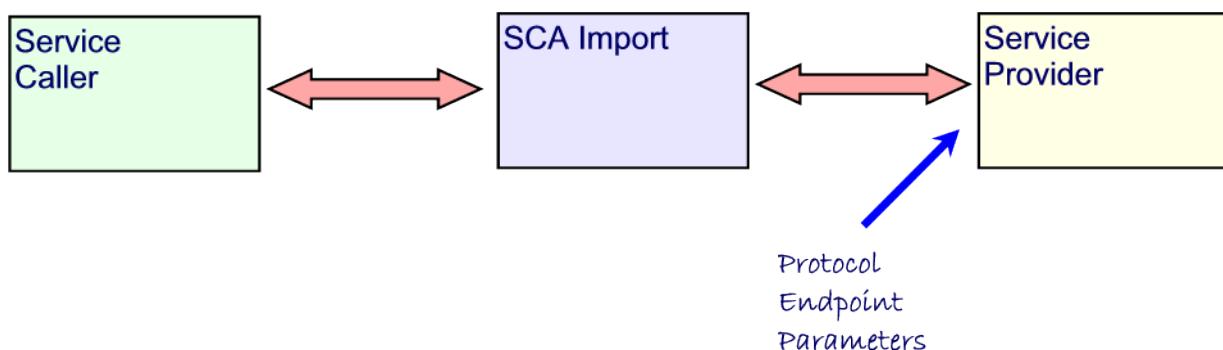
Service Component Architecture

Service Component Architecture (SCA) is a framework for solving one of the most basic issues relating to building distributed SOA applications. Imagine you have an external application that exposes itself as a callable service. You now wish to write a new application that calls the external service. How do you go about doing this? You could look at how the external application is to be called and you will find that it will be likely be Web Services, JMS, MQ, REST, EJB or some similar technology. You could then code your new application using the correct API and all will work. Well ... it will work for a time. You have introduced a hard dependency here. You have bound your application to a specific protocol and associated endpoint for the remote application you are calling.



If the nature of the service provider application changes such as its location, its communication protocol or its parameters, then the service caller will also need to be reworked. This means that the coupling between the caller and the provider is tight. Given that it is our desire to be loosely coupled and agile to change, maybe we can do better.

At its most simplistic level, SCA provides an abstraction between a service caller and a service provider. The service caller doesn't actually *care* about *how* the provider is called only that when it is time to invoke the services of the provider that the provider is then called. SCA provides just such a loose coupling. Instead of the service caller invoking the service provider directly, the service caller asks SCA to invoke the service provider. In turn, SCA then makes the actual call to the service provider using the the configured protocol, endpoints and parameters. At first glance, this doesn't appear to solve any problems ... but if we look closer we something wonderful has happened. The service caller which is customer written logic now no-longer needs to know mechanical information about the service provider. Instead, the caller asks SCA and SCA does the work on behalf of the caller. At the SCA level, the binding of protocols and endpoints are configured at a very high level without coding. Importantly, if the details of the service provider change, the service caller application (which if we remember, is business logic) is unaffected. Only the binding details at the SCA level need to be reworked.



The question now becomes one of how to actually use this SCA concept. I'll start by saying that

SCA is rich in function. This means that there is a lot of *stuff* in there. However, try not to let the Boeing 747 cockpit array of levers and switches deter you from realizing the benefits. We will take it slowly and carefully and expose the parts that are needed as when we need them.

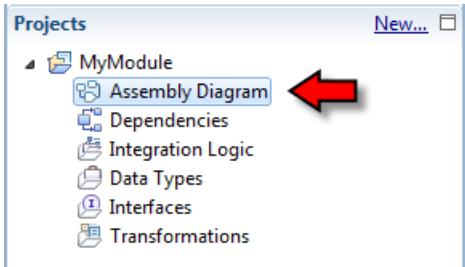
Let us start with what SCA actually is from a product perspective. SCA is a runtime framework that separates callers from called services. A caller no longer calls the target service directly but instead asks SCA to call the service on its behalf. To achieve this a "proxy" is put in place of the target service. This proxy is configured with the *actual* knowledge of how to invoke the real target service. The proxy is identified with a name. When the caller wishes to invoke the target service, it asks SCA to invoke the target by passing in the name of the proxy. We have thus decoupled the caller from the target service provider.

Somewhere the rubber must meet the road and some definitions have to be made. SCA asks for these definitions as an XML document that describes/declares the definitions. The format of the XML is defined as an XML Schema and conforms to an SCA specification called the Service Component Definition Language (SCDL). Thankfully, this XML is merely an academic part of the story as we will *never* see it. I explain it here only for your understanding. The SCA XML is (for our discussion purposes) interpreted at run time. When a caller says that it wishes to invoke a service with a name of "XYZ", the SCA runtime will parse the XML looking for the definition of "XYZ". This definition will include such things as what protocol to use to connect to the back-end service, where the service is located and a host of additional (an optional) attributes.

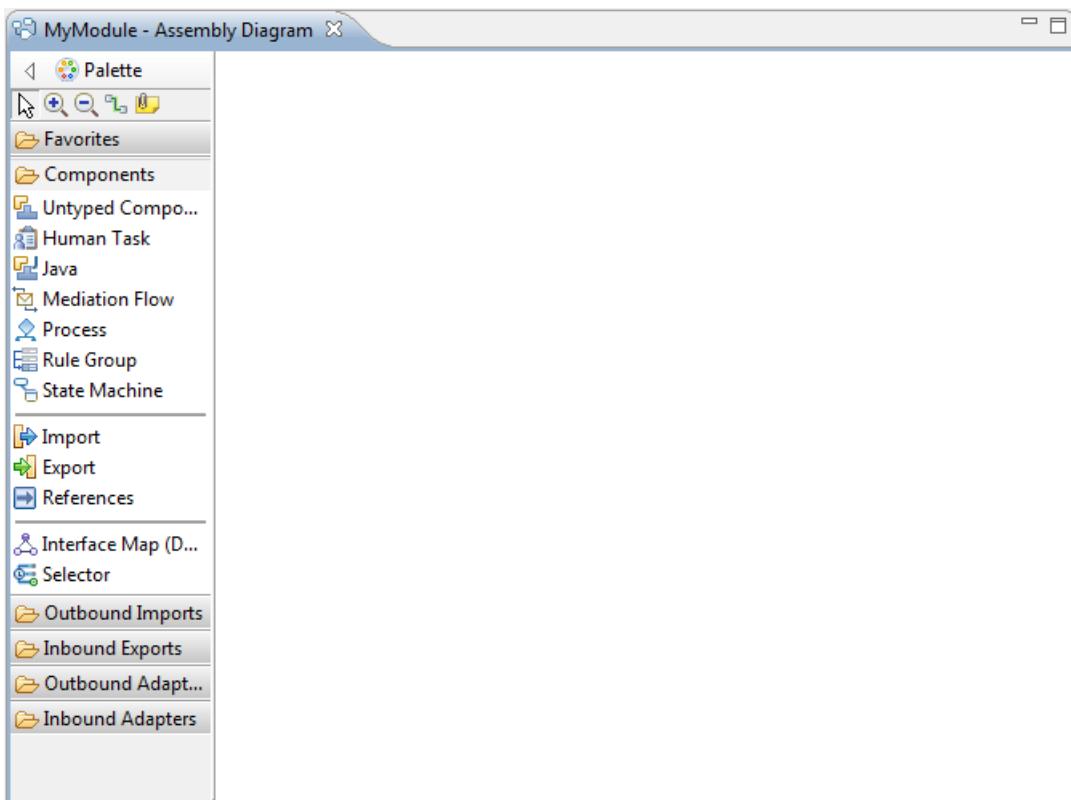
I mentioned that we will never have to work with the SCA XML directly, so how then do we describe the environment we wish to achieve to SCA? The answer to this is to use the IBM development environment called Integration Designer (ID). Integration Designer is a full IDE previously called WebSphere Integration Developer (WID). It is based on Eclipse and looks familiar to other Eclipse based products.

Within ID, we create modules which are really projects. Each module contains a description of a single SCA environment, the piece parts that are to be integrated and details of how those parts are configured together. The description of the SCA environment is called the *Assembly Diagram*. ID provides a dedicated editor for creating and modifying the contents of this diagram. Imaginatively, the editor is called the Assembly Diagram Editor. Using this visual editor the relationship between components can be wired together. As you may have already guessed, the diagram is basically a visual representation of the SCA XML document that is used to control SCA operations. The Assembly Diagram Editor is rich enough that users need *never* edit the raw XML files by hand. In fact, ID goes out of its way to hide those from you so that all you ever see is the assembly diagram.

The Assembly Diagram can be opened from the module view.



Once opened, a drawing canvas can be seen which shows the diagram area:



This is the mechanical aspect of working with SCA, now we turn our attention to the things that can be wired together on that canvas. Experience has taught me that this can be one of the more subtle concepts to get across, so we will take our time here.

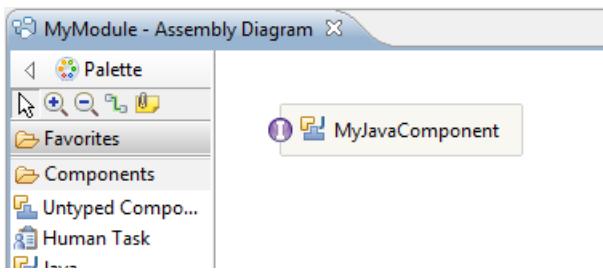
Within the SCA story there are basically two types of *things* that can be described. One type is code that will actually run on and within IBPM. This includes BPEL processes, Java code, and a variety of other goodies we have not yet had a chance to discuss. To help set the scene, imagine that we want to code a fragment of Java that will execute inside IBPM and that we want this Java code to call an external service. The Java code will be written and contained inside what SCA calls a *component*. A component is a unit of *thing* that is self contained and acts as a place holder for its implementation. From an SCA perspective, a finite set of SCA component types are available.

Specifically, these are:

- Java Component – A holder for Java code that is user written
- Process (BPEL) Component – A holder for a BPEL process that is user written
- Human Task Component – A holder for a user interaction using the Human Task Manager engine
- Mediation Flow Component – A holder for data conversion and handling. The core of an ESB.
- Rule Group Component – A holder for business rules that provide simple business rule functions.
- Selector Component – A holder for time based selection of control flow routing.
- State Machine Component – A holder for a process described as a Business State Machine that is user written.

We won't go into details on each of the component types here. Instead, there will be much written on them in subsequent sections. For now, let us simply realize that components are building blocks in the SCA diagram.

As an illustration, let us look at a Java Component. When added to the canvas area, it looks as follows:

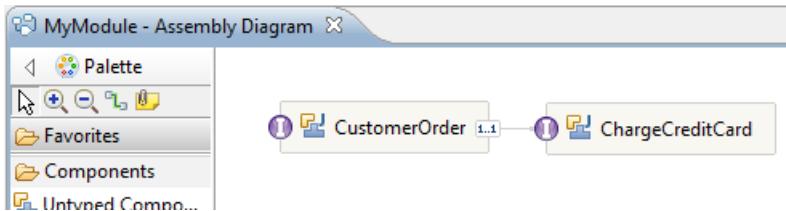


Each of the component types has a unique icon associated with it.

Every SCA component has a name that is unique to the module in which it lives. In this example, the component is called MyJavaComponent. On the Assembly Diagram, the visual box that is the component can be thought of as a container or holder for its actual content. We can drill down into a component to see what is inside it. In this instance, if we open it up, we would see Java code. If we had a BPEL process, we would see a BPEL process.

Now we get to take the next leap. Imagine that we have created a number of components on our Assembly Diagram where each component has a discrete and self-contained purpose. This is not yet a solution ... what we would have created would be a set of building blocks that we could construct our solution from. To build the solution we would need to connect the building blocks together to perform a bigger task.

For example, if we had a component that charges a credit card and we had a component that receives a customer order, we may wish for the order processing to invoke the services of the credit card charging. To describe this to SCA, we can draw wires from one component to another. This is illustrated in the next diagram:



It is important to understand that at the SCA layer, we are **not** describing flow control. Instead what we are saying here is that the component called CustomerOrder **can** (if it chooses) call another component called ChargeCreditCard. We are **not** saying that it will, must or does ... simply that if it wants to, it can. Going further, the CustomerOrder component may invoke a service (another component) that ships product to a customer. The assembly for this may look like:



SCA Interfaces and References

Every SCA component is capable of being invoked (called) by another. This means that the component has to be able to describe what it is capable of doing and what it expects as input should it be called. Consider a component that charges a credit card. What does it expect as input? These inputs are not prescribed by the caller, they are prescribed by the implementation of credit card processing function or service. An example of input might be:

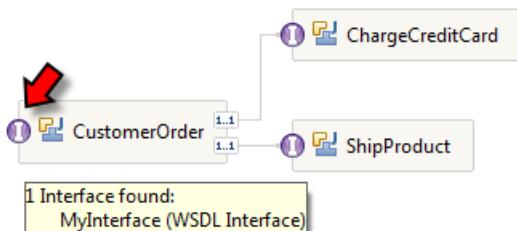
- Credit card number
- Owners full name
- Billing address of the credit card

These parameters are not negotiable. In order to use the credit card service, they must be provided. This can be thought of as a contract between the credit card service provider and anyone who may want to call it. Another analogy may also help to make this concrete. In the Java programming language there is a concept called a Java Interface. The Java Interface describes the methods and parameters of those methods as well as the return types. The Java Interface does **not** describe how the methods are implemented ... that is up to the Java programmer to decide. The Java Interface does however describe the relationship between a calling Java Class and an implementation of a class that conforms to the Interface.

This latter example is close to what we find in SCA. Every component in SCA exposes an Interface that describes the operations that can be requested of it. Each operation describes its expected input parameters and the nature of the parameters returned. It needs to be quickly stated that the Interface is **not** described as Java ... that would be too low level. Instead the interface is described in a (arguably) complex language called WSDL (Web Services Description Language). However ... IBPM ID hides the WSDL from you and exposes a full function interface description editor that removes all the complexity.

Let us review what we know so far. When we want to create an SCA component, we will build an interface that abstractly describes the contract that the component will expose which will include one or more operations. Each operation will have one or more input parameters and return zero or more output parameters. When we create an SCA component, we define the *type* of component that it is (eg. Java, BPEL, Mediation) and we associate an interface with that component. When we finally get around to implementing the component, both the type and the interface associated with it will govern the nature of the implementation.

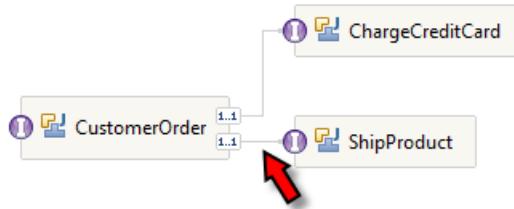
In the Assembly Diagram, the purple circle on the left with the capital letter "I" contained within is used to represent the interface possessed by the component. If we hover the mouse over the interface a pop-up appears showing us the name of the interface implemented by the component.



Another, but related concept associated with an SCA component is known as the *Reference*. The Reference is the indication on an SCA component that, during the operation of that component, it may call out to invoke the services of another component. An SCA component need have no reference associated with it. This would imply that the implementation of the component is fully self contained and it needs no further instance to do its work. However, if the component does need to call other components then it must have one or more references attached to it. A reference is again described by an interface description. Unlike an interface used by input into the component, the reference interface is used to describe the interface it expects to find should it need to call out to another component. Think of interfaces and references like plugs and sockets.

The interface that a component provides is like a socket and a reference that connects into that component is like a plug. If we sit and think about this for a while, an elegant and important concept comes to mind. Since one component provides an interface and another component that wishes to call the first provides a reference, unless both components agree on the same description of the interface it is *impossible* for the two components to be wired together accidentally. SCA enforces this desirable policing.

To wire two components together, the Assembly Diagram Editor allows us to draw a visual link between the two components.

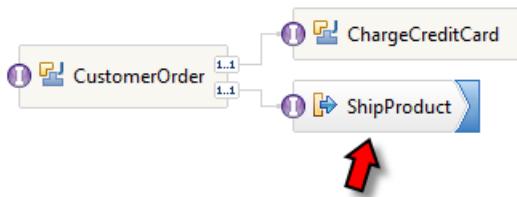


This clearly illustrates to us the relationship between them.

There is one further idea here that I want to bring to the surface. When a component wishes to invoke another component, the calling component sends the request **not** to the other component directly, but instead passes the request to its own local reference. At this point, SCA kicks in and the Assembly Diagram is consulted. Whatever a components reference is wired to is where the request will be sent. This means that the calling component is *loosely coupled* to the component that provides the service. At any time, the developer can rewire the relationship between a service caller and a service provider to point to a different service (with the same interface) and this will be honored. Another way of saying this is that calling SCA component has no idea at development time who, what or where the request will be delivered to. All it sees is the contract advertised through the interface. This is SOA in its ultimate form.

The SCA Import Component

So far we have touched upon one SCA component calling another where both components are hosted inside the SCA framework. In practice, this is rarely sufficient. Most solutions involve services that are hosted outside of a single SCA environment. For example, there may be a service that processes credit card billing that is exposed as a Web Service somewhere else in your organization. You may want to perform a ZIP code lookup which is owned by an external agency. The service you are calling may be hosted by CICS on a mainframe or accessed via WebSphere MQ. Simply calling from one SCA component to another inside the SCA environment is not sufficient. Fortunately, SCA comes to our aid yet again with an additional concept called the *SCA Import*. The SCA Import can be thought of as a proxy for communicating with a service that exists outside of our local environment. On the Assembly Diagram, it looks as follows:



Notice its distinctive icon. Just like other SCA component, the SCA Import exposes an interface definition. This means that it can be wired as another component's SCA reference. Unlike other components, the SCA Import does **not** have a native implementation. Instead, when it is called, it is responsible for building a network request in a specific format and protocol and transmitting that

request to the external system. When a developer adds an SCA Import to the diagram they are also responsible for *binding* that component to an external system. Putting it another way, the act of binding performs the following:

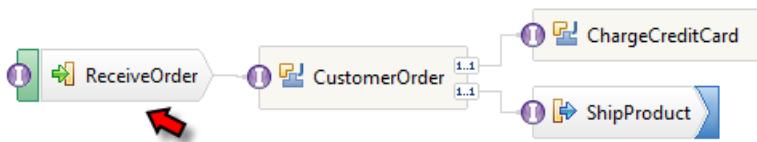
- Names the communication protocol to use such as Web Services, JMS, MQ, REST, EJB or others
- Names the endpoint information such as the location of the target service. This might be a machine name/port number, an MQ queue manager and queue or a JNDI name. The endpoint information is specific to the type of protocol selected in the binding.
- Names the data encoding (if applicable). Some bindings such as MQ and REST do not have absolute data protocol formats. Instead there must be agreement between the sender and receiver on the format of data to be passed. This may be XML, JSON, comma delimited, COBOL copybook or others. Depending on the protocol type selected for transmission, the developer needs to describe the physical format of the payload data that is to be built and received by the Import component.

There is much more to be said about the SCA Import component. Each of the different protocol types has its own story and parameters and these will be described later but for now it is sufficient to understand that the SCA Import is a proxy to a real external service. Once again we see that through the loose binding of SCA wiring with its interfaces and references, the caller has no idea that the request is going external. When the caller sends its request, that request surfaces at the reference and depending on how the developer has wired the diagram, the request will flow to the target component. If that component happens to be an SCA Import, then the configuration of the SCA Import will be honored and the request transmitted externally to the target. If at some later date, the target changes location or other nature, the SCA Import need only be reconfigured and all will be well. The business logic of the caller need have no knowledge of changes to the target. Again, a perfect "separation of concerns".

The choice of name for the proxy has always caused confusion. Why is it called "Import"? The relatively simple answer is that we are importing the services of an external provider. More on this naming later after we speak about SCA Exports.

The SCA Export Component

There is one further SCA component that we must address before we go too much further. This component is called the SCA Export. From a diagramming perspective, it looks as follows:



Again, note its distinctive icon. The purpose of the SCA Export is to expose an entry point into a module so that an external caller can invoke the logic contained in the module. We have already seen that an SCA assembly can be used to wire together various pieces of function to achieve a business goal but we have skipped the idea of how the components contained within the diagram are initially started. Components contained in an SCA Assembly Diagram are *not* exposed to be called directly from outside the diagram. What happens inside the SCA diagram is private (from the callers). To allow external access, we insert an SCA Export component. Just like other components, it has an interface associated with it. It is this interface that is exposed to the outside world. In addition to having an interface, the SCA Export always has a reference attached to it which is of exactly the same type as its exposed interface. This reference can then be wired to other down-stream SCA Components.

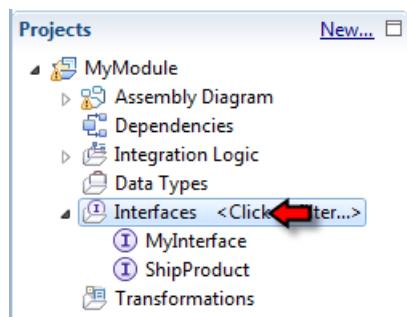
Similar to the SCA Import, the SCA Export acts as a proxy. When it is configured by the developer, it declares a communications protocol (Web Services, MQ, SOAP etc) that it is willing to listen upon. When the module containing the diagram is deployed for execution, the module is examined looking for export components. For each one found, the runtime automatically starts listening for incoming requests. When a request is received, a new instance of the module is started and control is given to the component to which the export is wired.

Once again we see loose coupling at work. An SCA component has no idea who has called it, only that it has been called. This means that the export component hides from other SCA components the details of how a request was received. A diagram can have multiple SCA Exports. Each different export could be bound to a different protocol allowing the business function as a whole to be exposed through a variety of technologies.

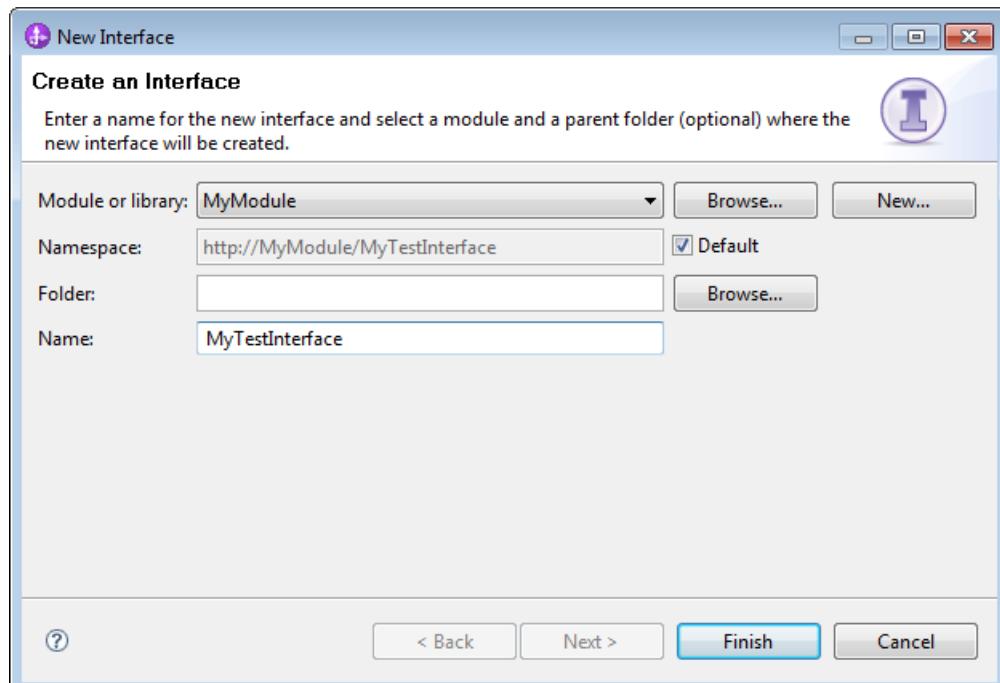
SCA Interfaces

When we started the discussion of SCA we quickly found that components have interfaces and may have references. Both of these are described using Interface descriptions. It was also mentioned that the interface descriptions are themselves WSDL files under the covers. Now it is time to look at how an interface is described to SCA.

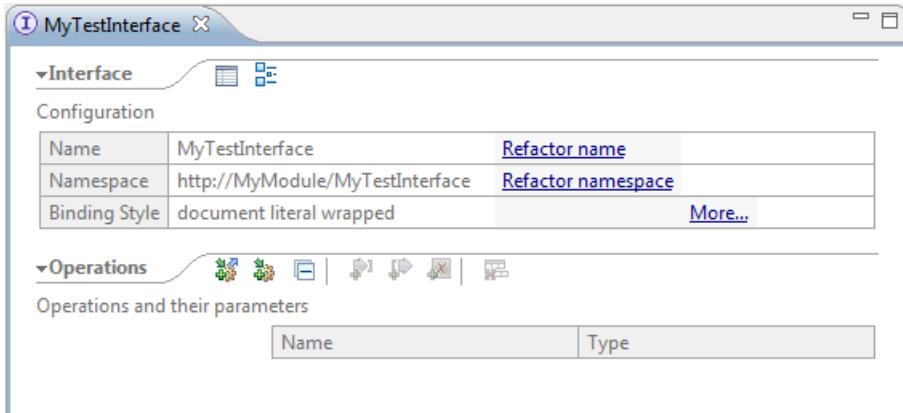
In a module, each interface defined is a named entity. When created or opened, the interface has its own editor called the Interface Editor.



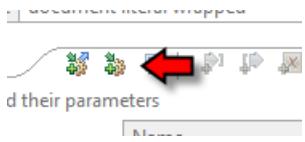
When we create a new Interface from scratch, a wizard page is displayed to us that looks as follows:



Within this page, we can enter the name that we wish to give the interface. This name combined with its namespace must be unique. Once a new interface has been created or an existing interface opened, the Interface Editor is shown.

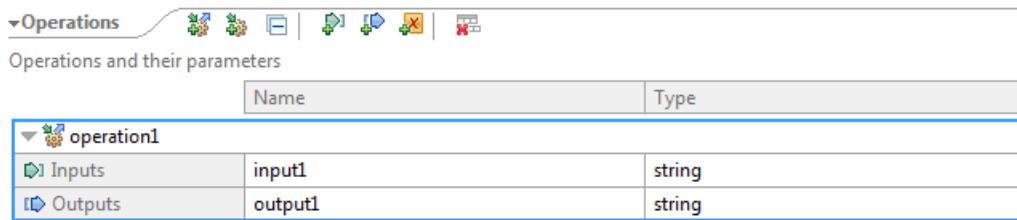


The editor shows two primary sections. One is an Interface section that shows the nature of the interface as a whole while the second section shows the operations exposed by that interface. Each operation is a possible entry point into the component described by the interface. When initially created, the interface has no operations defined to it. We can add operations by clicking on the add operation buttons:



There are two buttons corresponding to the two different styles of interface. One style is an interface that returns data while the other style has no return data. This second style is also called a *one-way operation*.

When an operation is defined, the properties for the operation can be changed.



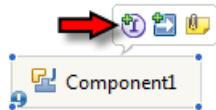
These include:

- The name of the operation
- The number of input and output parameters
- The names of the input and output parameters
- The data types of the input and output parameters

Once the operations have been defined and any changes made, the interface may be saved. Saving an interface results in the actual WSDL file that the interface represents being written. Again, under the covers the interface is described in the deeply technical WSDL language but yet we need never look at the interface from that perspective. In the vast majority of cases, we need only ever work with the Interface through the logical perspective as shown through the Interface Editor. Only the runtime execution cares that an interface is *actually* a mechanical WSDL file.

When we add a new SCA component onto the Assembly Diagram canvas, it has no interface or

references associated with it. If we hover the mouse over the component, a pop-up appears from which we can add and select the interfaces we wish:



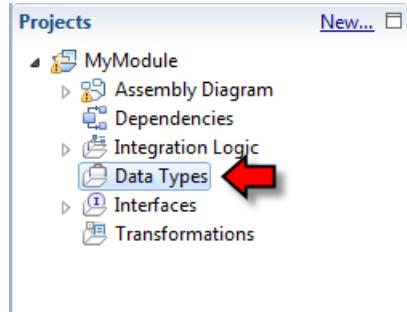
There is one button for adding interfaces and a second button for adding references.

SCA Business Objects

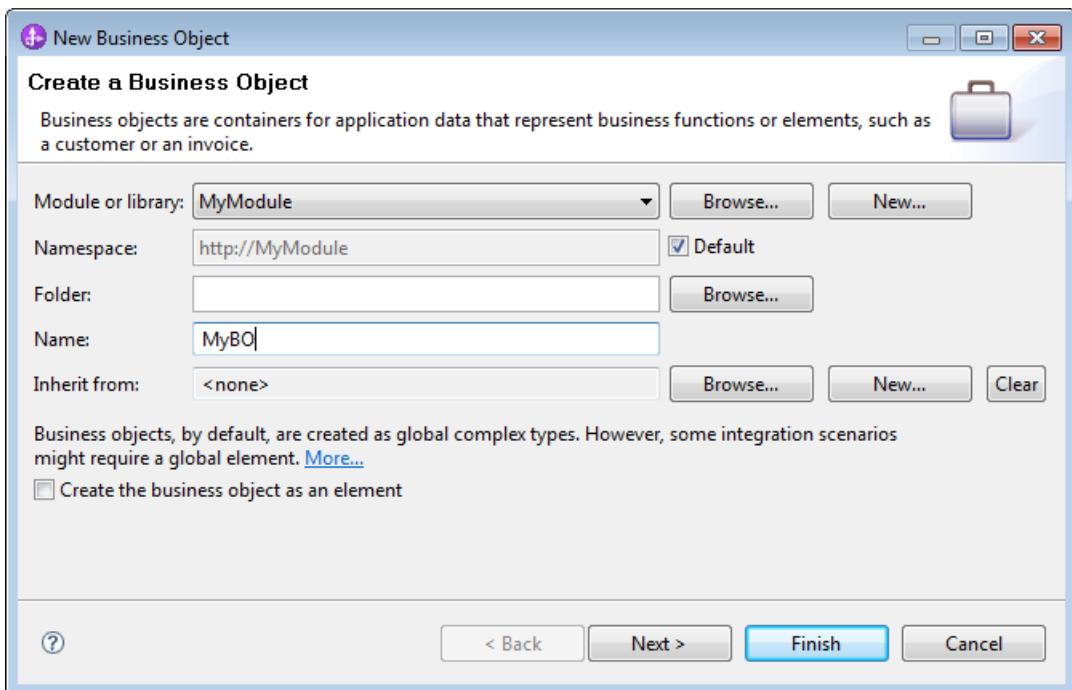
When we looked at SCA Interfaces, we saw that each operation in the interface may have multiple input and output parameters and that each parameter has a data type associated with it. The list of data types available include the usual suspects including strings, integers, floating points, dates, times etc. However, it is extremely common to want to create our own complex data types. These complex data types are collections of named fields which themselves have their own data types. For example, a customer will commonly have an address. We could create parameters on our interface for each of the expected items such as street, city, state and zip code but this is not as convenient as creating a single parameter that represents an address as a whole.

In SCA, these complex and structured data types are called Business Objects. Within the ID tooling, we can create our own named Business Objects and give them fields with names and data types that we desire. Under the covers, the Business Object is physically represented by an XML Schema Definition file but, just like with interfaces and WSDL, ID provides an elegant editor to hide this technical detail from us. The end result is an editor called the Business Object editor that provides all the capabilities we desire while at the same time hiding from us deeply technical constructs that in the majority of times, we have no use for.

Similar to Interfaces, Business Objects are first class entities and exist in the Data Types folder.



When a Business Object is created, a wizard allows us to enter its key characteristics. Core amongst these are the name of the Business Object (BO).



When the wizard completes for a new BO or if an existing BO is opened, the BO Editor is shown:

Name	MyBO	Refactor name
Namespace	http://MyModule	Refactor namespace

The BO editor shows a container (the BO type) into which the fields of the BO may be defined. Each field added can be given its own name and data type.



Fields can themselves be typed for other BOs and can also be typed as arrays of data. Once a data type is defined, that data type can be used within an SCA Interface definition:

Name	Type
operation1	
Inputs	input1
Outputs	output1

One of the primary purposes of SCA having Business Objects is to provide a normalized format of data passing between components. In addition, when raw physical data is received at an SCA Export, that data is parsed and a corresponding Business Object is constructed and passed forwards. This insulates other SCA components from having to concern themselves from the physical format of data sent by a service requestor. Conversely, when an SCA Import call is made and a parameter is a Business Object, the Business Object's data is serialized into a suitable physical format for outbound transmission. The Business Object's normalization of data moving around inside the SCA world make all of this possible.

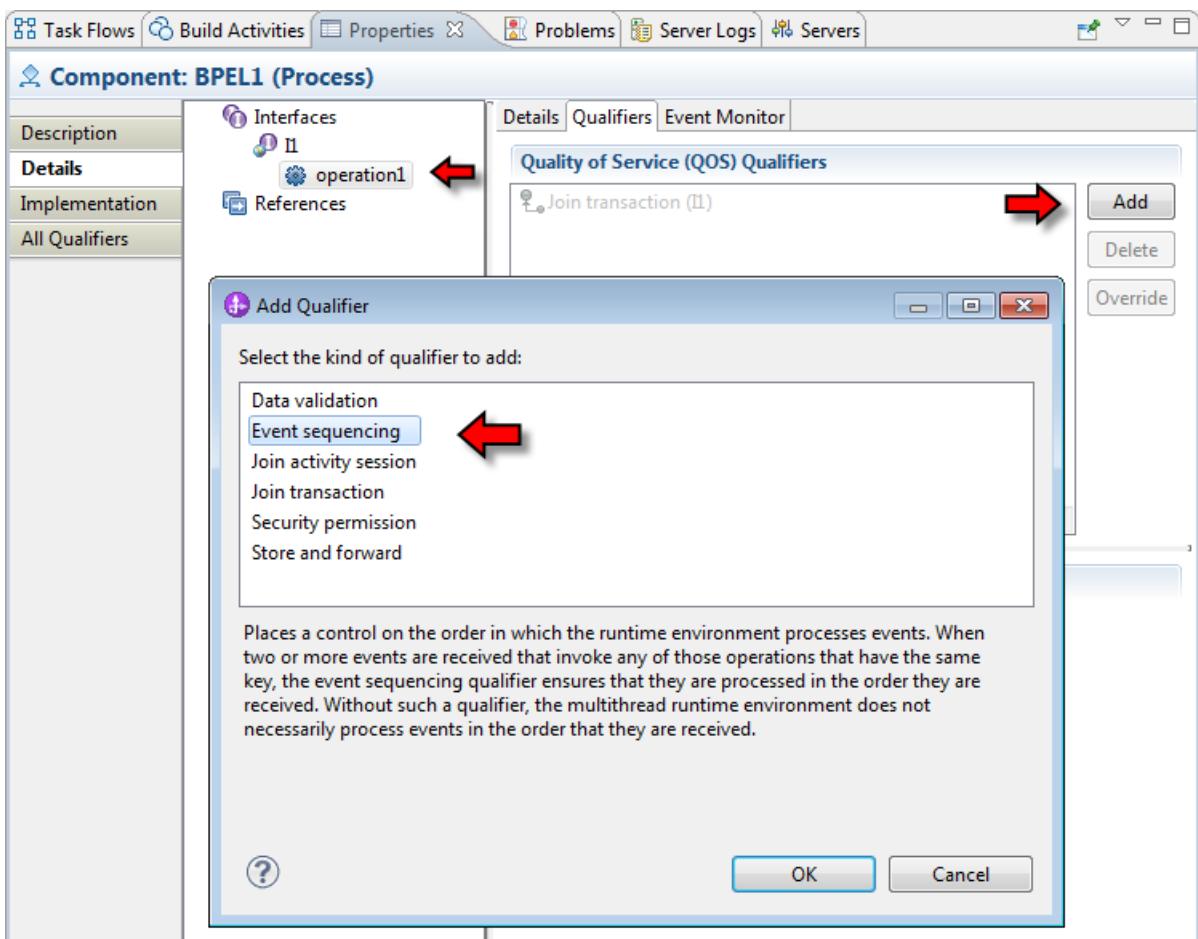
SCA Event Sequencing

The idea of event sequencing is to ensure that requests that arrive in a given order are processed in that order but **only** when this would make a difference. If it doesn't matter what order the requests are processed in, they can be executed in parallel. Consider for example two requests that arrive in order. The first request creates a bank account and the second request deposits \$100 in that account. If these two requests are attempted to be executed in parallel, it is conceivable that the action to add \$100 may take place before the creation of the bank account has completed.

Next consider a debit of \$50 from one account and a credit of \$75 to a different account. Since these accounts are different and there is no relationship between the requests, the activities can happily execute in parallel.

SCA Event Sequencing can be used to examine the data in an incoming request and, based on the content of that data determine if the request should be held until previous requests have completed or whether the request is eligible for immediate start.

Event Sequencing is enabled at the SCA diagram level. By selecting an interface on a component and then selecting an operation, we can add a quality of service and one of the options is Event Sequencing.



Once an Event sequencing quality of service has been added, we can then define the properties for this attribute:

This screenshot shows the 'Properties of Qualifier Event sequencing' dialog. At the top, it says 'Group name: DefaultGroup'. Below that is a table with columns 'Parameter name', 'Parameter type', and 'XPath expression'. There is one row defined: 'input1' of type 'BO1 : http://EventSequencing1' with an XPath expression '/key'. To the left of the table, there's a 'Key definition:' section with 'Reorder' buttons (up and down arrows). At the bottom of the dialog are several buttons: 'Add', 'Delete', and 'Edit' (highlighted by a red box), and a checked checkbox 'Process requests when error encountered'.

Here we define the parameters used as input of the service and the XPath expression to the field or fields that are to be used for sequencing.

The underlying implementation of Event sequencing is based on a number of WAS supplied applications and resources. Specifically:

- The table called PERSISTENTLOCK in the Common DB
- The Work manager resource called wm/ESWorkManager
- The WAS installed application called persistentLkMgr

See also:

- [Event Sequencing](#) - 2010-03-18
- [Event sequencing using WebSphere Process Server](#) - 2009-10-28

- [Differences between WebSphere InterChange Server and WebSphere Process Server support for event sequencing](#) – 2007-07-25

SCA Store and Forward

See also:

- DeveloperWorks - [Using the store-and-forward feature in WebSphere Process Server V7.0](#) - 2010-06-16

SCA Versions

SCA was an invention of IBM and some of her competitors. It was designed to provide interoperability between a variety of SOA players. SCA has become an industry standard and has undergone a variety of iterations. IBM's WebSphere Application Server provides an implementation of SCA as part of the base product, however, this is *not* the same SCA as is currently found in IBPM.

See Also:

- DeveloperWorks - [Integrating WebSphere Process Server and SCA Feature Pack, Part 3: Interoperability across SCA web services bindings](#) - 2010-09-29
- DeveloperWorks - [Integrating WebSphere Process Server and SCA Feature Pack, Part 2: Interoperability across SCA JMS bindings](#) - 2010-07-21
- DeveloperWorks – [Integrating WebSphere Process Server and SCA Feature Pack, Part 1: Interoperability across SCA bindings](#) – 2010-05-19

Installing an SCA Module through scripting

After having built an SCA module, we can export it as an EAR file for deployment. The deployment of the EAR can be accomplished through wsadmin style scripting. The command called "AdminApp" can be used to work with EAR applications. Full details of using this command can be found in the WAS InfoCenter.

```
AdminApp.install('<Path to EAR>')
```

An example of the output of the installation looks like:

```
wsadmin>AdminApp.install('C:/temp/SCAInstall.ear')
ADMA5016I: Installation of SCAInstallApp started.
CWLIN1002I: Creating the WebSphere business integration context
CWLIN1000I: Extracting the .ear file to a temporary location
CWLIN1007I: Initializing the WebSphere business integration context
CWLIN1005I: Performing WebSphere business integration precompilation tasks
CWLIN1001I: Compiling generated artifacts
CWLIN1006I: Performing WebSphere business integration postcompilation tasks
CWLIN1004I: Creating an .ear file from the temporary location
CWLIN1008I: Cleaning the WebSphere business integration context
ADMA5058I: Application and module versions are validated with versions of deployment targets.
ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
CWWBF0028I: Process components of SCAInstallApp have been successfully configured in the WebSphere
configuration repository.
ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
ADMA5081I: The bootstrap address for client module is configured in the WebSphere Application Server
repository.
ADMA5053I: The library references for the installed optional package are created.
ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
ADMA5001I: The application binaries are saved in
C:\IBM\WebSphere\AppServer\profiles\ProcCtr01\wstemp\Script13b19bdc42d\workspace\cells\win7-
x64Node01Cell\applications\SCAInstallApp.ear\SCAInstallApp.ear
ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
SECJ0400I: Successfully updated the application SCAInstallApp with the appContextIDForSecurity
information.
ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
```

```

ADMA5005I: The application SCAInstallApp is configured in the WebSphere Application Server
repository.
CWSACA3013I: Resources for the SCA application "SCAInstallApp" are being configured.
CWSACA3023I: The EAR file "SCAInstallApp.ear" is being loaded for the SCA module.
CWSACA3017I: Installation task "SCAModuleTask" is running.
CWSACA3017I: Installation task "Resource Task for SCA Messaging Binding and EIS Binding" is running.
CWSACA3017I: Installation task "Resource Task for SCA Messaging Binding and JMS Binding" is running.
CWSACA3017I: Installation task "SIBus Destination Resource Task for SCA Asynchronous Invocations" is
running.
CWSACA3017I: Installation task "EJB NamespaceBinding Resource Task for SCAImportBinding" is running.
CWSACA3017I: Installation task "SIBus Destination Resource Task for SCA SOAP/JMSInvocations" is
running.
CWSACA3017I: Installation task "Deployment Task for JaxWsImportBinding and JaxWsExportBinding" is
running.
CWSACA3014I: Resources for the SCA application "SCAInstallApp" have been configured successfully.
ADMA5113I: Activation plan created successfully.
ADMA5011I: The cleanup of the temp directory for application SCAInstallApp is complete.
ADMA5013I: Application SCAInstallApp installed successfully.
```

```

After making changes through this command, remember to call:

```
AdminConfig.save()
```

to save the configuration changes.

Once installed, the application can be seen from the WAS admin console. Applications installed through scripting are initially in the stopped state:

The screenshot shows the 'SCA modules' page in the WebSphere Admin Console. The left sidebar has a tree view with 'Applications' expanded, showing 'New Application', 'Application Types' (with 'SCA modules' selected), and 'Global deployment Settings'. The main panel title is 'SCA modules'. It contains a brief description: 'This page shows all installed Service Component Architecture (SCA) modules and their associated applications. SCA modules encapsulate services, so you can make changes to services without affecting users of the service. To use the SCA module services you start the associated application.' Below this is a 'Preferences' section with buttons for 'Start', 'Stop', 'Install', and 'Uninstall'. A table lists resources with columns for 'Select', 'Module', 'Version', 'Application', and 'Status'. The table shows five entries: BFMIF, HTMIF, HTM PredefinedTaskMsg\_V8000, HTM PredefinedTasks\_V8000, and SCAInstall. The 'SCAInstall' row has a red asterisk icon next to the status column, indicating it is stopped. A red arrow points to the 'SCA modules' link in the sidebar, and another red arrow points to the 'SCAInstall' row in the table.

| Select                   | Module                                               | Version | Application                                        | Status |
|--------------------------|------------------------------------------------------|---------|----------------------------------------------------|--------|
| <input type="checkbox"/> | BFMIF_win7-x64Node01_server1                         |         | BPEContainer_win7-x64Node01_server1                |        |
| <input type="checkbox"/> | HTMIF_win7-x64Node01_server1                         |         | TaskContainer_win7-x64Node01_server1               |        |
| <input type="checkbox"/> | HTM_PredefinedTaskMsg_V8000_(win7-x64Node01_server1) |         | HTM_PredefinedTaskMsg_V8000_win7-x64Node01_server1 |        |
| <input type="checkbox"/> | HTM_PredefinedTasks_V8000_(win7-x64Node01_server1)   |         | HTM_PredefinedTasks_V8000_win7-x64Node01_server1   |        |
| <input type="checkbox"/> | SCAInstall                                           |         | SCAInstallApp                                      |        |

# SCA Java Components

One of the component types available in an SCA assembly is a "Java Component". This is a Java class that can be used to implement an Interface. When the assembly calls the component, appropriate entry points in the Java code are called.

A Java component is an SCA component that has been implemented in Java. Thus, if you have an interface and you think the best implementation of that interface would be in Java, you can drag the interface onto your assembly diagram and choose to implement it in Java.

When doing this, a natural question arises: "if I am a component, I can have references. How do I access them?"

## *Calling a Reference*

When a Java component wishes to call a reference defined as WSDL, there are two cases. The first case is the easy one: the WSDL is defined such that the input and output of the operation is a DataObject. In such a case, the programming model is the SCA model we know and love.

```
Service myService = (Service)ServiceManager.INSTANCE.locateService("Name_Of_My_Reference");
DataObject inputObject = DataFactory.INSTANCE.create("Namespace_of_object",
"name_of_object");
DataObject result = (DataObject)myService.invoke("operation_name", inputObject);
```

But what if the WSDL operation hasn't been defined to take a DataObject but, rather, is specified as a collection of primitives? In this case, you know you need a DataObject (since that's what the "invoke" method wants) but you don't have one. The 'Type' support in SCA/SDO is the answer:

```
Service myService = (Service)ServiceManager.INSTANCE.locateService("Name_Of_My_Reference");
Reference ref = myService.getReference();
OperationType opType = ref.getOperationType("Name_Of_My_Operation");
Type targetType = opType.getInputType();
DataObject input = DataFactory.INSTANCE.create(targetType);
//
// input now points to a DataObject with an element for each input parameter on the interface.
//
```

## *Java Component Error Handling*

Java Components provide the unique ability to leverage the SCA programming model directly. This direct interaction allows a fine grained control of the client to service invocation. SOA can be described as a series of client and service provider interactions where a service can be a client for other services. Consider the following interaction

Component A calls component B.

Component B calls component C.

Component A is a client to the services provided by component B.

Component B is a service provider to component A and a client to component C.



When implementing Java components that promote a comprehensive error handling strategy there are a few factors to consider.

Defensive programming and programming with problem determination in mind encourage system and solution stability.

The problem determination strategy is an important subset of the overall error handling goal and should be consistent across all Java component implementation.

These considerations can be best described in the perspective of the client and the service provider.

## **Client Programming with Java Components**

The purpose of this section is to understand service interactions and how they relate to the client implementation.

There are two primary aspects of a client and service provider interaction. The operation type that the service provides and the invocation pattern that the client uses to invoke the service. Refer to the section call SCA Programming Model for a more detailed description of operations and invocation patterns.

Additionally, when Java components are implemented there are many opportunities for the component implementation to raise unexpected exceptions. For example, if the Java component implementation uses a variable that is not set this could raise a NullPointerException. The best practice should be to catch all exceptions prior to return. So as not to push the implementation of the Java Component beyond its interface.

### Synchronous Invocation Patterns

The try and catch blocks can be used to control the upstream propagation of expected and unexpected exceptions. Failure to handle exceptions can lead to unpredictable results and overall system instability.

Try and catch blocks should be used in the operation implementation. This will prevent the inadvertent propagation of business and run-time exceptions.

```
// Call the request response service and catch any exception
try {
 DataObject errorBOReturned = ei.reqresp(errorBO);
} catch (ServiceBusinessException sbe) {
 System.out.println("***** Business exception caught insync/reqresp");
 System.out.println("***** " + sbe.toString());
}

catch (ServiceRuntimeException sre) {
 System.out.println("***** Runtime exception caught in sync/reqresp");
 System.out.println("***** " + sre.toString());
}

catch (Exception e) {
 System.out.println("***** some other exception caught in sync/reqresp");
 System.out.println("***** " + e.toString());
}
```

The code sample above makes a call to the reqresp method of the service via a reference.

Any exception returned from this invocation will be caught and printed to the system out log.

Business exceptions should be thrown when the java component encounters an exception that was defined on the interface. Business exceptions must represent the signature of the fault/throws defined on the service interface.

There are a couple of ways to create and throw a service business exception in a java component.

The first way is to construct the business exception using a string.

```
ServiceBusinessException sbe = new ServiceBusinessException("***** Service Business Exception
constructed by string");
throw sbe;
```

The second way is to construct a data object that represents the BO defined by the service interface. Populate the data object and throw.

```
BOFactory bof = (BOFactory) sm.locateService("com/ibm/websphere/bo/BOFactory");
DataObject faultBO = bof.create("http://ErrorHandling/com/ibm/summercamp",
"FaultBO");
faultBO.setString("faultString", "faultBO fault string");
ServiceBusinessException sbe = new ServiceBusinessException(faultBO);
throw sbe;
```

## Service Run-time Exceptions

Run-time exceptions should be thrown when the java component has encountered an exception where the exception cannot be associated to a fault/throws defined in the interface.

Dependent on the invocation style used by the client, the run-time exception will be recorded as a failed event in the case of asynchronous type (one way, deferred response, and callback) or the run-time exception will be returned to the client to catch and handle or propagate further up stream.

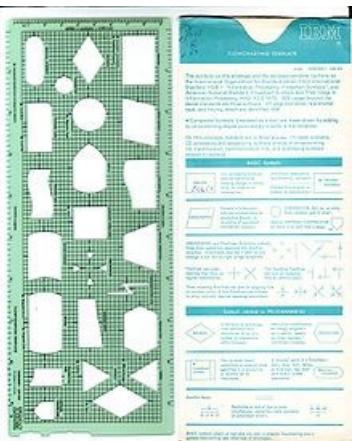
Service Run-time Exceptions are handled differently when the client is a BPEL component.

# BPEL - Business Process Execution Language

When the Web Services standard was being baked and shortly after its early adoption, the hopes for it were extremely high. Businesses looked to create reusable services across their enterprise. All looked good. As the services started to appear, applications were then written to build solutions from these by writing code that called the services when needed. As time passed, more and more services were built and patterns started to appear in the methodology of how solutions were built. Common discussions went along the lines of "First we call Service A, then we call Service B and depending on the outcome, we call Service C or Service D". In effect what we were seeing was the creation of the desire to aggregate services together to flesh out the solutions. The path taken to call the services tended to be linear with decision points along the way. Naturally this could be coded up in Java or another programming language, but this required coders, skills and it was not readily apparent by looking at the code just what was going on.

From this state of affairs, an idea was born. What if we could provide a descriptive language that was oriented towards invoking services in sequence? This language would accommodate calling services, would have variables and would provide decision points where different paths could be followed based on the values of variables and previous service calls. This was the beginnings of the language called Business Process Execution Language (BPEL).

At its most basic level, BPEL is an XML document that conforms to the BPEL prescribed XML Schema. The tags in the document have precise semantics associated with them. For example, a tag called <Invoke> causes a service to be called. By describing the sequence of steps to be executed in BPEL and by having a runtime engine that can understand BPEL, we effectively have a control language for invoking services in sequence. Effectively, we have created a "flowchart" like language (that goes back to the 1970s).



BPEL became an industry standard championed by OASIS and supported by many vendors. IBM brought a product to market that was called WebSphere Process Server (WPS) that acted as a BPEL runtime engine. That BPEL engine is what is now found in IBM Business Process Manager Advanced.

A BPEL process description could be written with a text editor if one knew the XML syntax for the language ... however ... that would be error prone and would basically have the similar maintenance and readability issues as was previously found by writing solutions in application code. IBM solved this problem by providing a sophisticated BPEL editor. This editor allows the user to build BPEL solutions visually. A canvas areas is shown and into that canvas, the various BPEL activity types can be dragged and dropped from a palette. The activities can then be visually wired together to describe the control flows. The end result is a flowchart style diagram which is actually a visual representation of the underlying BPEL. This BPEL editor is now a core part of the

IBM Integration Developer IDE used with IBM Business Process Manager Advanced.

With this brief background on BPEL, the remainder of this section will drill down into the BPEL language and how it can be utilized with IBPM.

## **BPEL Activities**

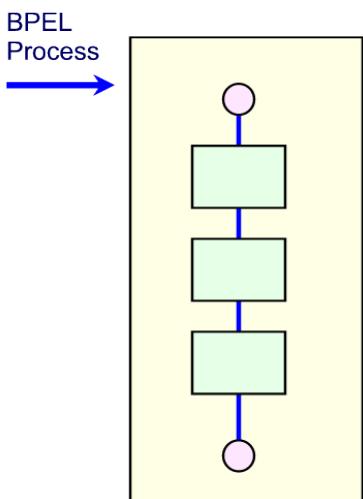
When one learns a new programming language (say Java for instance), you would be taught the basic constructs of the language. This would include variable definitions, assignments, loop syntax, function calling and others. Once you have an understanding of the grammar and can then make sense of a single line of code when you see it, only then can you start to go further and put together multiple lines of code to build a program. I am going to treat our learning and understanding of BPEL in the same way. First we will examine all the separate building blocks of a BPEL program and, once we are comfortable with those, then we can turn our attention to building complete solutions. First, we will look at the nature of a BPEL Process.

### **The notion of a BPEL Process**

Notice that the "P" in BPEL stands for "Process" and the "B" in BPEL stands for "Business". Arguably, BPEL was an early beginning to the BPM story. It was definitely a start in making the IT solutions that run a business much more comprehensible to the business process workers in such a company. But ... it could never have been written by business oriented staff. BPEL is quite definitely in the domain of needing IT skills to build and maintain. BPEL pre-dates BPMN (which was originally *just* a drawing language) and the hope was that it would bring business process owners and IT staff responsible for creating such processes much closer together. To some degree, that did indeed happen however because of the IT nature of BPEL, there was never truly a daily pollination between these two camps.

So where does that leave BPEL today? My opinion is that BPEL is still extremely relevant for the central core of what it was designed to do ... namely to invoke multiple external services in a specific order. Whether those services are business related (debit one account, credit a second account) or IT related (Add a row to the DB2 database, place a message on an MQ queue), the tactic is still the same. BPEL is IT centric while BPMN is business centric.

However one wishes to perceive BPEL, knowing what it is and what it isn't is extremely important to understanding all of IBPM. Part of understanding BPEL is to understand its history and some of its language and that brings us back to "Process". In BPEL, the most coarse grained entity we can build is called a "Process". Going back to our flowchart analogy, the process can be thought of as the flowchart as a whole. Before we can create BPEL activities to describe what we want to achieve, we must first create the process.



## Variables in BPEL

When we think of an Java application, we can think of it as a combination of control statements and variables. As the application runs, it executes sequences of statements that update and use the values of variables within the program. At any point in time, the state of the program is characterized by where we currently are in terms of the next Java statement to execute plus the values of the variables in effect. BPEL is not that different. Just like programming, BPEL has the notion of variables. A variable is a named entity within the process that has an associated data type. As the process executes a sequence of BPEL activities, these activities can retrieve or update the values of these defined variables. The data types of the variables can be any of the simple data types such as string or integer but they can also be types as Business Objects allowing a variable to represent a complex structure.

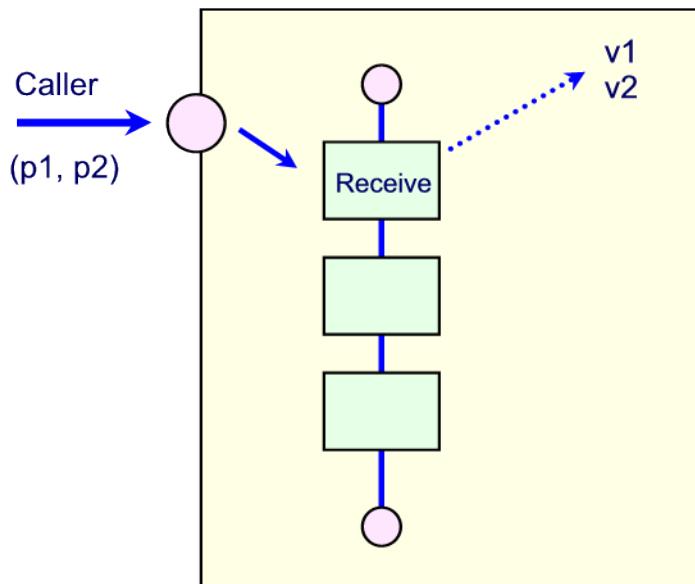
## Receive Activity



When a BPEL process template is created, that does not mean that there are now running instances of that process. Somehow instances of the BPEL processes has to be brought into existence. This is where the BPEL Receive activity comes into play. The Receive activity is commonly found at the start of a BPEL process template. Its purpose is to flag the starting point of the process. Associated with the Receive is an Interface and operation. The Interface is what the BPEL process exposes to the outside world to allow it to be invoked. If a caller invokes an operation on the interface, it is the Receive activity within the process that is given control.

When we think of an Interface and operations on that interface we should recognize that an operation can have input parameters. When we define a Receive activity we must also define BPEL variables that will be used to hold copies of the passed in parameters. When an external caller invokes the BPEL process, any parameters supplied by the caller can then be found in the variables within the process by subsequent activities following the Receive.

In the following diagram, we illustrate this idea in more detail. A caller invokes the exposed interface to the process and passes in parameters p1 and p2. The Receive activity is configured to be associated with the interface and operation and also to map the incoming parameters to local variables (v1 and v2). Following the completion of the Receive activity, control passes onwards to the next activity in the flow.

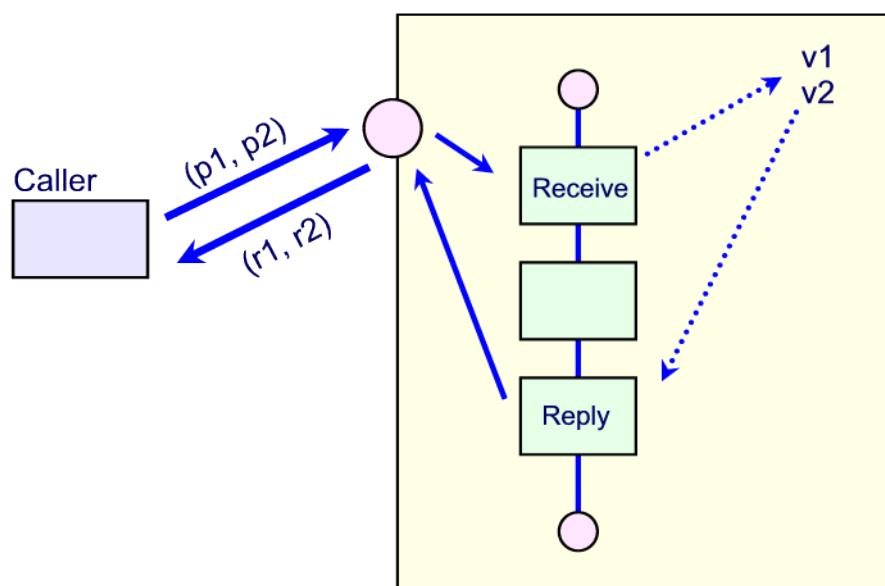


## Reply Activity



The Reply activity is closely related to the Receive activity. When a Reply activity is encountered in a process it sends a response back to a caller that previously caused a Receive activity to be executed. Similar to the Receive activity, the Reply activity maps variables to the expected response parameters of the call. When the Reply is reached, the values of the variables in effect at that time are used as the response values to the caller. Typically, processes conclude with a Reply activity but it is important to note that Reply does **not** indicate the end of the process. If there are additional activities following the Reply, these will be executed as normal. Only one reply can be sent to a previous caller.

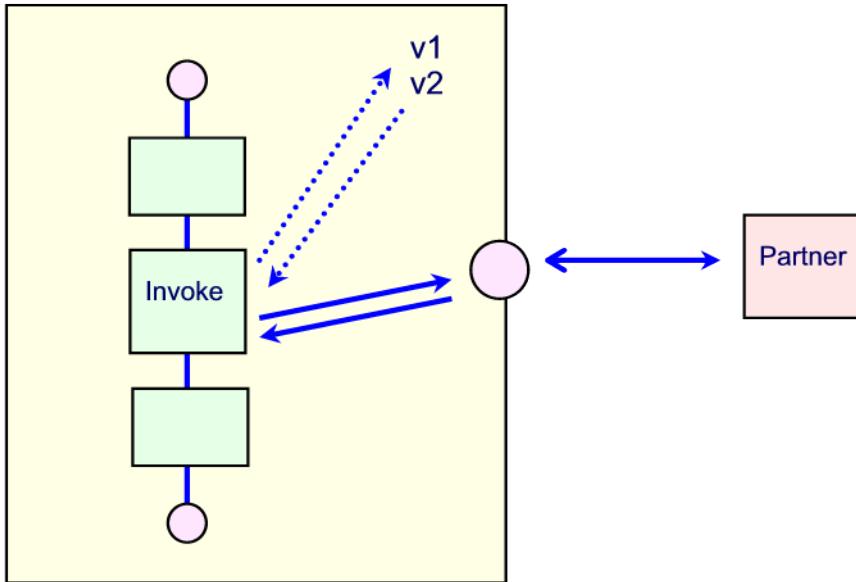
In the following diagram, we illustrate the use of the Reply activity. When Reply is reached, it is associated with the interface and operation that had a previous Receive. In addition, the values of the variables  $v1$  and  $v2$  are used to build the response values to the caller of  $r1$  and  $r2$ . Note that the variables used for reply don't have to be the same variables used in the Receive.



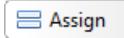
## Invoke Activity



The Invoke activity is arguably the heart of the BPEL language. Its purpose is to call outbound to a service that lives outside the process and wait for a response. When the Invoke activity is added to the process diagram, it must name an interface and operation that it wishes to call. In the likely event that the operation has input parameters, the Invoke activity names BPEL variables who's values will be used and passed to the partner when the call is made. For return parameters from the Invoke, these are also mapped to BPEL variables.



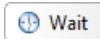
## Assign Activity



In a BPEL process there are variables. The values of these variables may have been set by input data when the process was started or set by returns from Invoke activities. It is not uncommon for us to want to set or change the values of these variables while the process is executing as part of the process. For example, an Invoke activity may call a service that calculates the cost of a customer's order while a second Invoke calculates the shipping cost. The result may need to be a total cost and hence we must update a variable used for the Reply with the sum of the two previous values.

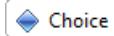
When added to the BPEL process, the assign activity allows us to specify expressions using the XPath language. We can assign values to the variables from either constant values or expression based on the values of other variables.

## Wait Activity

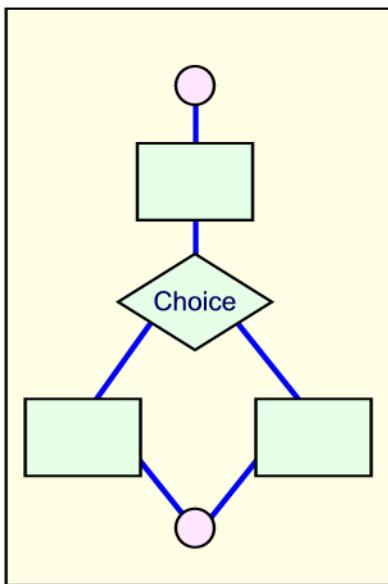


The Wait activity causes the process to suspend its own execution for a period of time. The amount of time can be described as either a relative value from when it is executed or a specific date and time in the future.

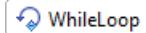
## Choice Activity



The Choice activity is a decision point mechanism within the process. When reached, a series of XPath based expressions are executed. The first of these expressions that evaluates to true then causes activities beneath that expression it to be executed. Associated with the Choice are one or more such expressions that are called "cases".



## While Loop Activity



The While Loop activity executes its child activities while a boolean expression associated with the loop remains true. It is assumed that the activities will cause the expressions outcome to eventually change by updating some variable's value where the expression uses that variable.

## Repeat Until Loop Activity



Similar to the While Loop, the Repeat Until Loop executes its child activities until some condition becomes false. Since the expression is evaluated after each iteration of the loop, the body will be assured to be executed at least once.

## For Each Loop Activity



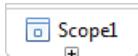
The For Each Loop activity executes its body some defined number of times. This can be a fixed loop or can be based on the number of elements in an array. The For Each Loop had a variety of additional options that can be used to control its operation.

## Parallel Activities Activity



The Parallel Activities activity is a container for other activities. When reached, it causes all the activities contained within to start executing concurrently with each other. The Parallel Activities container does not terminate until all of the contained activities have themselves terminated.

## Scope Activity



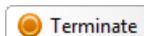
The Scope Activity is another container activity. When reached, it provides a new and nested environment in which its contained activities execute. Think of it as providing a "scope of control". Amongst its functions it includes the ability to define BPEL variables that are local in scope only to the Scope Activity. When the Scope Activity completes, any variables associated with that activity are deleted. In addition to variables other BPEL constructs can also be associated with the Scope such as fault handler for handling errors, event handlers for handling out of band events and compensation handlers for handling transactional characteristics.

## Generalized Flow Activity



Normally when activities are added to a BPEL process, one activity explicitly follows another. There is no actual visual wiring that need be done as inserting an activity after or between two existing activities implicitly declares where it will be executed in the sequence of work. On occasion, we may wish much finer grained control and choose to explicitly wire one activity to another. It is here where the Generalized Flow Activity comes into play. Activities contained within this container must be explicitly wired together to describe their order of execution.

## Terminate Activity



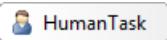
The Terminate Activity causes the instance of the process to come to an immediate conclusion. This activity is implicitly present as the last activity in a process and hence need not be explicitly included. When reached, the process simply and cleanly ends. The Terminate Activity is not an error indication.

## Throw Activity



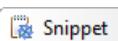
The Throw Activity results in an error condition being indicated. It is similar to the Java Language throw construct. How the caught error is handled is dependent on whether or not there are event handlers in place.

## In-line Human Task Activity



This activity is not currently part of the BPEL specification but is instead an IBM addition. When reached, it causes the BPEL process to suspend and a new work item to be created for human attention. The work item is created in the Human Task Manager engine which is distinct from the Human Task manager functions found in the BPMN engine.

## Snippet Activity



The Snippet Activity is another IBM extension to the BPEL specification. The Snippet Activity allows us to supply in-line Java Code that will be executed within the context of the process when it is reached. The activity has access to all the variables in the process both for read and write. By escaping down into Java code, virtually any algorithmics that need be performed can be achieved.

## BPEL Transactionality

Transaction processing is the notion that when a program or process runs, its outcome should be "all or nothing" and not some indeterminate state. Consider for example a process that transfers funds from one bank account to another. A simple implementation may first debit one account and then immediately credit another. But what if there is a failure (such as a power outage) following the debit but before the credit? The first account will have had funds reduced while the second will not have had the funds added. We can not simply re-execute the process because this would result in two debits to the first account and again we will be in trouble.

What is needed is the ability to react to a failure by undoing the work that was previously done so that the process can behave as though it had never started in the first place. If it had never started, then we would be safe to restart after a failure.

The classic IT solution to this problem is to use a paradigm called two-phase commit. In this story when an update is made to some data by a process, the data owning system *knows* that it is part of a recoverable transaction. A good example of such a data owning system would be a database. When data is changed, added or removed, it is not *really* changed. It is held in a pending state until either the transaction as a whole commits (completes successfully) or rolls back (indicates a failure). In the event of a commit, the changes are made permanent and can no longer be undone. For a roll back, the data owning system has the ability to undo any changes made as part of that transaction. At the conclusion of the roll-back, the data is returned to the state it was in prior to the transaction having started.

Two-phase commit has been around for a long while and is implemented as part of the WAS server. Updates to recoverable resources include databases, JMS, and Java EE resources such as EJBs. Unfortunately, there are some limitations to two-phase commit. In order for it work, all the participants in the transaction that can own or change data must be able to *know* that they are part of a transaction. This is commonly achieved through an industry standard transaction processing protocol called "XA". Typically, Web Service calls do not participate in transactions. Web Services have always been considered *stateless*. What this means is that when a Web Service call is made to a Web Service provider, upon conclusion of the provider processing the request, it forgets everything about the previous call and passively waits for the next one to arrive. A key concept of two-phase commit is that the data owning system remember previous interactions and be informed about the final outcome of the transaction. A second problem is that transactional updates to data

owning systems must be of short duration. Imagine if this were not the case. Consider debiting an account by \$100 and then not concluding the transaction for a long period of time. During this period another program or system may come along and ask for the value in the account. To return a value that is \$100 less than when the original process started would probably be a mistake as there is an opportunity that the monies will be returned if the transaction is rolled back. What commonly happens in these instances is that other users of the data are suspended until the transaction concludes one way or another. If a transaction takes a long time to complete (say many minutes or hours), the system can quickly grind to a halt and worse, deadlock scenarios can be introduced.

For both the preceding reasons, the BPEL standard does **not** support the two-phase commit protocol. This would seem to imply that BPEL can't support transactions. Fortunately, BPEL has come up with a solution. That solution is called *compensation*.

If we think back to the primary goal of transactionality which is to return the state of a system back to its original state that existed before the process started, in the event of a failure, we can perform an explicit undo operation to undo what was previously done. Back to our funds transfer example. If we debit an account by \$100 and then fail to credit a second account, if we perform a credit of \$100 to the original account, the end result will be that the original account is back to its starting value. We say that the original debit has been compensated. Compensation is the act of performing a secondary operation that reverses the first operation.

Within BPEL we can define what are called *Compensation Handlers*. A Compensation Handler is a set of additional BPEL activities that can be associated with other BPEL activities. In the event that a failure is detected, BPEL can automatically cause the Compensation Handlers to be executed. Each Compensation Handler would thus be responsible for undoing what was done previously.

## **BPEL and AIS**

When we create an AIS we are saying that when called it will invoke an SCA module. Within that module we can have a BPEL process. This means that we can invoke a BPEL process from an AIS. There are some very interesting effects that we can achieve with this combination.

## **BPMN compensations using BPEL**

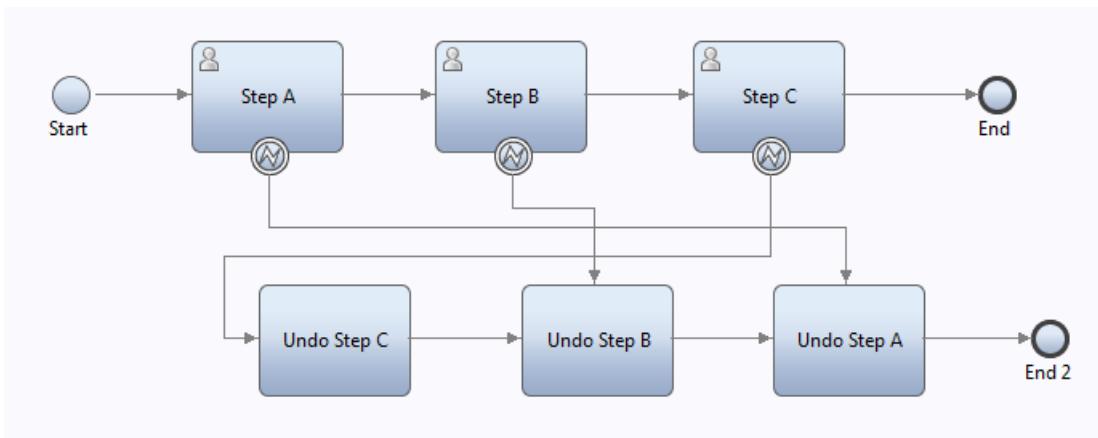
One of the more interesting abilities is the idea of using BPEL compensations to undo the effect of a BPMN described process. To illustrate this, I will keep the story generic to explain the principles which can be applied to an arbitrary process solution. Let us start by examining a simple BPMN process as follows:



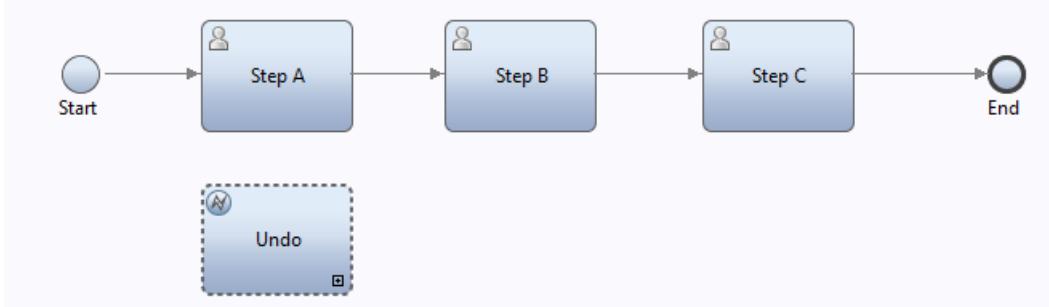
In this story, we see that the process is composed of three steps each executed one after another. We will assume that these steps also update external data systems (systems of record). Since the process updates external systems, if we were to terminate or cancel the process, we may very well be left with an inconsistent data environment. For example, if a step scheduled someone's time for some future work and the process was canceled, that person may still have the booked time on their calendar even though there is now no need for them to do any work. Another example would be allocating parts in a warehouse for assembly which is now no longer going to happen. Those parts would be considered "reserved" and would incorrectly not be able to be used for other projects.

What we need to do is "undo" the steps that were previously executed to return the environment to a

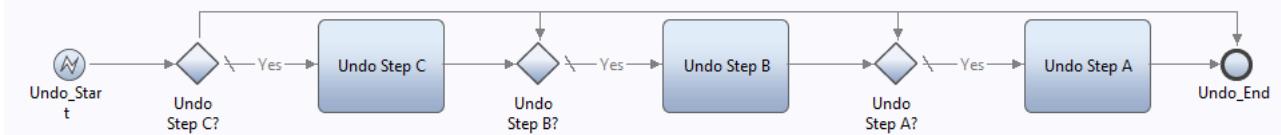
logically consistent and accurate state. One way to achieve this is to explicitly model it within our process. An example might be:



Hopefully you can see how this would work but it also starts to get awkward very quickly. Another possibility for modeling would be to use the "Event Sub-process" notion:



Here the "Undo" step is executed in the event of an exception. The implementation of that might look like:



This looks a little better but now we have to explicitly track which steps in the process were actually performed so that we can know which ones to undo. In addition, the story starts to break down if we have branches or loops in the original process.

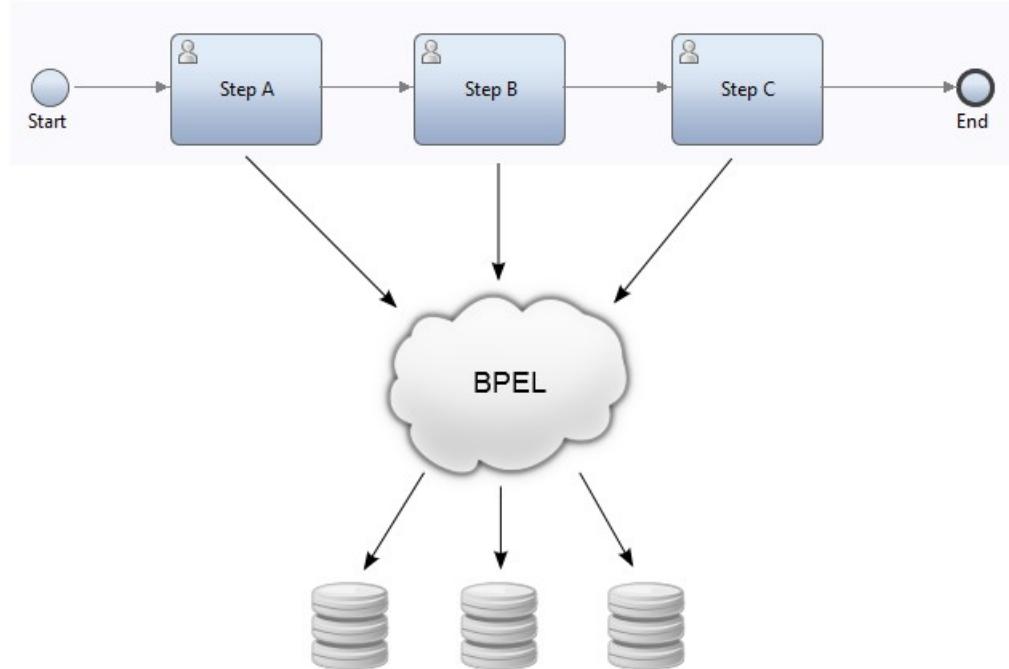
Ideally what we would like is the notion that the steps performed by the process which update external systems are somehow "tracked" by the BPM run-time. Since BPM would then "know" which steps were previously performed, BPM would then be able to automatically execute the undo steps to rollback or "compensate" what the process had done so far before it was canceled.

Fortunately, we can achieve exactly this capability by leveraging the compensation architecture of BPEL. When a BPEL process executes a step, BPEL remembers that it has executed that step. In addition, we can instruct BPEL on what to do "if" that step was ever asked to be undone. Since BPEL now knows what steps were actually performed and what to do if the step is to be compensated, BPEL can now automatically go through its history of steps performed and execute the compensation steps to undo those.

This is a great notion, but we want to avoid writing our business processes in BPEL but instead use the power of BPMN and its business level comprehensibility to model. How then can we merge the story to achieve the best of both worlds?

With IBM BPM Advanced, we have the ability to define steps as being executed by an "Advanced Integration Service" or AIS. When an AIS is reached in IBM BPM, this is an instruction that the work is to be performed by an SCA module and an entry into the SCA module is achieved. Within the SCA module, we can create a BPEL process and wire the AIS call as an entry into that BPEL process. Putting this in simpler terms, we can have an AIS call within BPMN be a call into a BPEL process.

What we can then picture in our minds is something along the following lines:



What this picture represents is the notion that the BPMN process does not make the changes to the external data systems. Instead, BPMN uses AIS to ask BPEL to make the changes. Since BPEL is making the changes, BPEL knows which steps were performed and how to undo those in the event of a failure.

So far so good, so let us start taking this notion to the next set of steps with the goal of actually achieving this design. The first thing to realize is that when the BPMN process runs we can logically think of it as being a process that runs for some period of time until it has a final outcome. Similarly, we need a corresponding BPEL process to also exist for the same duration. The BPMN process and the BPEL process are distinct from each other. There is no known technique that will cause the BPEL process to complete simply because the BPMN process completes. This means that we have to introduce a signaling mechanism that is executed at the very end of the the BPMN process to say that the process has completed and that the corresponding BPEL process can also complete. It is important that we do tell BPEL that it is complete so that it no longer has to maintain the knowledge of how to undo previous steps. If we failed to do this, BPEL would have to remember how to undo previous steps that are no longer ever going to be asked to be undone. We would have a resource leak that would hurt us.

The second thing to consider is that if the BPMN process is asked to undo, we need to tell the BPEL process that it too has to undo its previous work. This introduces a second signal that should be sent from the BPMN process that asks BPEL to rollback. Putting these together we are saying that the BPMN process must either tell BPEL that we are complete or that we are to compensate. One of these must be performed before the BPMN process can be forgotten.

We will call these two types of signals `BPEL_Commit` and `BPEL_Compensate`.

Given that there may be many instances of the BPMN process in existence, it is important that we direct requests to the corresponding correct instance of the BPEL process. This is achieved through the concept of "correlation" which is a first class entity in the BPEL world. In order to achieve correlation, it is suggested that we use the Process Instance ID of the BPMN process as the correlation key.

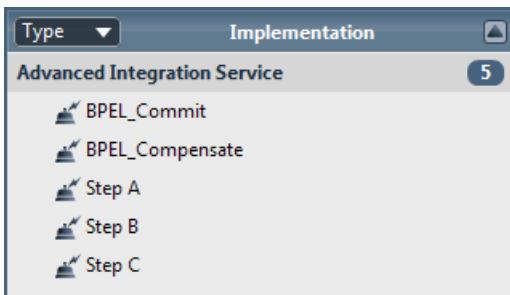
The signature of the two signals then becomes:

- `BPEL_Commit(String correlationId)` – When called, causes the BPEL process to complete and release any resources needed for subsequent compensation.
- `BPEL_Compensate(String correlationId)` – When called causes the BPEL process to execute compensation for all previously executed steps.

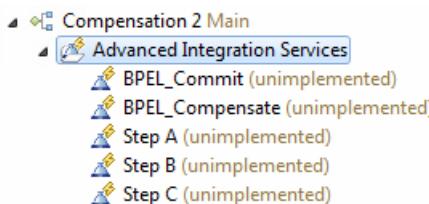
Looking back at the implementation of the data updates that the BPMN process wishes to perform, we saw that each of these will be modeled in BPMN as AIS services. We now also see that these AIS calls will also need the correlation ID so that all work is performed within the same BPEL process instance.

Hopefully we now see all the pieces that we need to provide in the BPMN side of the house to achieve our model and now we turn our attention to the details of the BPEL story.

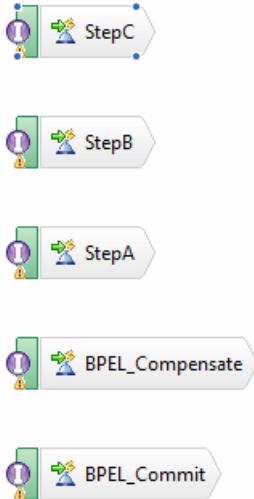
We see that in Process Designer, we have the following AIS definitions.



When we open the project in IID, we see the same items:



Next we implement these choosing "Empty implementation". At the end of this, we now have entry points for each of the AIS's:



Before we go any further, let us look at a generated interface for one of these AISs:

**Interface**

Configuration

|               |                              |                                                                                              |
|---------------|------------------------------|----------------------------------------------------------------------------------------------|
| Name          | BPEL_Commit                  | <a href="#">Refactor name</a>                                                                |
| Namespace     | http://COMP2/AIS/BPEL_Commit | <a href="#">Refactor namespace</a>                                                           |
| Binding Style | document literal wrapped     | <a href="#">Change binding style to document literal non-wrapped</a> <a href="#">More...</a> |

**Operations**

Operations and their parameters

| Name    | Type          |        |
|---------|---------------|--------|
| invoke  |               |        |
| Inputs  | correlationId | string |
| Outputs |               |        |

Notice that the operation name generated is "invoke". This is the same operation name for *each* of the services. We will run into problems if all our operations have the same name as the BPEL interface built for us won't be able to distinguish one operation from another. What we now should do is rename the operation names in each of the generated interfaces. This is an easy task and a good name for the operation is the same name as the service as there will only ever be one operation per service.

**Interface**

Configuration

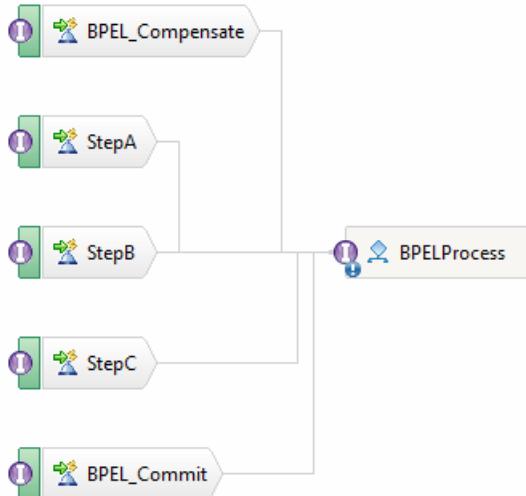
|               |                              |                                                                                              |
|---------------|------------------------------|----------------------------------------------------------------------------------------------|
| Name          | BPEL_Commit                  | <a href="#">Refactor name</a>                                                                |
| Namespace     | http://COMP2/AIS/BPEL_Commit | <a href="#">Refactor namespace</a>                                                           |
| Binding Style | document literal wrapped     | <a href="#">Change binding style to document literal non-wrapped</a> <a href="#">More...</a> |

**Operations**

Operations and their parameters

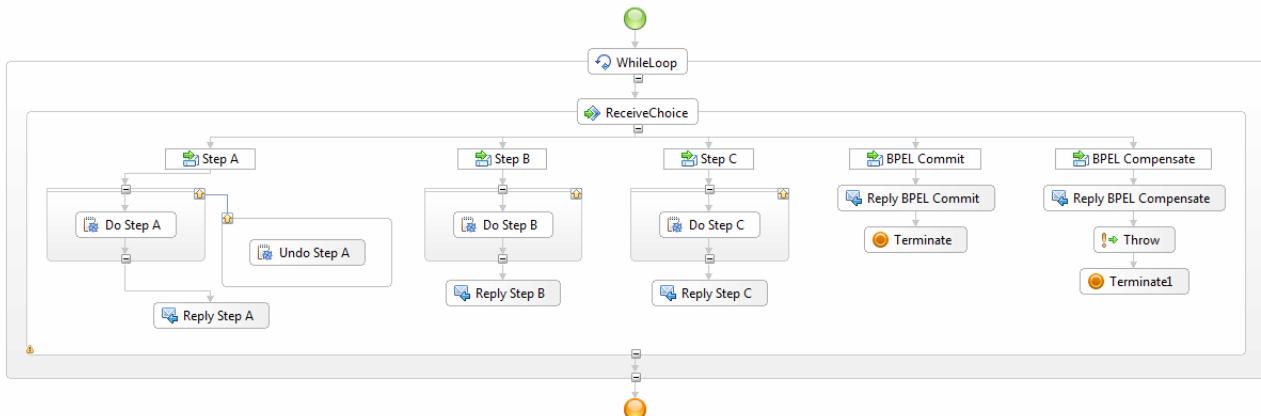
| Name        | Type          |        |
|-------------|---------------|--------|
| BPEL_Commit |               |        |
| Inputs      | correlationId | string |
| Outputs     |               |        |

With this done, we can create a new BPEL process and wire each of the SCA entries into the process. The result will be:



The way to interpret this is that each AIS call will now result in an entry into the BPEL Process.

Now comes the fun part. Now we have to implement the BPEL process to achieve all of these steps. One possible solution looks as follows and is extensible to a wide variety of situations:



To fully comprehend the elegance of this solution, one really has to have an understanding of BPEL. What follows is a summary description.

The core of the solution is a "while loop" which simply loops forever. Within the while loop is the BPEL "Receive Choice" activity. This suspends the process until such time as one of the interfaces to the process is called. The Receive Choice will be waiting for any and all of the different types of interface in our story. Through the magic of BPEL, there won't actually be an instance of this process in existence before the first call from BPMN. The Receive Choice is flagged to create a new process if none already exist. This causes the BPEL run-time to listen as though there was a process but no actual process is present. When the first call arrives for which there is no corresponding correlation ID, a new instance will be created. For all subsequent calls with the same correlation ID, the requests will be directed to the already existing instance for processing.

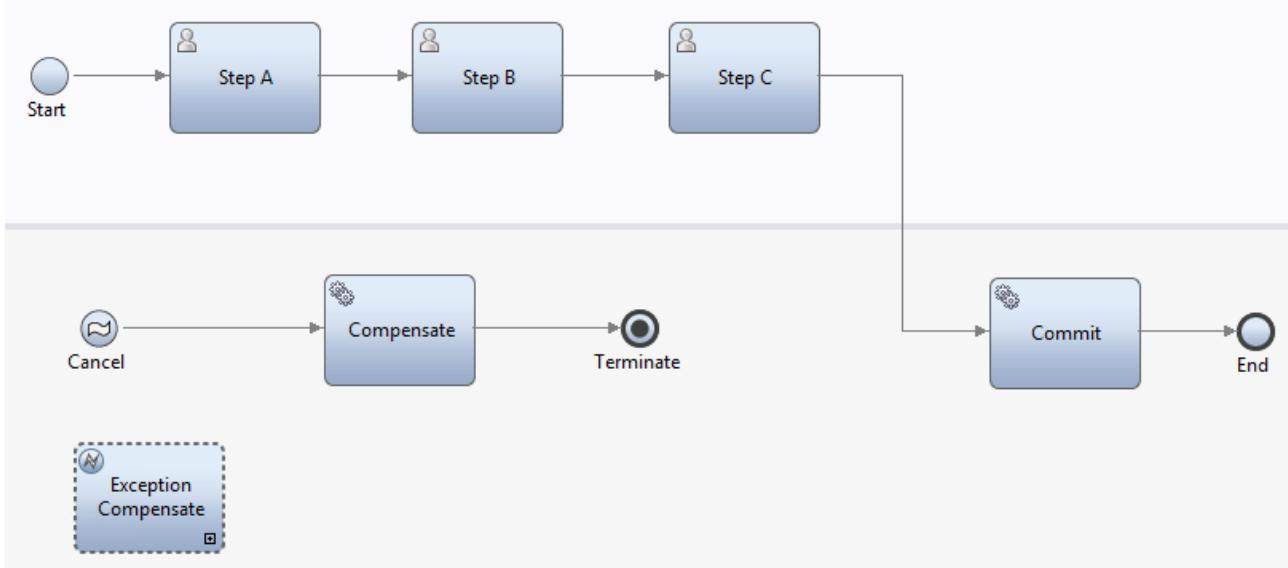
The "function" steps (Step A, Step B and Step C in this example) have a BPEL "Scope" attached to them. Within this Scope we perform the logic necessary to achieve the required request. This could be a database update, a Web Service call, a messaging call or a wide variety of other possibilities. Associated with each BPEL Scope is a compensation handler. This is the logic that will be invoked to "compensate" for the work done in the body of the scope should compensation be needed. It is this logic plus the associated data that the BPEL run-time remembers should it need to subsequently perform compensation. The logic to actually perform the compensation is itself defined in the compensation handler and can again be any of the back-end integrations that may be

needed. Remember also that the data needed to perform the compensation is also part of the compensation scope and hence quite subtle and sophisticated permutations are handled automatically for us.

The logic of "BPEL Commit" is merely to terminate (gracefully and normally) the BPEL process. This causes the run-time to clean up any data it may be hanging on to that allows it to perform future compensation.

Finally, the "BPEL Compensate" throws a BPEL exception to cause the BPEL process to compensate itself.

The final BPMN process that takes advantage of this story looks as follows:



Here we execute the BPEL\_Compensate AIS if either an Ad Hoc request to cancel the process arrives or an unexpected exception is trapped.

See also:

- Advanced Integration Service
- Inter-operating between a BPMN process and an SCA module

# Mediations and ESB

Having looked at SCA and BPEL we are now ready to examine another core concept of the IBPM Advanced product. SCA allows us to decouple service callers and service providers from the tight coupling that might be needed to invoke them. BPEL allows us to provide process flow control to describe the sequence of execution. There is still another level of function that we may need. That function is generically called "mediations".

Mediations deals with aspects of the physical bridge to the outside world. When we think about Web Services calls, JMS calls, MQ calls or other interactions, we can easily imagine *simple* invocations of these technologies. Nine times out of ten, these simple interactions are all that are needed but in some circumstances, we need to exercise detailed and low level control and access. Consider for example a JMS message that contains a property in its header that defines when the message was originally sent. This is not usually surfaced through SCA as SCA abstracts away the details of technical interaction. Consider a Web Service provider that, when called, expects a special token in the SOAP header. Again, an SCA Import doesn't normally accommodate such things as the call to SCA hides the identity and nature of the target system from whomever may be trying to call it.

To solve such problems, lower level access to the physical headers of incoming and outgoing requests has been exposed.

- Book – [WESB 7.0 Development Guide](#)
- DeveloperWorks – [Building mediation flows in WebSphere Integration Developer for deployment on WebSphere ESB or WebSphere Process Server](#) – 2009-08-19

## Service Message Object (SMO)

The Service Message Object (SMO) is a representation of the data to be transformed and the resulting output of that transformation. The SMO contains more than just application data. It contains headers for MQ, SOAP and others as well as contextual information that can be saved and later made available. It is essential to learn the SMO in order to use the mediations capabilities.

When a message arrives at an SCA module from outside of IBPM, it may have protocol information associated with it such as MQ or JMS headers or SOAP envelope information. In addition if a request is made outbound from a module, we may wish to set the protocol information. This is above and beyond the payload of the message itself.

In order to work with such information, a tree data structure called the Service Message Object is created by the system. When building a mediation flow, it is the SMO we work with. The SMO is a large and complex data structure. Fortunately it is well organized and doesn't require full mastery of all of its fields in order to be useful.

| context |             |               |  |  |
|---------|-------------|---------------|--|--|
|         | correlation |               |  |  |
|         | transient   |               |  |  |
|         | failinfo    |               |  |  |
|         |             | failureString |  |  |
|         |             | origin        |  |  |

|         |                  |                       |                     |       |
|---------|------------------|-----------------------|---------------------|-------|
|         |                  | invocationPath        |                     |       |
|         |                  |                       | inTerminal          |       |
|         |                  |                       | name                |       |
|         |                  |                       | outTerminal         |       |
|         | predecessor      |                       |                     |       |
|         | primitiveContext |                       |                     |       |
|         |                  | EndpointLookupContext |                     |       |
|         |                  |                       | endpointReference   |       |
|         |                  |                       | registryAnnotations |       |
|         |                  |                       | classification      |       |
|         |                  |                       | relationship        |       |
|         |                  | FanOutContext         |                     |       |
|         |                  |                       | iteration           |       |
|         |                  |                       | occurrence          |       |
|         |                  | WTXContext            |                     |       |
|         | shared           |                       |                     |       |
|         | dynamicProperty  |                       |                     |       |
|         |                  | propertySets          |                     |       |
|         |                  |                       | group               |       |
|         |                  |                       | properties          |       |
|         |                  |                       |                     | name  |
|         |                  |                       |                     | value |
|         |                  |                       |                     | type  |
|         |                  | isPropagated          |                     |       |
|         | userContext      |                       |                     |       |
|         |                  | entries               |                     |       |
|         |                  |                       | name                |       |
|         |                  |                       | value               |       |
|         |                  |                       | type                |       |
| headers |                  |                       |                     |       |
|         | SMOHeader        |                       |                     |       |
|         |                  | MessageUUID           |                     |       |
|         |                  | Version               |                     |       |
|         |                  | MessageType           |                     |       |
|         |                  | Operation             |                     |       |
|         |                  | Action                |                     |       |
|         |                  | Target                |                     |       |
|         |                  | AlternateTarget       |                     |       |
|         |                  | SourceNode            |                     |       |
|         |                  | SourceBindingType     |                     |       |
|         |                  | Interface             |                     |       |

|             |               |                   |  |  |
|-------------|---------------|-------------------|--|--|
|             | JMSHeader     |                   |  |  |
|             |               | JMSDestination    |  |  |
|             |               | JMSDeliveryMode   |  |  |
|             |               | JMSMessageID      |  |  |
|             |               | JMSTimestamp      |  |  |
|             |               | JMSCorrelationID  |  |  |
|             |               | JMSReplyTo        |  |  |
|             |               | JMSRedelivered    |  |  |
|             |               | JMSType           |  |  |
|             |               | JMSExpiration     |  |  |
|             |               | JMSPriority       |  |  |
|             | SOAPHeader    |                   |  |  |
|             |               | namespace         |  |  |
|             |               | name              |  |  |
|             |               | prefix            |  |  |
|             |               | value             |  |  |
|             | SOAPFaultInfo |                   |  |  |
|             |               | faultcode         |  |  |
|             |               | faultstring       |  |  |
|             |               | faultactor        |  |  |
|             |               | extendedFaultInfo |  |  |
|             | properties    |                   |  |  |
|             |               | name              |  |  |
|             |               | value             |  |  |
|             |               | type              |  |  |
|             | MQHeader      |                   |  |  |
|             |               | md                |  |  |
|             |               | control           |  |  |
|             |               | header            |  |  |
|             | HTTPHeader    |                   |  |  |
|             |               | control           |  |  |
|             |               | header            |  |  |
|             | EISHeader     |                   |  |  |
|             | WSAHeader     |                   |  |  |
| body        |               |                   |  |  |
| attachments |               |                   |  |  |
|             | contentID     |                   |  |  |
|             | contentType   |                   |  |  |
|             | data          |                   |  |  |
|             | bodyPath      |                   |  |  |

## **Mediation Primitives**

A mediation flow component is itself composed of a sequence of one or more primitive actions that are visually composed in WID. In some products such actions are called *nodes* while in WPS they are called *primitives*. A mediation flow is characterized by the combination of these primitives and the configuration attributes associated with each of the primitives.

### **Input primitive**

Every mediation flow starts with an Input primitive. This is the start of the transformation.

### **Callout primitive**

The callout primitive is usually the end of the transformation. When a callout is reached, the SMO passed into it is sent out the reference terminal of the SCA Mediation Flow node and sent onwards to what ever is wired to that terminal on the assembly diagram.

The properties of the callout primitive are shown below:

| Callout : operation1 : I1Partner |                                                                                                                                               |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Description                      | Reference name: I1Partner                                                                                                                     |
| Terminal                         | Operation name: operation1                                                                                                                    |
| Details                          | <input checked="" type="checkbox"/> Use dynamic endpoint if set in the message header                                                         |
| Retry                            | Async timeout (seconds): 120                                                                                                                  |
| Promotable Properties            | <input type="checkbox"/> Require mediation flow to wait for service response when the flow component is invoked asynchronously with callback. |
|                                  | Invocation Style: Default                                                                                                                     |

**Async timeout (seconds)** – The amount of time in seconds after a request message is sent before waiting for a response is abandoned. The default value of this is only 5 seconds which is probably too low, especially when performing manual testing. Other values for this include “0” for immediate return and “-1” for no timeout (unlimited duration).

**Invocation Style** – One of Synchronous, Asynchronous or Default. Default will eventually resolved to either Synchronous or Asynchronous based on some convoluted rules.

### **Business Object Map primitive**



BOMapper1

### **Custom Mediation primitive**



CustomMediation1

The Custom mediation primitive provides the ability for the programmer to code mediation logic in the Java programming language. Within the primitive, the programmer has addressability to the SMO data structure and can see/play with all aspects of the incoming data. The logic can also choose which of the defined output terminals an outgoing SMO will be sent.

In the code for the Custom Mediation there are a number of predefined objects:

- smo – ServiceMessageObject – The DataObject that contains the data passed in
- inputTerminal – InputTerminal – The terminal of the node on which the data arrived
- out – OutputTerminal – The terminal of the node that is the output terminal

```

BOFactory boFactory =
(BOFactory) ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory");
DataObject newBody = boFactory.createByMessage("http://Med1Tests/I1",
"operation1ResponseMsg");
DataObject operation1Response = newBody.createDataObject("operation1Response");
DataObject output1 = operation1Response.createDataObject("output1");
output1.setString("x", "xVal");
output1.setString("y", "yVal");
output1.setString("z", "zVal");
smo.setBody(newBody);
out.fire(smo);

```

## Data Handler primitive

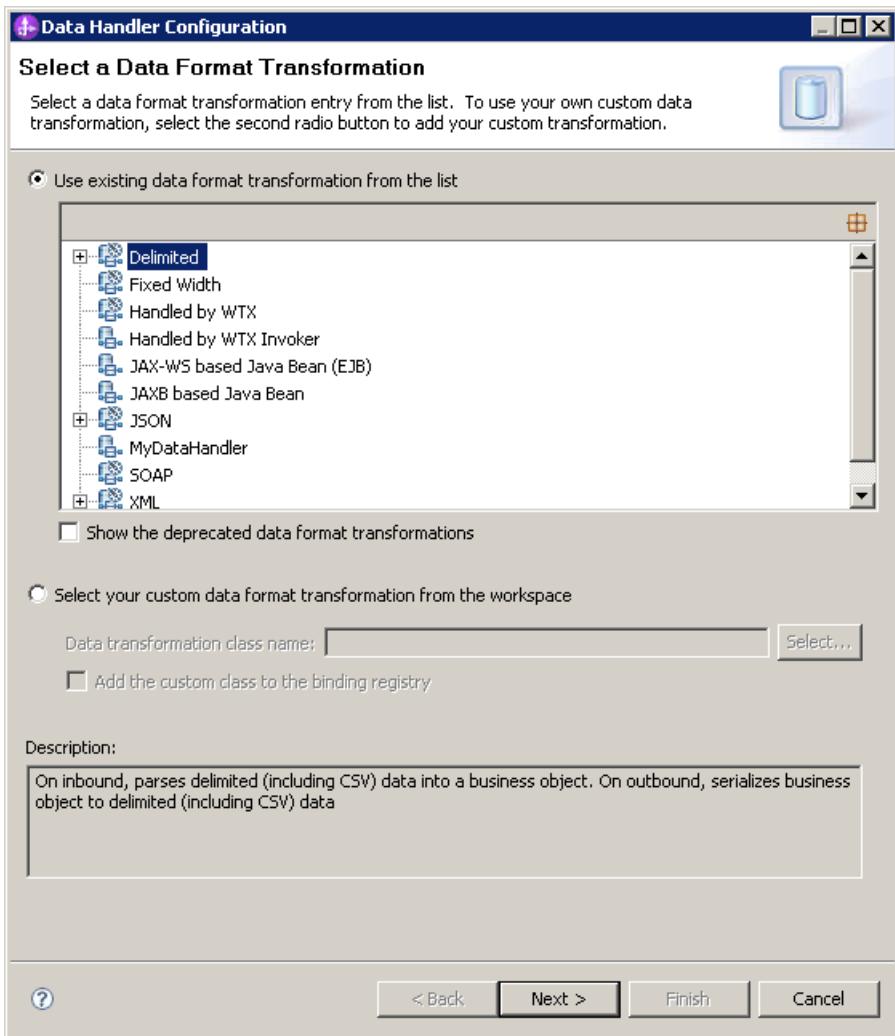


DataHandler1

The purpose of the DataHandler primitive is to transform data from one format to another by executing a DataHandler. This primitive's configuration allows us to name the data handler type to be used. The configuration is made in the Details section in the Properties view within WID. The following screen shot illustrates this screen.

| Weakly typed field | Actual field type | Action                                               |
|--------------------|-------------------|------------------------------------------------------|
|                    |                   | Convert from native data format to a business object |

First, the DataHandler configuration is selected. This is where we select the DataHandler to be invoked at runtime. Clicking on the Browse button opens the DataHandler configuration window:



From here we can select one of the existing DataHandler known to the DataHandler registry. Alternatively, we can select a custom DataHandler from the lower part of the screen. This will examine the workspace looking for a Java Class that implements DataHandler. Optionally, this DataHandler class can be added to the binding registry.

Once the DataHandler has been selected, the Source XPath and Target XPath locations can be supplied. These are locations within the SMO where the input data to the DataHandler will be sourced from and where the output data from the DataHandler will be stored. If a custom data handler is being written, this is data that will be sent to the source parameter of the transform method. The data type of the target field will be the class supplied in targetClass.

## Database Lookup primitive



This mediation primitive enables the message to be augmented with information from a database. The primitive is configured with the JNDI name of a JDBC data source, the name of a table in that database and the name of the column in that table that should be used as a key. An XPath expression is then given that defines where that key is located in the input message. At runtime, the row matching that key is retrieved. If a matching row is not found then the message is propagated to the "keyNotFound" terminal unchanged. The primitive can then be configured with the columns in the

returned row that should be set in to the message at locations given by XPaths. The augmented message is then passed to the out terminal. If processing fails, for example because a connection to the database cannot be achieved, then the message is passed to the fail terminal.

A common pattern is likely to be to use a database lookup to place information in to the transient context of the message and then wire the output to a filter which will then use the information that has been retrieved to make a routing decision.

## Endpoint Lookup primitive



### See Also

- DeveloperWorks – [Use mediation flows to integrate WebSphere Service Registry and Repository with WebSphere Process Server](#) – 2008-11-10

## Event Emitter primitive



## Fail primitive



## Fan In primitive



### See also:

- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 1: Introduction to aggregation](#) – 2008-03-13
- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 2: Service invocation](#) – 2008-03-27
- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 3: Best practices and patterns for aggregation](#) – 2008-04-24
- DeveloperWorks - [Building an aggregation function using WebSphere ESB](#) - 2007-08-15

## Fan Out primitive



### See also:

- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 1: Introduction to aggregation](#) – 2008-03-13

- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 2: Service invocation](#) – 2008-03-27
- DeveloperWorks - [Aggregation functionality in IBM WebSphere Enterprise Service Bus V6.1, Part 3: Best practices and patterns for aggregation](#) – 2008-04-24
- DeveloperWorks - [Building an aggregation function using WebSphere ESB](#) - 2007-08-15

## Flow Order primitive



## Gateway Endpoint Lookup primitive



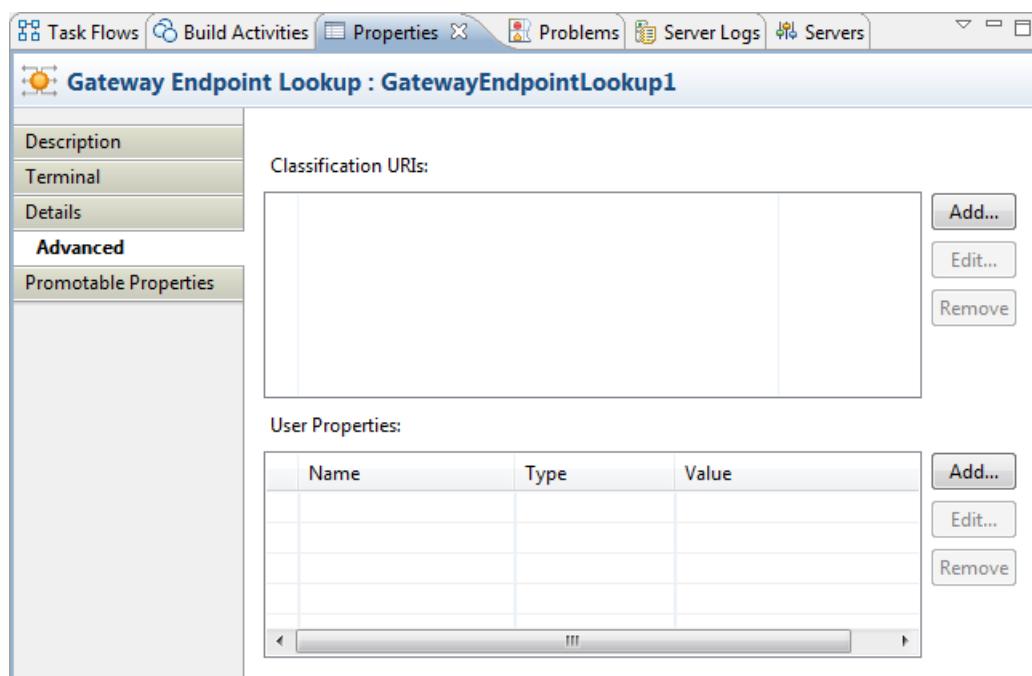
The purpose of this primitive is to perform a lookup of a service registry to dynamically retrieve the endpoint of a target service. This primitive is used in the Gateway and Proxy service patterns.

There are three different techniques for resolving an endpoint via lookup. These methods are:

- URL – The URL to the gateway module has had the name of a virtual service appended to it. For example, if the gateway can be reached at `http://localhost/MyExport` then sending a request to `http://localhost/MyExport/Gold` will attempt to lookup a target service with the virtual service name of Gold.
- XPath – The XPath location of a field in the SMO that is resolved to a value at runtime. This field is then used as the Virtual Service name in the lookup in the built-in proxy gateway configuration set by the administrator through Business Space.
- Action

The information returned from the lookup causes changes to be made to the SMO. Specifically, the following changes are made:

- The endpoint address of the resolved target is placed in /headers/SMOHeader/Target/address
- If there are alternate targets, a list of these is placed in /headers/SMOHeader/AlternativeTarget
- Details of all services found are placed in the context /context/primitiveContext/EndpointLookupContext



## HTTP Header Setter primitive



## JMS Header Setter primitive



## Message Element Setter primitive



## Message Filter primitive



MessageFilter1

## Message Logger primitive



MessageLogger1

## Message Validator primitive



MessageValidator1

## MQ Header Setter primitive



MQHeaderSetter1

In an MQ message, there can be multiple *headers* that prefix the payload of the data. These include:

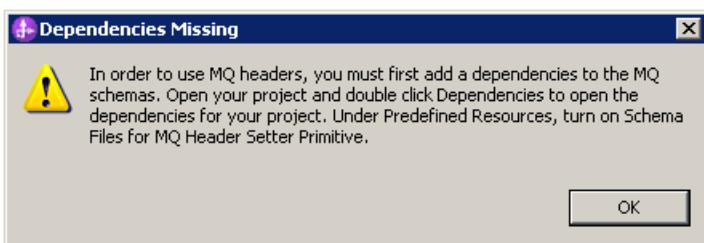
- MQRFH2
- MQCIH
- MQIIH

in addition there is a mandatory header of which there can be only one called MQMD.

The purpose of this primitive is to create, copy or delete whole headers as well as set values in those headers. We can create multiple instances of MQRFH2, MQCIH and MQIIH but because there is only one MQMD, it doesn't mean anything to try and create a second instance or delete the original instance of that structure.

This primitive can be very useful for creating an MQHeader in an SMO where no MQHeader exists (for example when the flow was started by something other than MQ).

When using this primitive for the first time, the following instructional box may appear:



In the dependencies section of the module that contains the primitive, check the box for MQ Header primitive schemas.

#### ▼ Predefined Resources

Select the resources to import into this module. [More...](#)

|                                                                         |
|-------------------------------------------------------------------------|
| <input type="checkbox"/> Apache SOAP schema file                        |
| <input type="checkbox"/> ATOM schema files                              |
| <input type="checkbox"/> Collaboration Scope Resources                  |
| <input type="checkbox"/> Microsoft ADO.NET DataSet schema file          |
| <input checked="" type="checkbox"/> MQ Header Setter Primitive Schemas  |
| <input type="checkbox"/> Native Body schema for Native Body DataHandler |
| <input type="checkbox"/> Service gateway interface                      |
| <input type="checkbox"/> SOAP Encoding schema file                      |

#### Description:

These files are required to properly use the MQ Header Setter Primitive Add/Edit Wizard

## Policy Resolution primitive



PolicyResolution1

When a mediation flow is given control, it executes based on the logic in the mediation in combination with the data supplied with the incoming message. For example, the logic in the mediation may say that a discount applied if the price is over \$1000. The price of an actual order may be found in the body of the incoming message. This works just fine but suffers from a maintenance issue. If the decision logic changes periodically, such as the price at which the discount applies or the value of the discount itself, we would have to open up the Mediation in Process Designer, make the change, save the module and redeploy to the server. Hardly an agile environment. Fortunately, the mediation technology provides a potential solution for this through the use of exposed promoted properties. A property of a mediation can be a variable value used in the solution or a variety of other configurable artifacts. If a property is promoted then it can be dynamically changed through the WAS admin console. This is an improvement but still requires detailed WAS admin console access and means that all mediation flow instances have to use the same single value.

The latest solution is to make use of WSRR and the WS-Policy specification. Using this technique, a set of one or more WS-Policy files can be registered with WSRR. At runtime, the Policy Resolution primitive can be used to contact the WSRR server and dynamically select one or more of those policies based on the values of data passed in as part of the incoming message. Because the policy data is stored as part of WSRR, it is much easier to maintain and modify.

#### See Also

- DeveloperWorks – [Mediation policy for target services: Constructing a dynamic mediation for WebSphere ESB V7 based on which target service has been selected](#) – 2010-01-27
- DeveloperWorks - [Policy lookup with WebSphere Service Registry and Repository using the policy resolution primitive in WebSphere ESB V6.2](#) – 2009-06-24
- DeveloperWorks - [What's new in WebSphere Enterprise Service Bus V6.2, Part 3: Mediation policy](#) - 2009-06-17

## Service Invoke primitive



ServiceInvoke1

## **Set Message Type primitive**



SetMessageType1

## **SLA Check Primitive**



SLACheck1

## **SOAP Header Setter Primitive**



SOAPHeaderSetter1

## **Stop primitive**



Stop1

## **Subflow primitive**



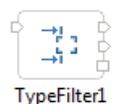
rrr

## **Trace primitive**



Trace3

## **Type Filter primitive**



TypeFilter1

## **UDDI Endpoint Lookup Primitive**



UDDIEndpointLookup1

## Mapping primitive



The purpose of a Mapping transformation is to build a new SMO using an existing SMO as input data. The ID based tooling present a source tree structure representing the input SMO and a destination or target tree structure representing the result. The developer can then perform *wiring* to map the source fields to the target fields. When a wiring is made, this is called a *transform*. ID provides a set of pre-defined transforms to achieve the mapping.

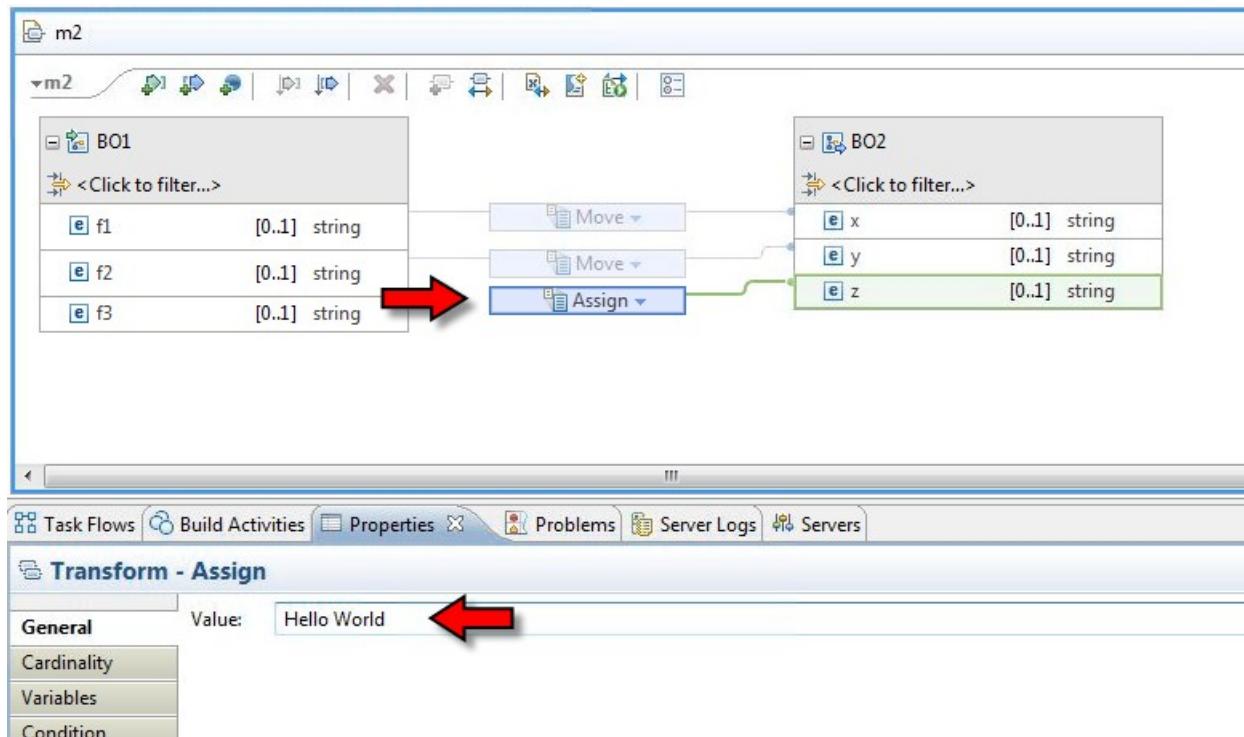
The primitive is called an XSLT transformation as the result of using ID to generate the mappings is saved in a proprietary IBM format called a ".map" file. This .map file is then itself transformed automatically into an XSLT file. It is this XSLT file that is executed by IBPM at run-time using an XSLT 1.0 or XSLT 2.0 processing engine.

When defining the XSLT transformation, you have the ability to select how much of the SMO you wish to map to/from. There are four choices:

- / – Map the whole SMO
- Context – Map only the context section of the SMO
- Headers – Map only the headers section of the SMO
- Body – Map only the body section of the SMO

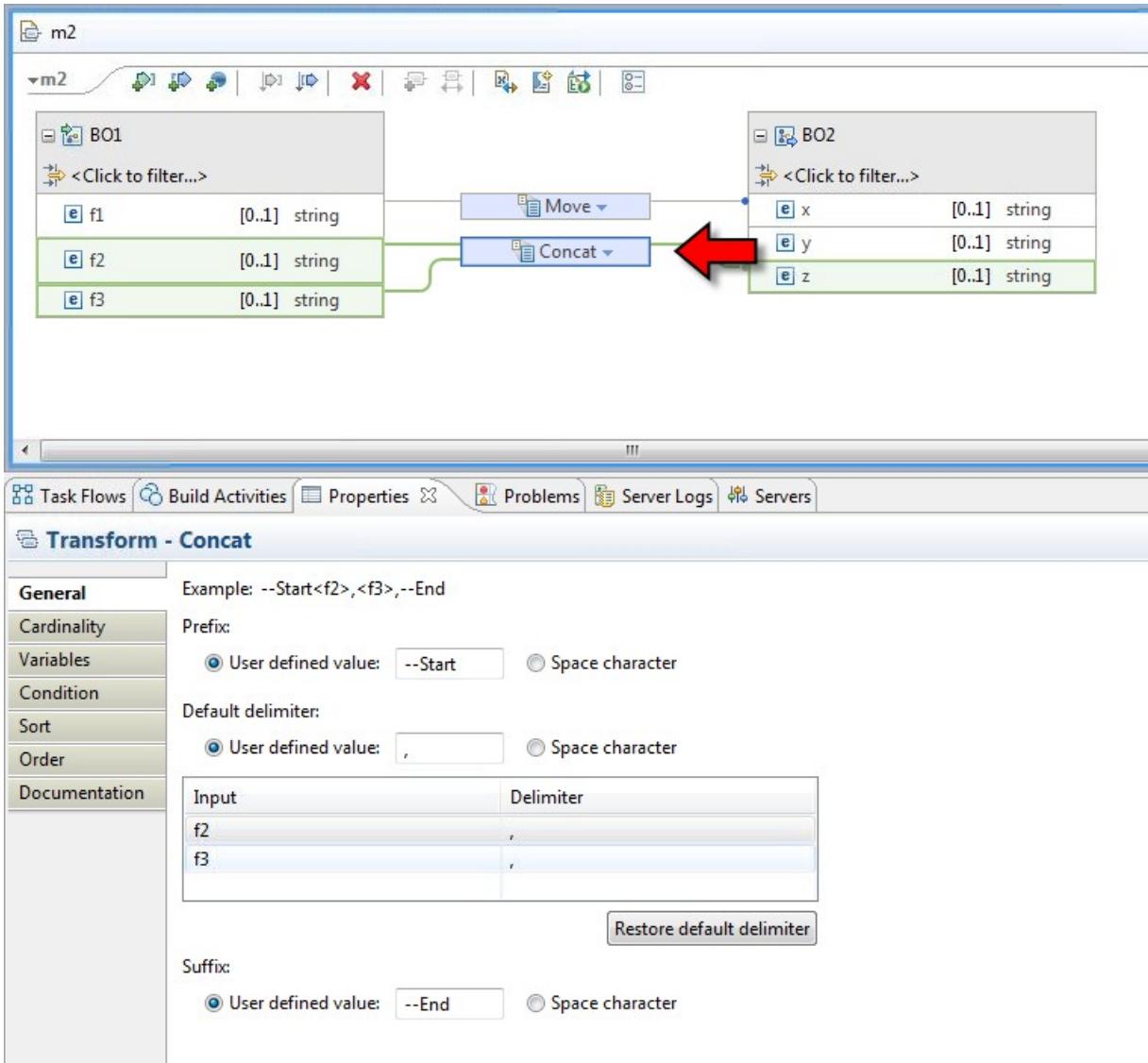
## Core transformations

- append – ???
- assign – Assign a specific hard-coded value to the output element.



- concat – concatenate two or more input elements into an output element. Suffixes, prefixes

and delimiter characters can be introduced.



- convert – Transform a simple data type to another simple data type
- custom – Invoke direct code to perform the transformation. The code can be supplied as Xpath, XSLT or Java. See also
  - developerWorks - [XML mapping in WebSphere Integration Developer V7, Part 2: Working with complex XML structures](#) - 2010-03-24
- for each – Iterate over an array executing a transformation for each element in the array. The output must also be an array.
- move – Move is the simplest transformation. It copies the value of the source field to the target field.
- group – ???
- if, else if, else – ???
- join, merge – Merge is similar to local map but instead of just providing a single input source, multiple input sources can be supplied
- local map – A local map is a stylistic assistance. It allows a mapping from a complex structure to another complex structure to be nested into an easier to manage sub-mapping. Unlike

submaps, a local map provides no reuse.

- lookup – ???
- move – ???
- normalize – White space is removed from the string.
- submap – A submap invokes a named pre-existing map to perform the transformation. This allows a map to be created once and re-used in multiple transforms.
- substring – ???

## XSLT and XPath functions

### Calling Java

Within an XSLT primitive mapping, there is the ability to invoke a Java function. This Java must be implemented in a Java class available to the project. Within the Java class, any method that is to be called from the XSLT must be defined as static. The parameters to the method will be taken from the input parameters supplied in the WID editor. The return value from the method will be the value stored in the target.

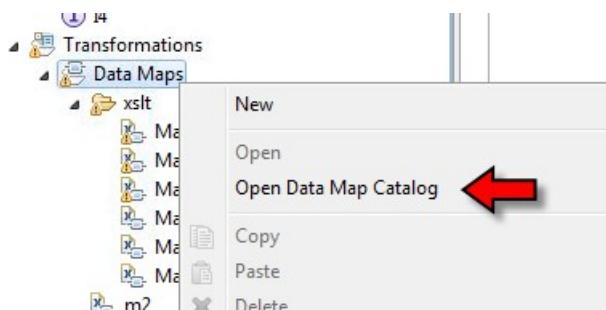
### The catalog of Data Maps

Over time it is not uncommon to create large numbers of Data maps as part of an enterprise solution. ID provides a view call the Data Map Catalog that shows all the available maps plus their inputs, outputs and other related information. From this you can quickly find the map you may wish to use or reuse.

The Data Map Catalog can be opened from the menu bar:



or from the Data Maps folder:



### TechNotes

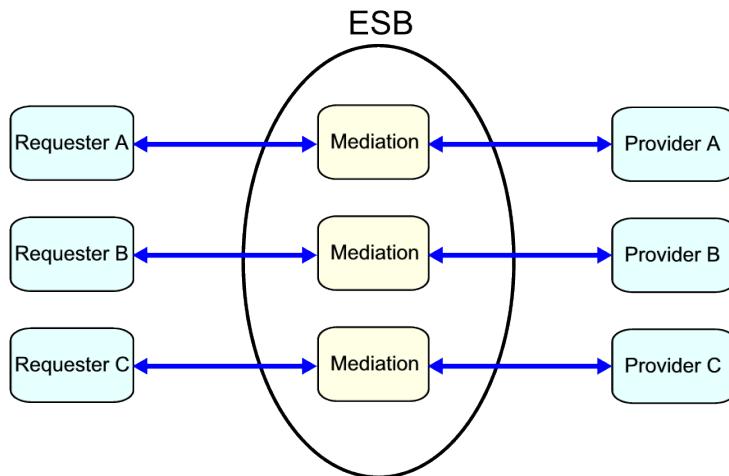
| ID                      | Date     | Description                                                                                                       |
|-------------------------|----------|-------------------------------------------------------------------------------------------------------------------|
| <a href="#">2141767</a> | 01/28/10 | MediationRuntimeException and CWSXM1025E on Move transformation on array elements in XSL Transformation primitive |

## See Also

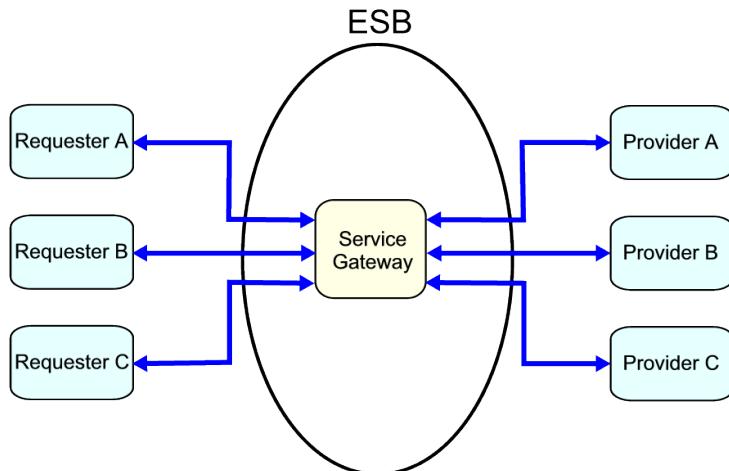
- DeveloperWorks - [Processing an array using an XML map in WebSphere Integration Developer V7](#) - 2011-05-25
- DeveloperWorks - [XML mapping in WebSphere Integration Developer V7, Part 1: Using the Mapping Editor to develop maps](#) - 2010-03-24
- DeveloperWorks - [XML mapping in WebSphere Integration Developer V7, Part 2: Working with complex XML structures](#) - 2010-03-24
- DeveloperWorks - [XML mapping in WebSphere Integration Developer V6.1.2, Part 1: Using the XML Mapping Editor to develop maps](#) - 2008-11-26
- DeveloperWorks - [XML mapping in WebSphere Integration Developer V6.1.2, Part 2: Working with complex XML structures](#) - 2008-12-17
- DeveloperWorks - [Using XSLT primitives for data transformation in WebSphere ESB](#) - 2007-10-31
- DeveloperWorks - [WebSphere Integration Developer and WebSphere ESB advanced topics, Part 2: Data enrichment, transformation, and validation](#) - 2007-10-31
- DeveloperWorks - [Advanced XSL transformation mediation of arrays in WebSphere ESB](#) - 2006-10-18

## *The Service Gateway pattern*

Consider the following illustration. It shows a series of service requesters invoking a series of service providers using ESB technology to mediate between them. As is commonly the case, the ESB may be protocol switching or performing data content/format transformation or some other such task. Obviously this works but can we do better?



Now lets us consider this next diagram:



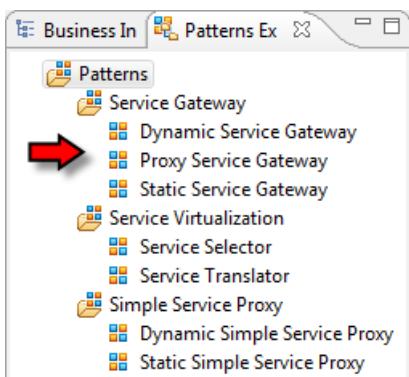
Once again we have a series of service requesters and service providers but now the routing flows through something called a *Service Gateway*. A Service Gateway is nothing new from an SCA Mediation perspective, instead it is a *pattern* of usage as opposed to a radically new technology. The Service Gateway can be thought of as an aggregation of function into a single mediation where previously there would otherwise have been multiple mediations.

Looking at the Service Gateway pattern, it isn't immediately obvious what problem it solves or benefits it adds. However, there are many. Some of them include:

- Common and consistent processing of requests such as logging or auditing.
- Addition or removal of service providers dynamically.
- Additional quality of services such as security.

By using the Service Gateway pattern, the target provider services all become accessible through a single incoming endpoint. The providers that are exposed through the Service Gateway effectively become a *set* of target services that become logically grouped together.

Within Integration Developer, there is a wizard that is available to create these patterns. It can be found at Window > Show View > Patterns Explorer.



The Service Gateway patterns do not need to be created through the wizard; they can just as well be created by hand.

See also

- DeveloperWorks - [What's new in WebSphere Enterprise Service Bus V6.2, Part 2: Service gateway patterns](#) - 2009-06-10

## Proxy Gateways

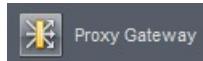
Consider an example where we have three possible back-end service providers. These may relate to customer service levels such as Silver, Gold and Platinum. If we have a calling service, this would appear to need bindings to each of the three back-end services. This is not a great solution. It could easily end up being brittle. Binding the client to three back-end services doesn't give us much flexibility if we need to change a service location. In addition, we would have to replicate three qualities of service for each of the different back-ends. For example, if we wanted to add logging to the logical service, we would have to perform that task three times.

The solution provided by the mediations functions is the introduction of something called a *virtual service*. Think of a virtual service as being a logical name of a target service which, at runtime, is resolved via a lookup to the actual service that is to be called. By utilizing this technique, we have again separated the caller from the target service. The caller now supplies the name of the virtual service to be invoked and the mediation performs the determination of how to reach the physical service that the virtual service represents. This pattern is called a *Proxy Gateway*.

When a caller invokes a mediation that uses this Proxy Gateway pattern, the mediation looks at the request and then performs a calculation to determine the final destination and then routes the request onwards for processing. The mediation thus becomes a proxy (or stand-in) for the real target service.

The core technology behind the Proxy Gateway pattern is the mediation primitive called the Gateway Endpoint Lookup primitive.

Proxy Gateways are managed from within Business Space using the widget called Proxy Gateway. This can be found in the Solution Operations folder.



Let us look at an example. Imagine we have a Gateway Endpoint Lookup that has the following definition:

| Proxy Group Name |
|------------------|
| proxyGroup1      |

Now we need to introduce the concept of a Proxy Group. A Proxy Group is a logical set of service providers that are collected together in a named group/set. By grouping them together we have a way to refer to them as a unit. Notice that in our example we have defined a proxy group called "proxyGroup1". Once the module containing the mediation that contains the gateway endpoint lookup is deployed, a new entry will be found in the Business Space Proxy Gateway widget:

| Proxy Group | Proxy Gateway   |
|-------------|-----------------|
| proxyGroup1 | ServiceGateway1 |

Note the name of the proxy group and the name of the proxy gateway. The proxy gateway name is the name of the module. On the right hand side of the entry there is a pencil icon which indicates an editing session. This is where we associate endpoints with the group. Here is an example of some endpoints already added.

| Proxy Gateway                |                      |                                                     |
|------------------------------|----------------------|-----------------------------------------------------|
| proxyGroup1 Proxied Services |                      |                                                     |
| Enabled                      | Virtual Service      | Endpoint                                            |
|                              | Silver_I1HttpService | http://localhost:9080/ServiceGateway1Web/sca/Silver |
|                              | Gold_I1HttpService   | http://localhost:9080/ServiceGateway1Web/sca/Gold   |

WSDL Location:  Add Service... Cancel

Note the primary attributes of an entry. There is the virtual service name, the endpoint and whether or not it is enabled. To add a new virtual service, enter the location of the **wsdl** in the WSDL location text box and click the Add Service... button.

| Proxy Gateway                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                    |  |               |          |                                             |                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|---------------|----------|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Port Type: {http://ServiceGateway1/I1}I1                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                    |  |               |          |                                             |                                                                                                                                                     |
| Virtual Service Name:                                                                                                                                                                                              | <input type="text" value="Platinum_I1HttpService"/>                                                                                                                                                                                                                                                                |  |               |          |                                             |                                                                                                                                                     |
| Virtual Service URLs:                                                                                                                                                                                              | <table border="1"> <thead> <tr> <th>Proxy Gateway</th> <th>Endpoint</th> </tr> </thead> <tbody> <tr> <td>ServiceGateway1</td> <td>http://localhost:9080/ServiceGateway1Web/sca/Silver/Platinum_I1HttpService http://localhost:9080/ServiceGateway1Web/sca/Gold/Platinum_I1HttpService</td> </tr> </tbody> </table> |  | Proxy Gateway | Endpoint | ServiceGateway1                             | http://localhost:9080/ServiceGateway1Web/sca/Silver/Platinum_I1HttpService http://localhost:9080/ServiceGateway1Web/sca/Gold/Platinum_I1HttpService |
| Proxy Gateway                                                                                                                                                                                                      | Endpoint                                                                                                                                                                                                                                                                                                           |  |               |          |                                             |                                                                                                                                                     |
| ServiceGateway1                                                                                                                                                                                                    | http://localhost:9080/ServiceGateway1Web/sca/Silver/Platinum_I1HttpService http://localhost:9080/ServiceGateway1Web/sca/Gold/Platinum_I1HttpService                                                                                                                                                                |  |               |          |                                             |                                                                                                                                                     |
| Endpoint URLs:                                                                                                                                                                                                     | <input type="text" value="http://localhost:9080/ServiceGateway1Web/sca/Platinum"/> <div style="display: flex; justify-content: space-between;"> <span><input type="button" value="Add Endpoint"/></span> <span><input type="button" value="^"/></span> <span><input type="button" value="▼"/></span> </div>        |  |               |          |                                             |                                                                                                                                                     |
| <input checked="" type="checkbox"/> Enable Virtual Service                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                    |  |               |          |                                             |                                                                                                                                                     |
| <b>Advanced Service Properties</b> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="button" value="Add Property"/></td> <td></td> </tr> </tbody> </table> |                                                                                                                                                                                                                                                                                                                    |  | Name          | Value    | <input type="button" value="Add Property"/> |                                                                                                                                                     |
| Name                                                                                                                                                                                                               | Value                                                                                                                                                                                                                                                                                                              |  |               |          |                                             |                                                                                                                                                     |
| <input type="button" value="Add Property"/>                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                    |  |               |          |                                             |                                                                                                                                                     |
| <input type="button" value="Save"/> <input type="button" value="Cancel"/>                                                                                                                                          |                                                                                                                                                                                                                                                                                                                    |  |               |          |                                             |                                                                                                                                                     |

We are presented with the details of the new virtual service to add. Clicking Save will save the definitions. A Virtual Service is the mapping of a named entry to a concrete or real endpoint. So based on what we have seen, a Proxy Gateway is a module that can have one or more Gateway Endpoint Lookup primitives. Each one of these can define one or more Proxy Groups and a Proxy Group is a collection of one or more Virtual Services.

Obviously this information has to be stored somewhere. IBM Advanced provides a built-in storage mechanism for hosting this data. It is the Business Space widgets that provide the administration of these logical functions.

For additional details on using the Proxy Gateway pattern, see the Gateway Endpoint Lookup primitive.

## See Also

- DeveloperWorks - [Developing and deploying a proxy gateway using WebSphere ESB V7, WebSphere Integration Developer V7, and Business Space powered by WebSphere](#) – 2010-07-14

## SOAP Message Attachments

With Web Services, a request from a caller is sent to a provider. The caller builds an XML document that conforms to the industry standard SOAP specification that encodes the request that is desired to be made. This is then usually transmitted over HTTP to the service provider which unpacks the SOAP request, performs the work requested of it and may send back a response.

There are cases however where the caller wishes to send over what we generically call *attachments*. These are commonly much larger items that parameter fields and include such things as images, scanned fax documents, zip files, or office documents. Typically these attachments are natively represented as binary data which is not suitable for encoding within the normal SOAP body as parameters. The product has a limitation of no more than 20 MBytes for the total attachment sizes.

The trick to passing these attachments is to encode them using the Multipart Media (MIME).

Here is an example SOAP message with an attachment:

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary; type=text/xml;
 start=<claim061400a.xml@claiming-it.com>
Content-Description: This is the optional message description.

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim061400a.xml@claiming-it.com>

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
 ..
 <theSignedForm href="cid:claim061400a.tiff@claiming-it.com"/>
 ..
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: image/tiff
Content-Transfer-Encoding: binary
Content-ID: <claim061400a.tiff@claiming-it.com>

...binary TIFF image...
--MIME_boundary--
```

There are a number of styles of SOAP with attachments. The first is called an *unreferenced attachment*. It is called this because in the SOAP envelope there is simply no reference to the contained attachment. The alternative is known as a *referenced attachment*. In this case, an element in the SOAP message logically refers to the attachment data.

See Also:

- DeveloperWorks - [Using attachments in SOA applications with WebSphere Integration Developer V7](#) – 2010-03-17
- DeveloperWorks - [Using SOAP with attachments in WebSphere Enterprise Service Bus 6.2.0.1](#) – 2009-08-05
- WS-I - [Attachments Profile Version 1.0](#) - 2006-04-20
- W3C - [SOAP Message Transmission Optimization Mechanism](#) – 2005-01-25
- W3C – [SOAP Messages with Attachments](#) – 2000-12-11

## Tracing

A suitable WAS trace flag for tracing mediations is:

com.ibm.ws.sibx.\*=all

# Business Calendars

See also:

- DeveloperWorks - [Using WebSphere Process Server business calendars in business processes](#) – 2009-03-25
- DeveloperWorks - [Comment lines: Bhargav Perepa: Business calendars and timetables can be fun](#) - 2009-01-28

## JCA Adapters

Many back-end systems that we wish to integrate with may not be exposed as logical services. This means that they have no generic interface to them. There is no Web Service, JMS, MQ or REST access. In order to interoperate with services like these, we would have to write connector code. Classically such connectors are called *adapters*. IBPM Advanced supplies a set of adapters that can be integrated with the SCA framework. These adapters are based on the Java Connector Architecture (JCA). The following section provides descriptions of these adapters and how they might be utilized.

### ***Flat File Adapter***

The Flat File adapter provides the ability to write to a file or read from a file from an SCA Import or Export. At a high level, the adapter provides the following functions:

- Create a new file where one did not previously exist
- Append to an existing file or overwrite the content of an existing file
- Get the content of a file
- Obtain a list of file names in a given directory
- Delete an existing file
- Determine if a named file exists or not
- Periodically check a folder for the existence of new files

Inbound processing is when the adapter is watching a folder for the arrival of new files. The adapter polls the directory on a configurable frequency. When a file is found, its content is read and turned into a BO. The transformation from physical file data to a BO is performed by a DataHandler.

Outbound processing is when the SCA module wishes to create, update, delete or list files. Again, BO to physical data transformation has to take place and again this is performed through a DataHandler.

### **Sample – Reading a Flat File with delimited data**

Here is a sample walk through of reading a Flat File as input which contains delimited data.

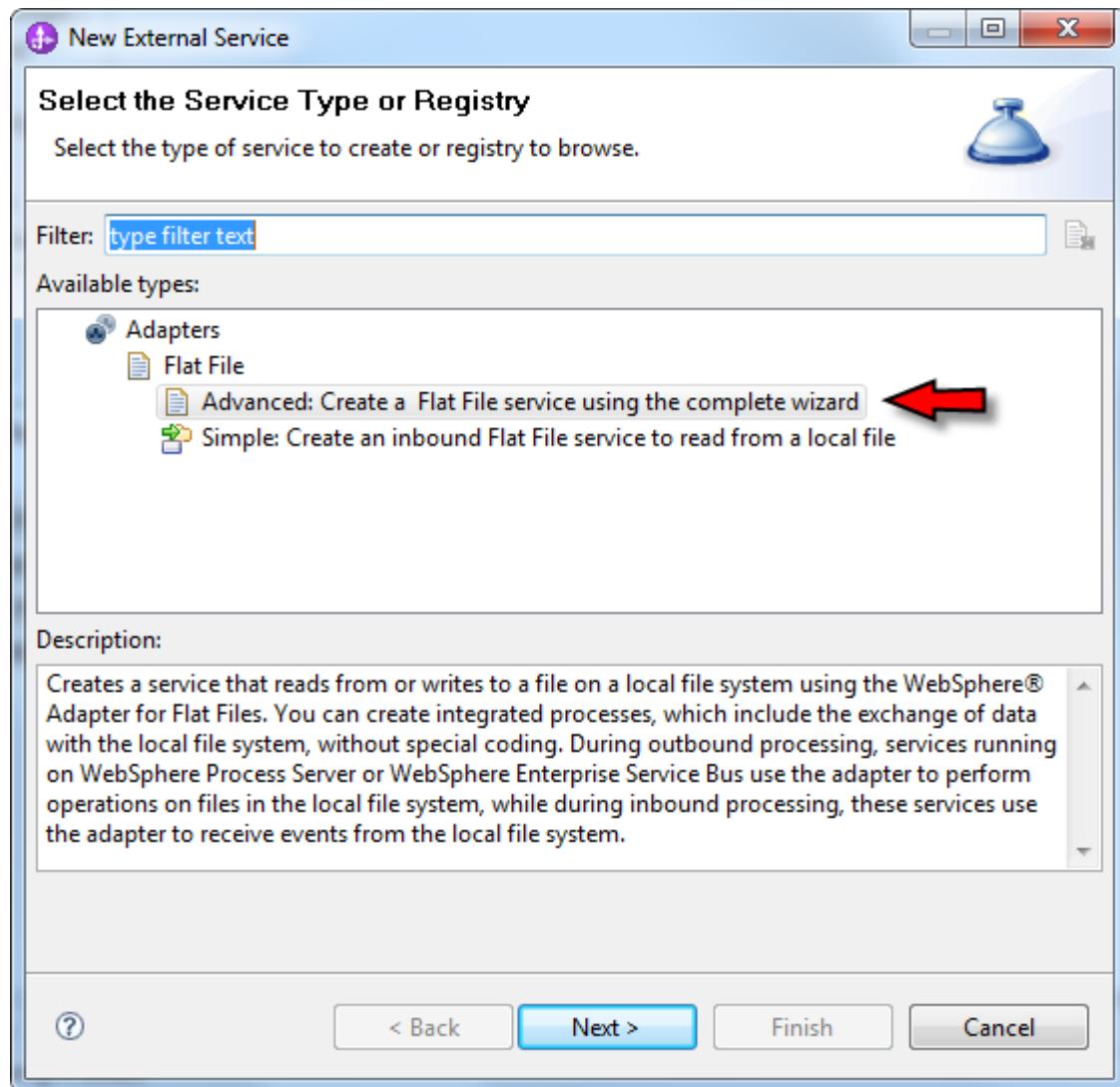
We will start by considering a Business Object definition that looks as follows:

|                      |           |
|----------------------|-----------|
|                      | BO1       |
| <Click to filter...> |           |
|                      | f1 string |
|                      | f2 string |
|                      | f3 string |
|                      |           |

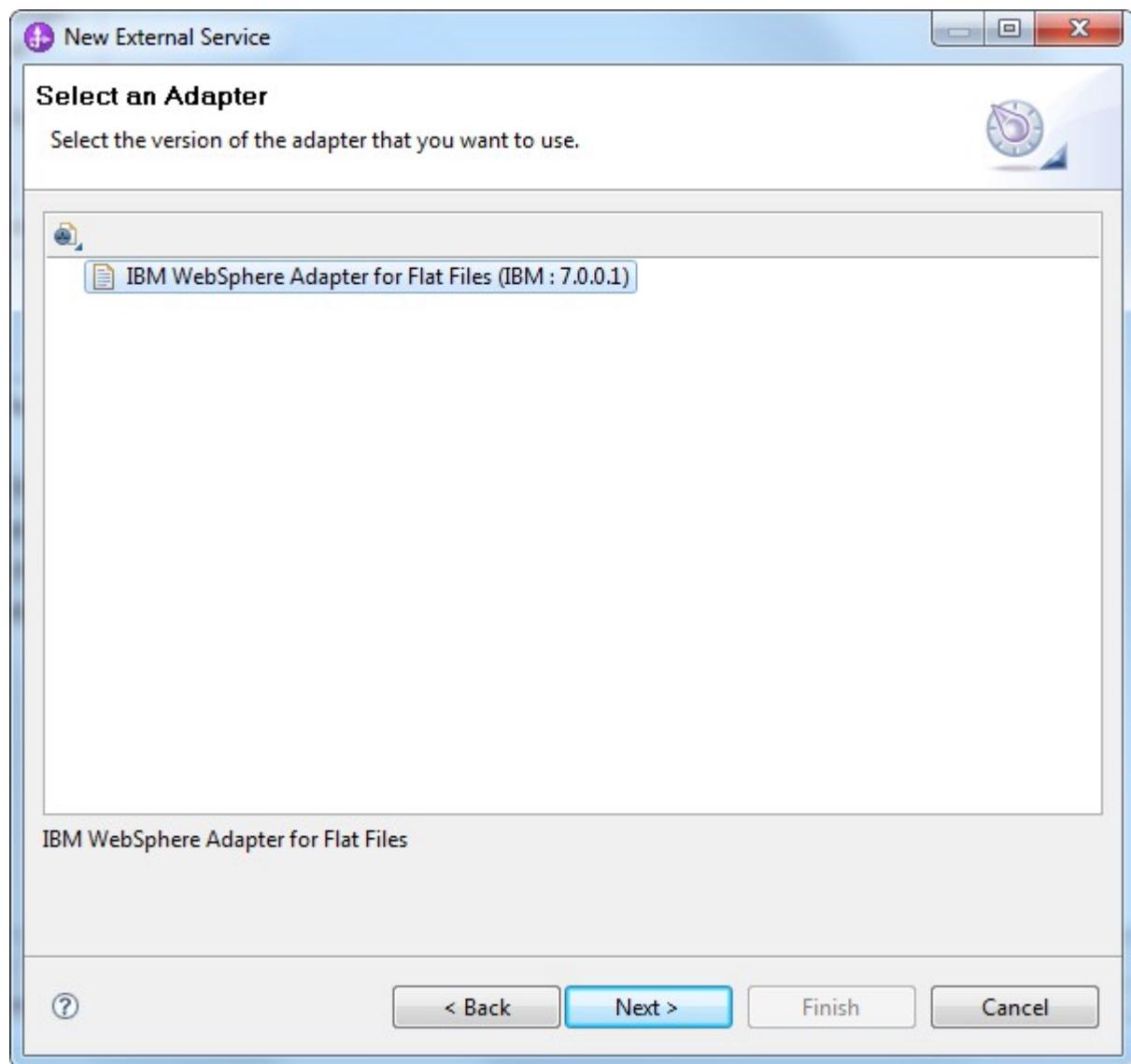
Our goal is to read a record from a file which looks like:

"Val1", "Val2", "Val3"

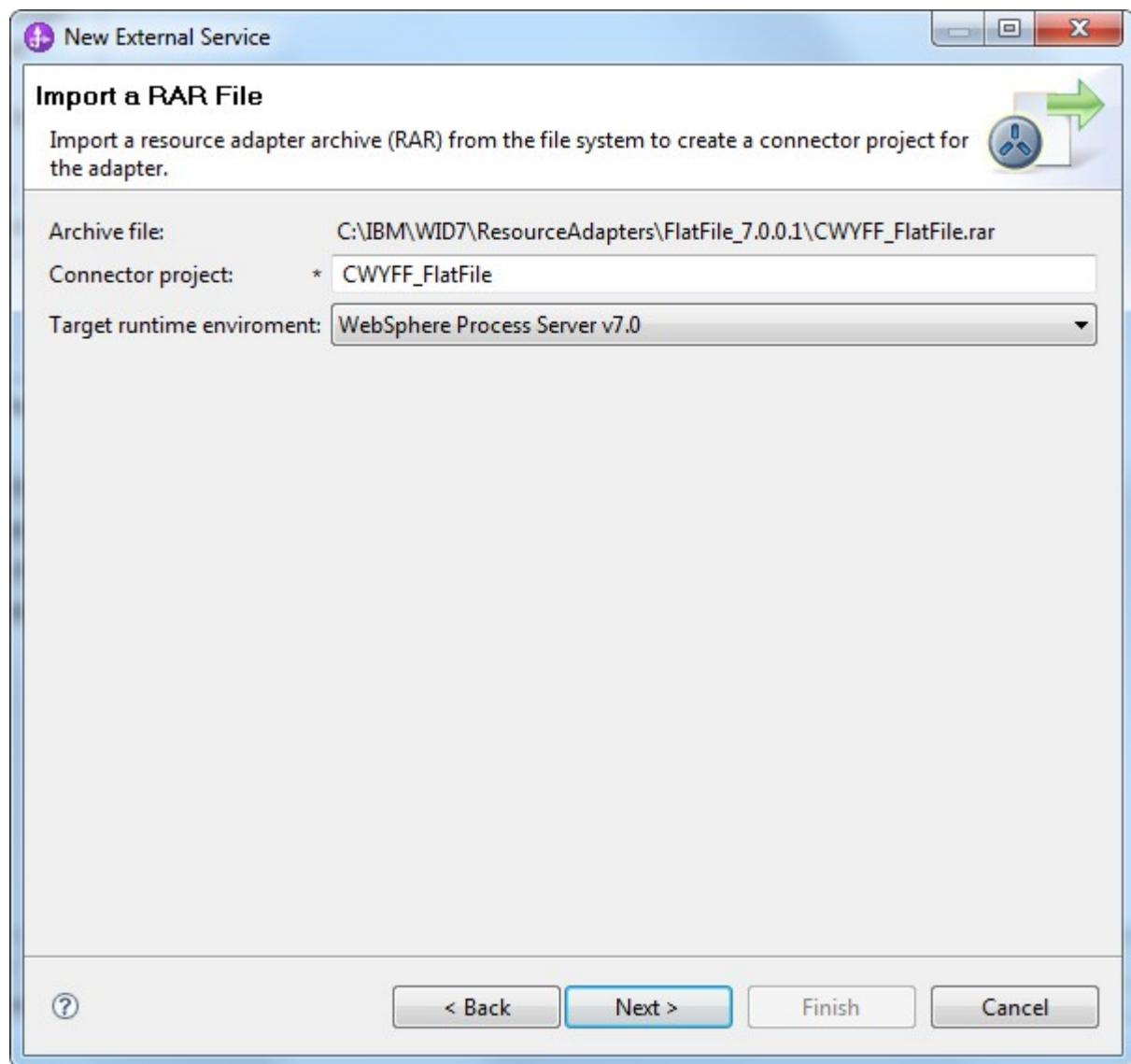
which will create an instance of the BO. We create an Inbound Flat File adapter which launches the wizard:



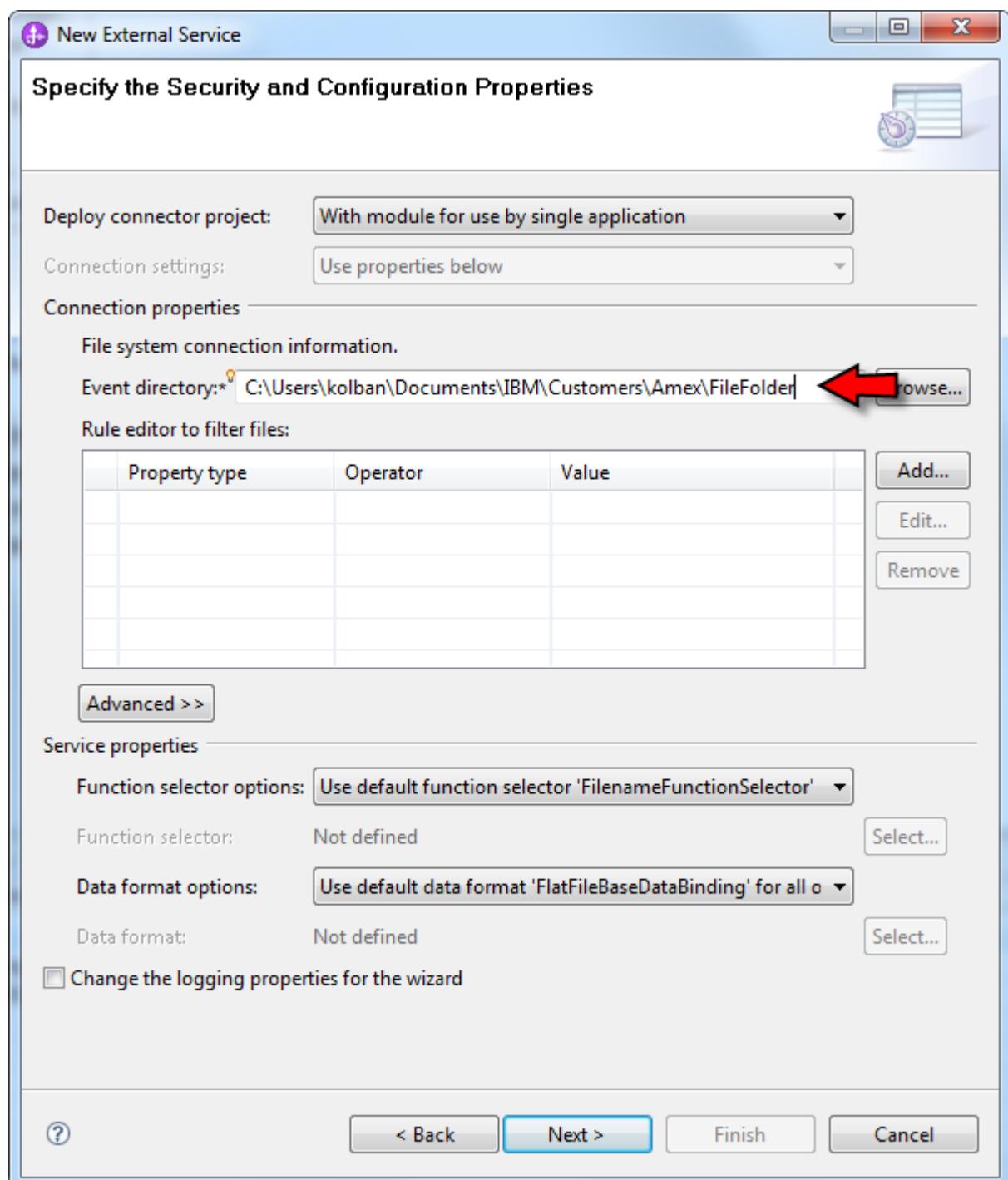
We next select the implementation of the Adapter to be used.

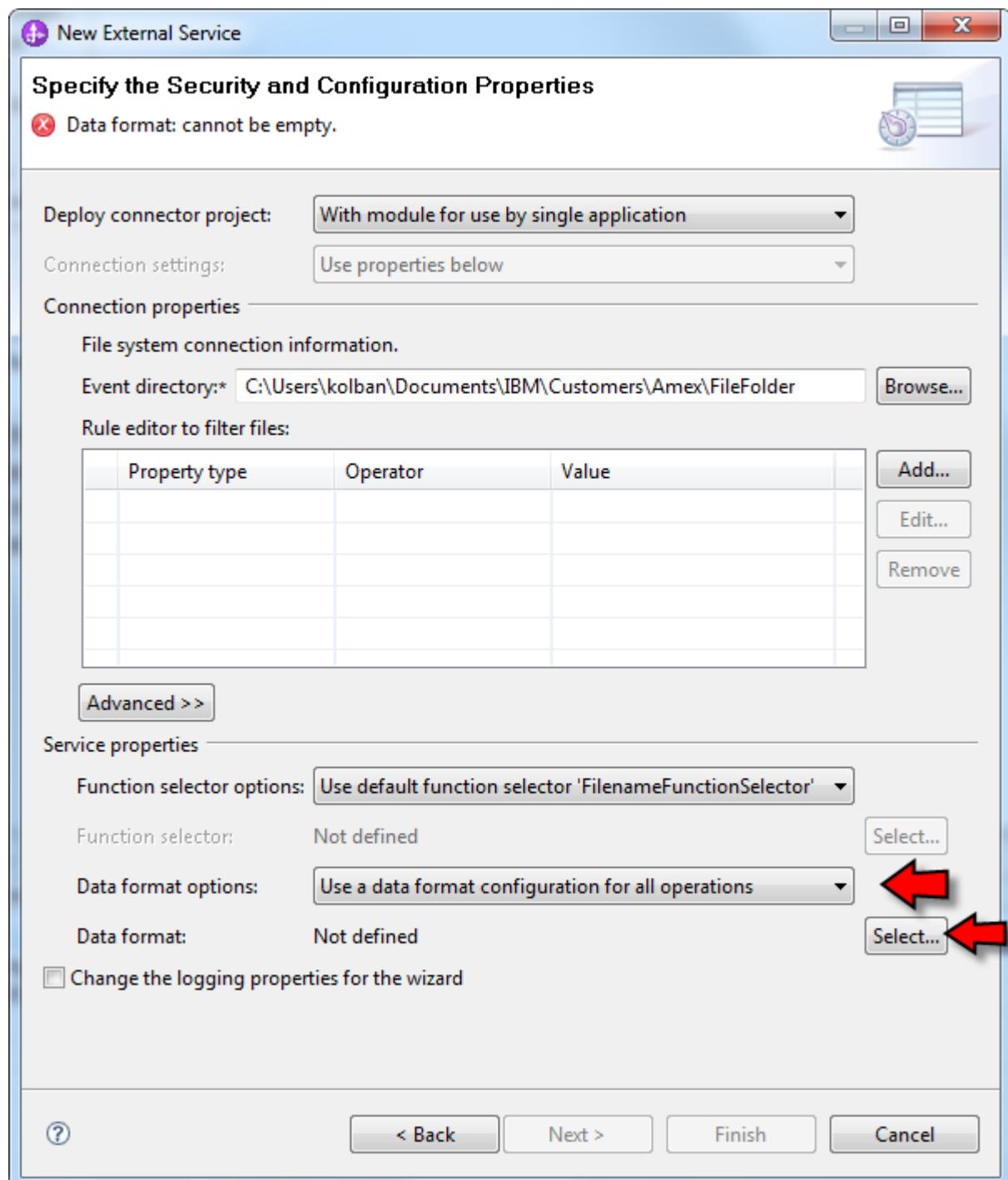


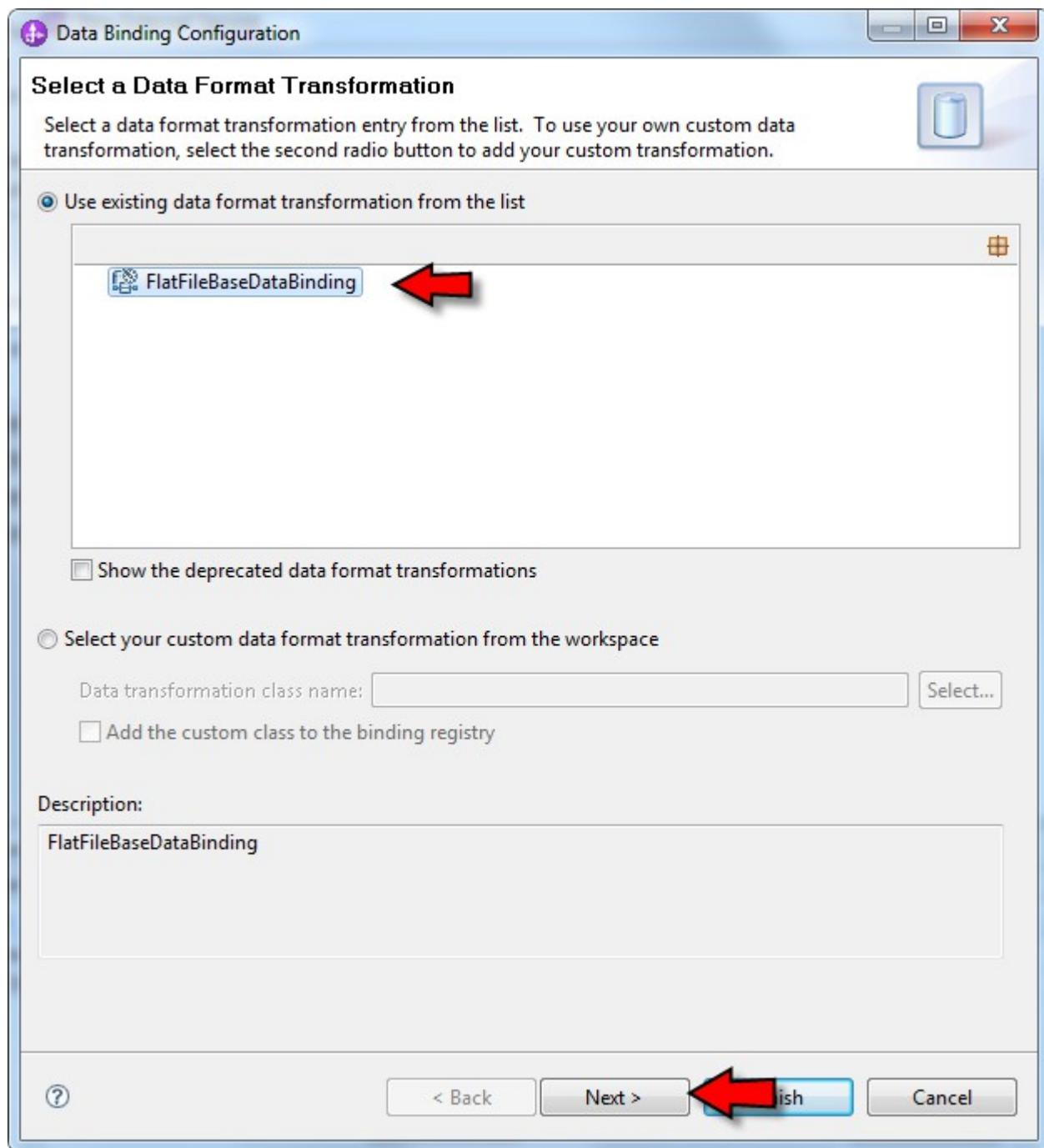
The adapter to be read is loaded into its own Java project. The adapter is loaded from a RAR file. We are asked to name the Workspace project that will hold the RAR.

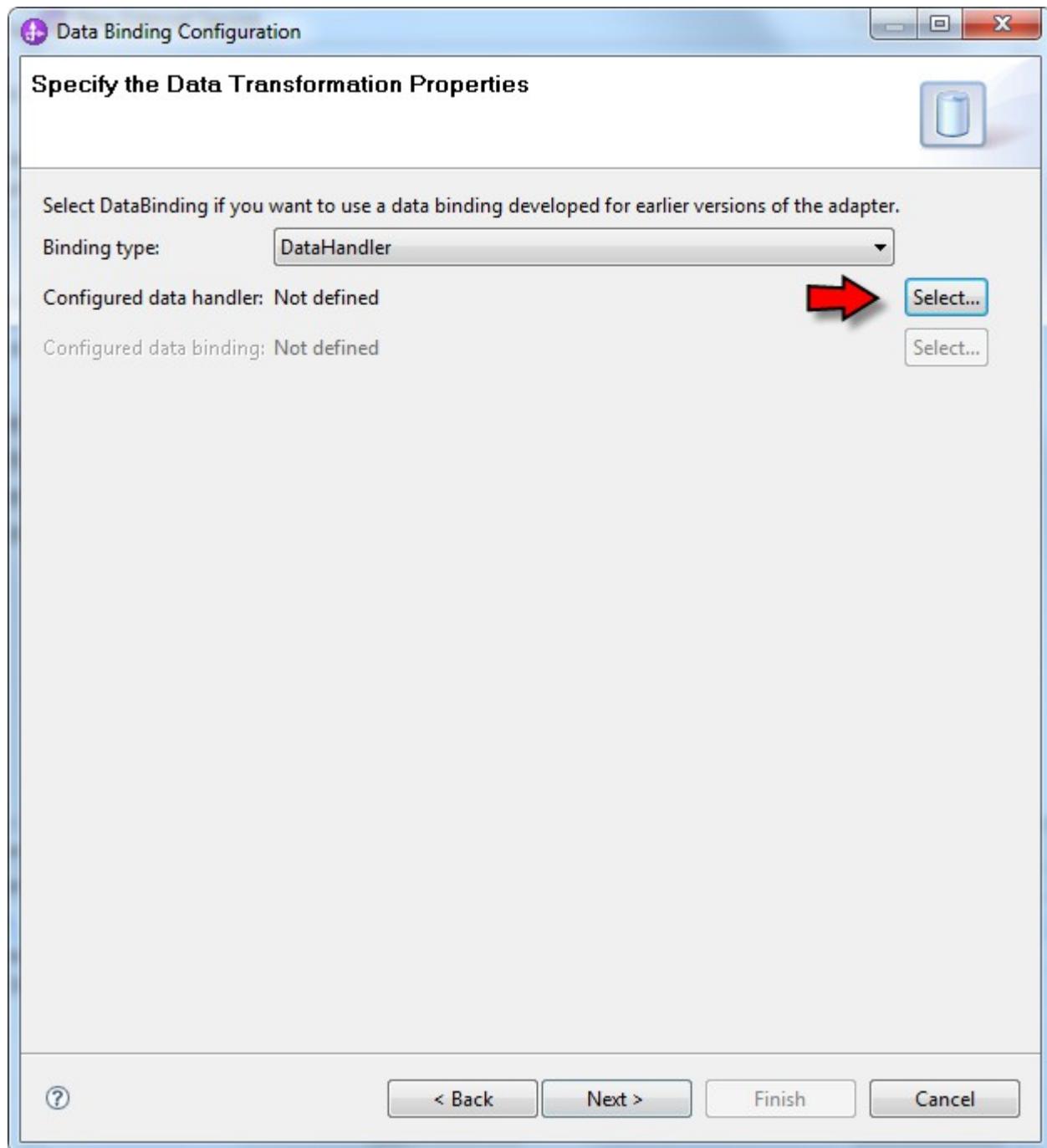


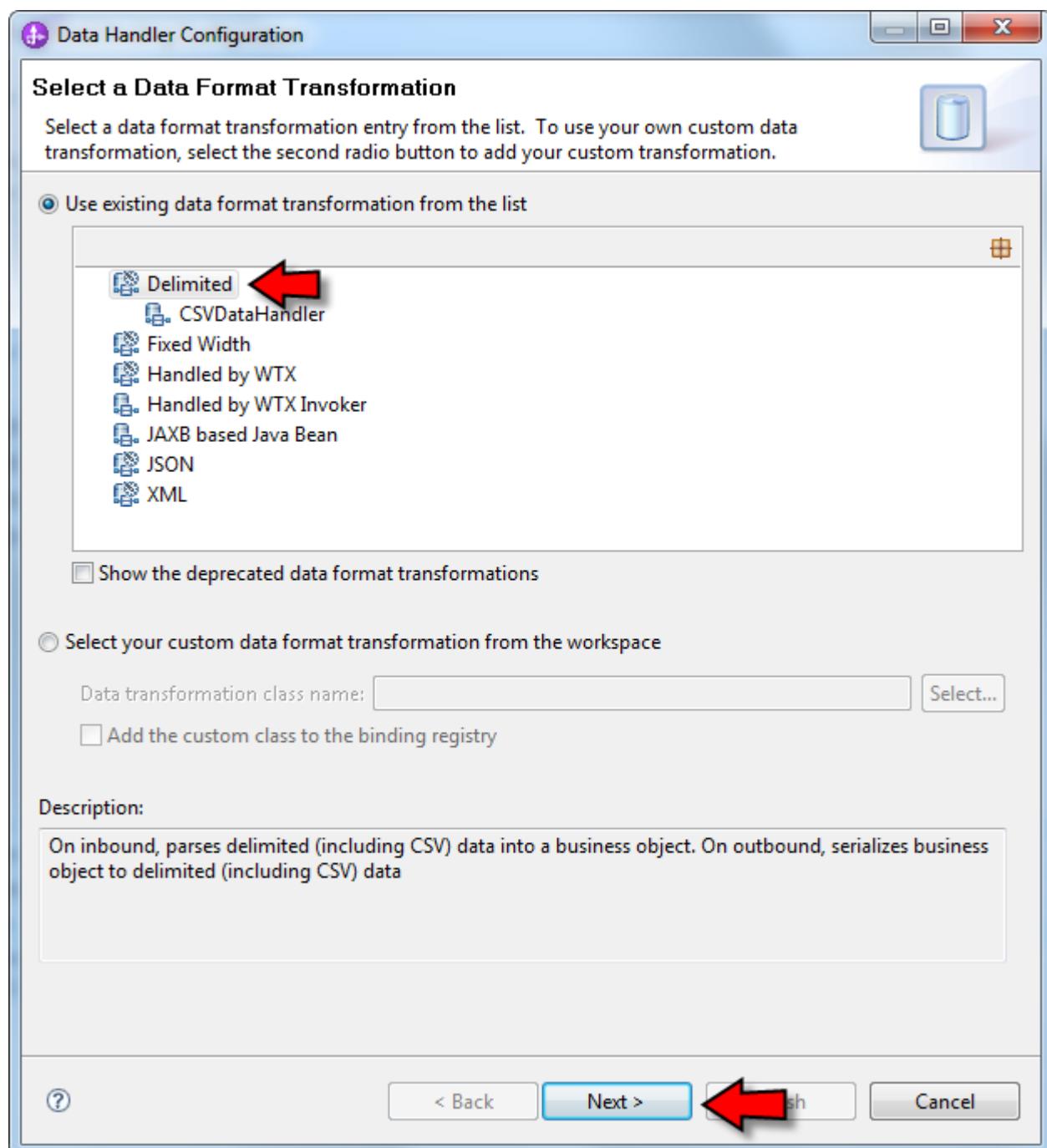
Now we get into the core wizard screens of the Flat File adapter. There are a lot of properties here covering the myriad of permutations available from the adapter.











**Data Handler Configuration**

**Specify the Data Transformation Properties**



Delimiter: \* ,

Escape character: |

Encoding: UTF-8

Header present

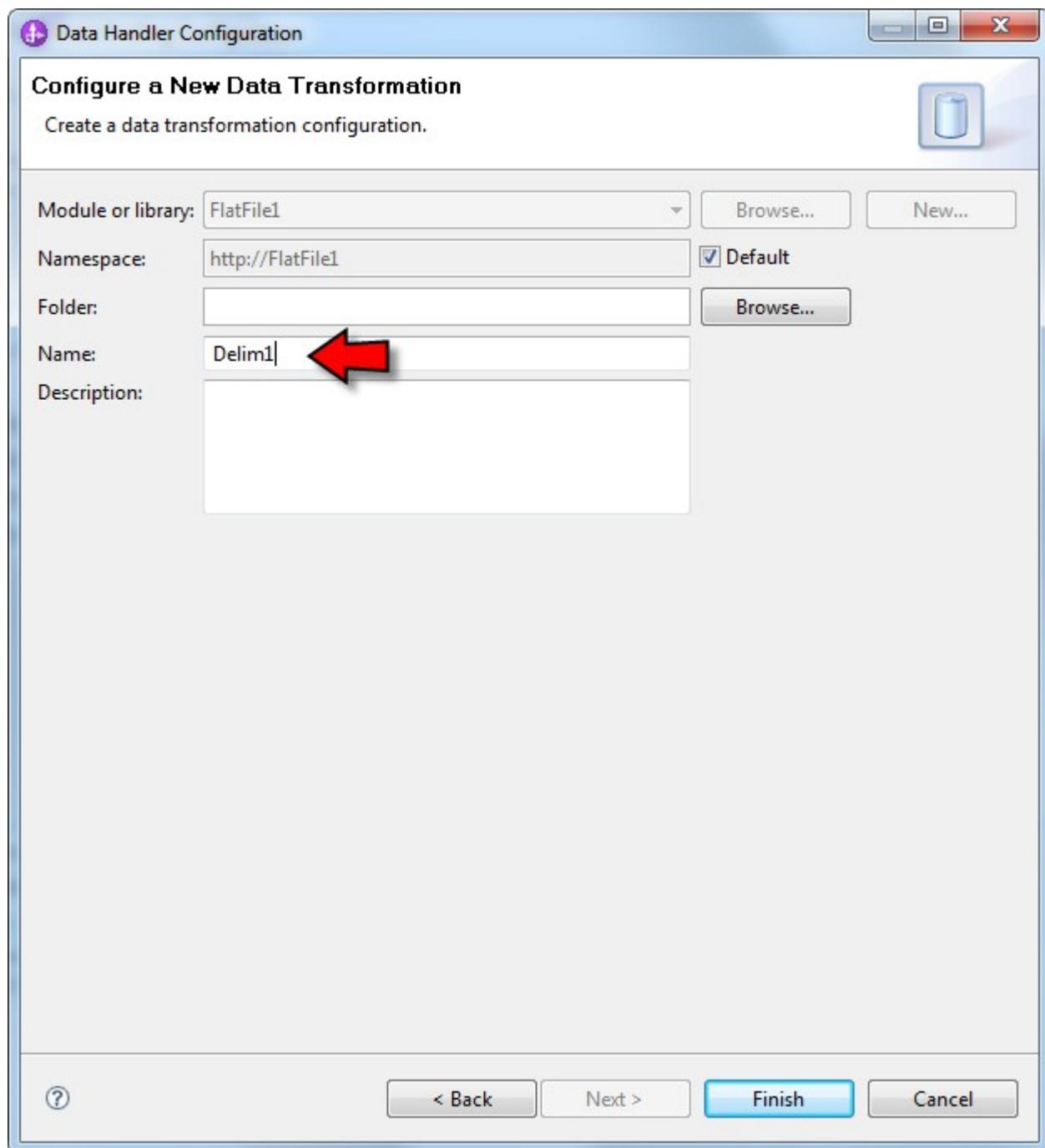
Text qualifier: "

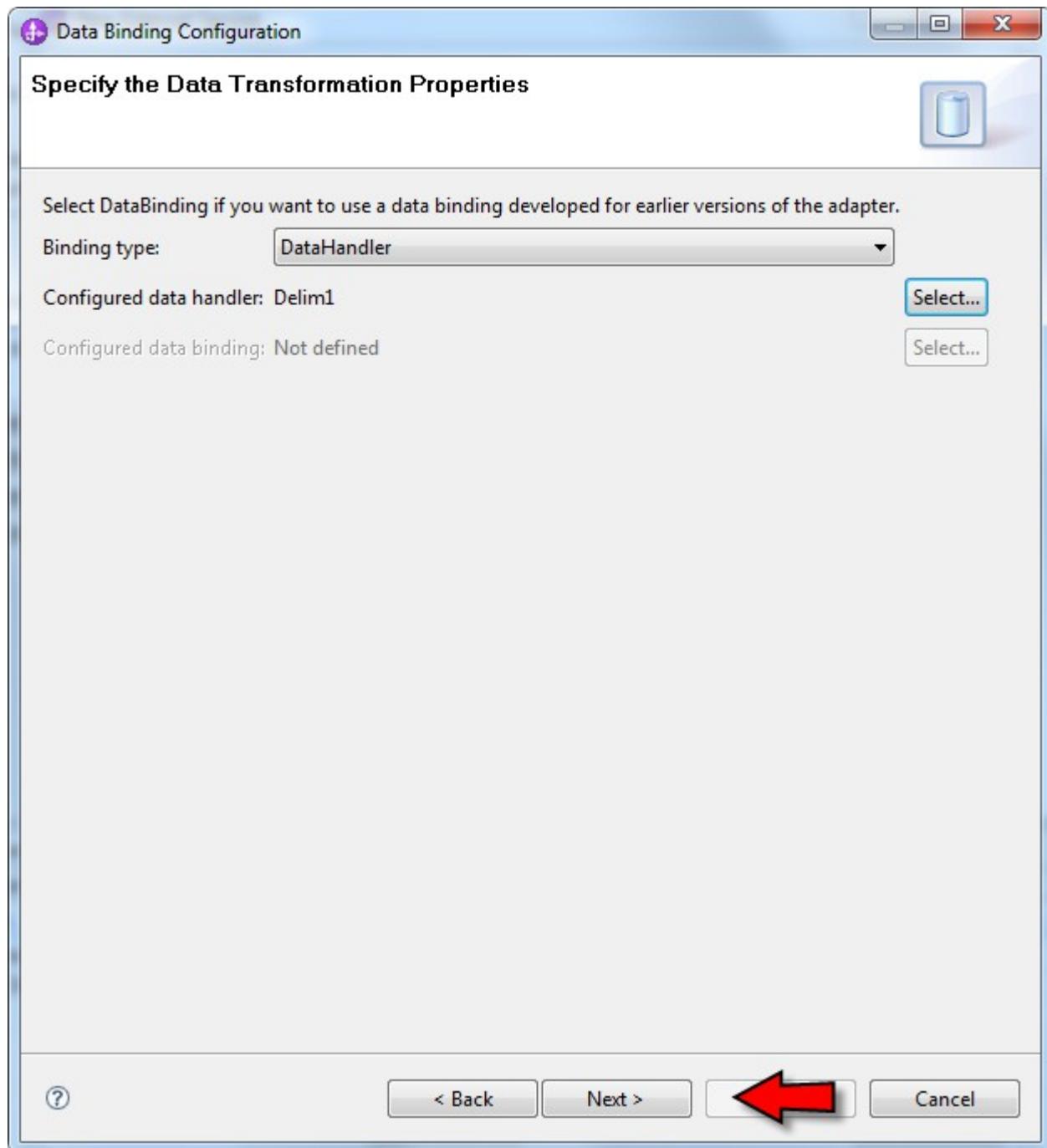
End of record delimiter: \* EOL

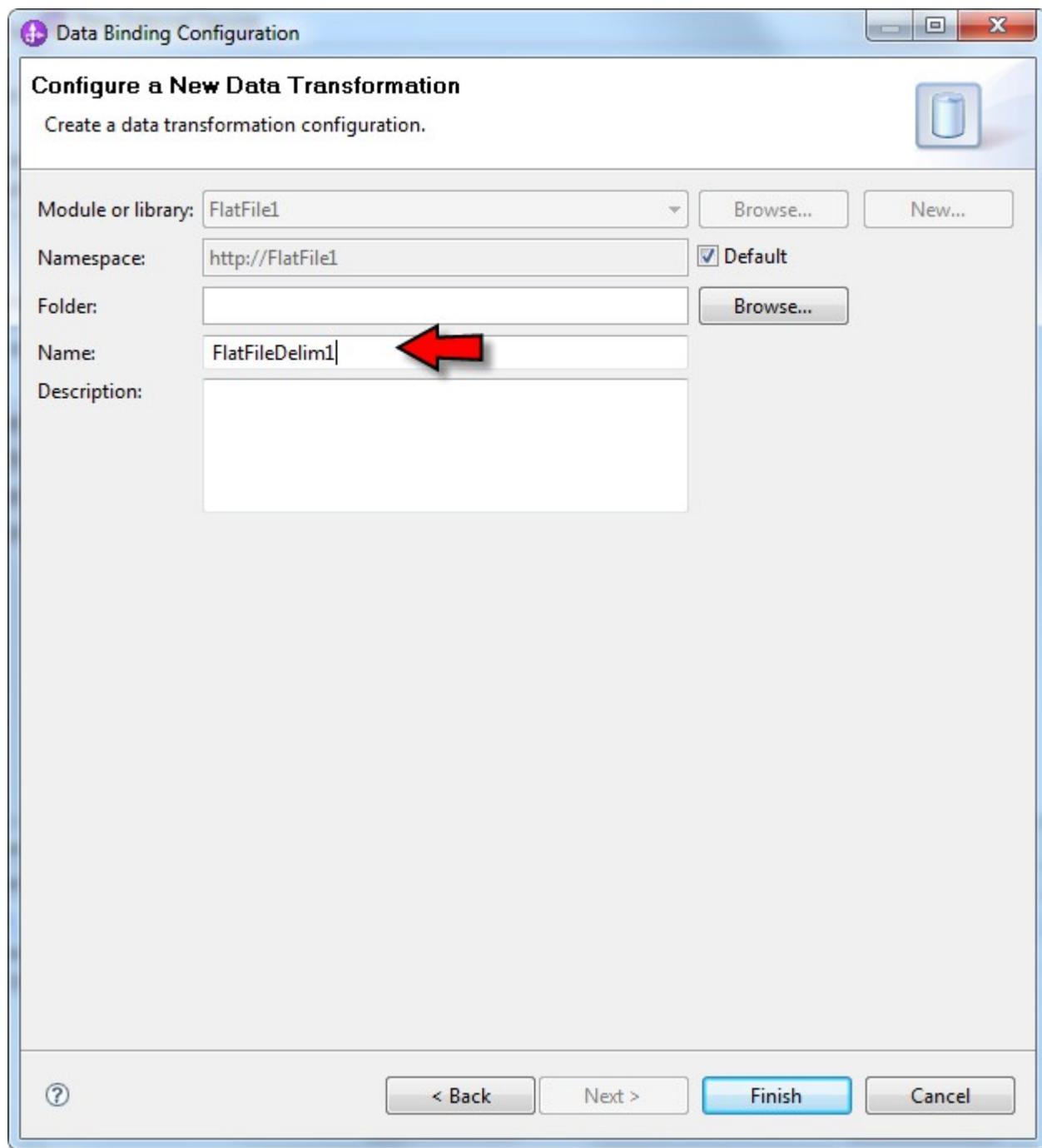
Value of null:

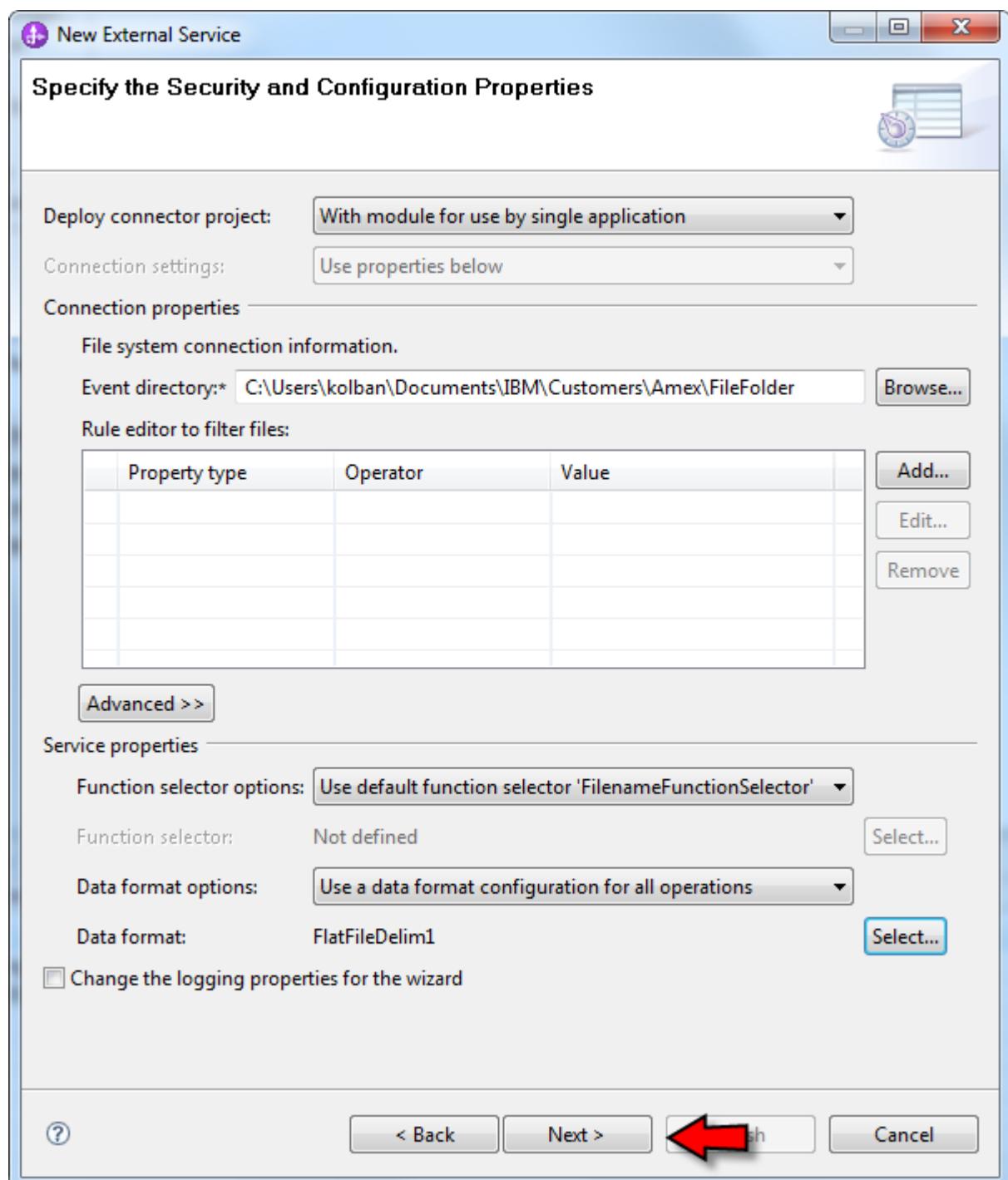
Business object name in stream

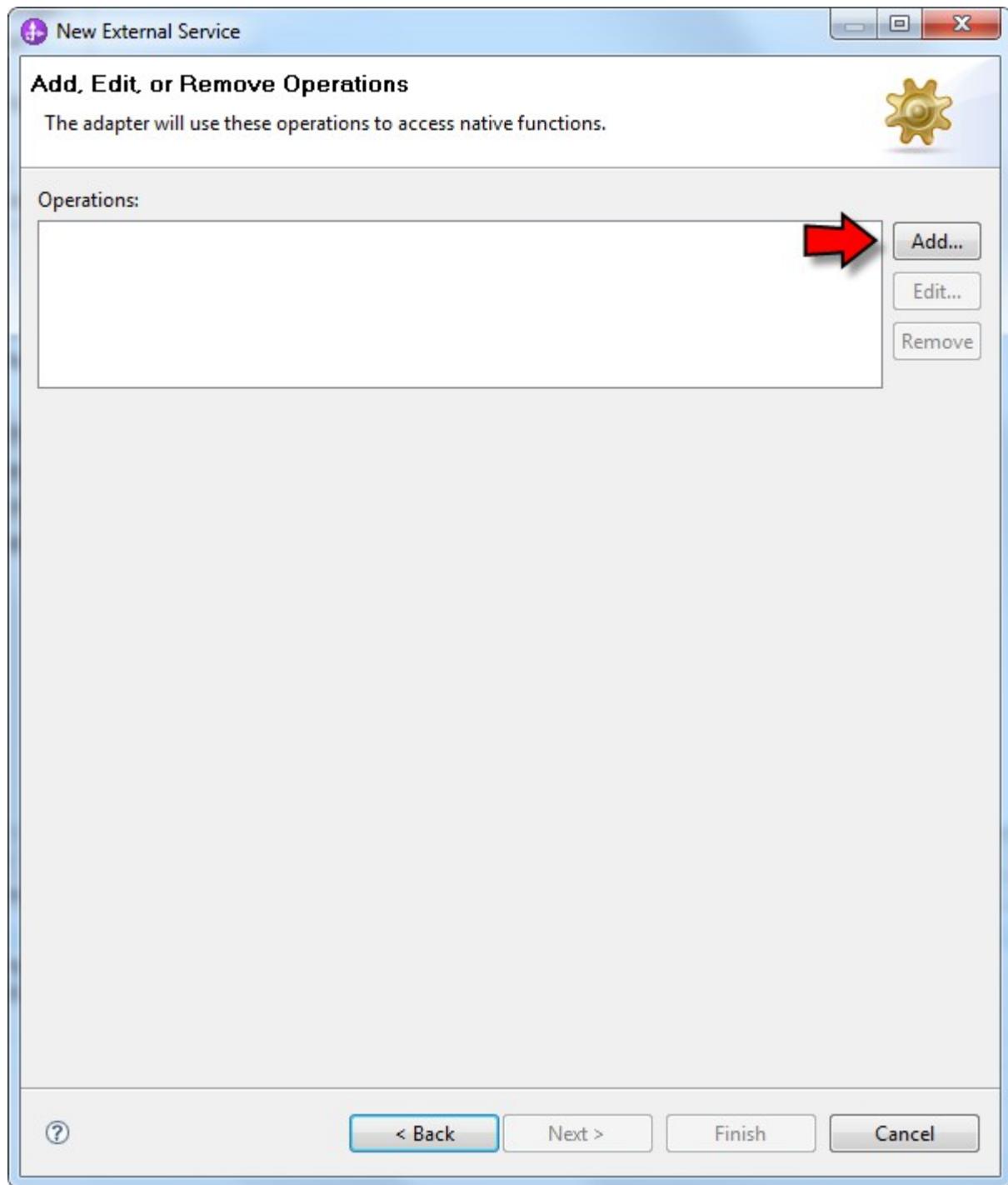
Business object namespace:

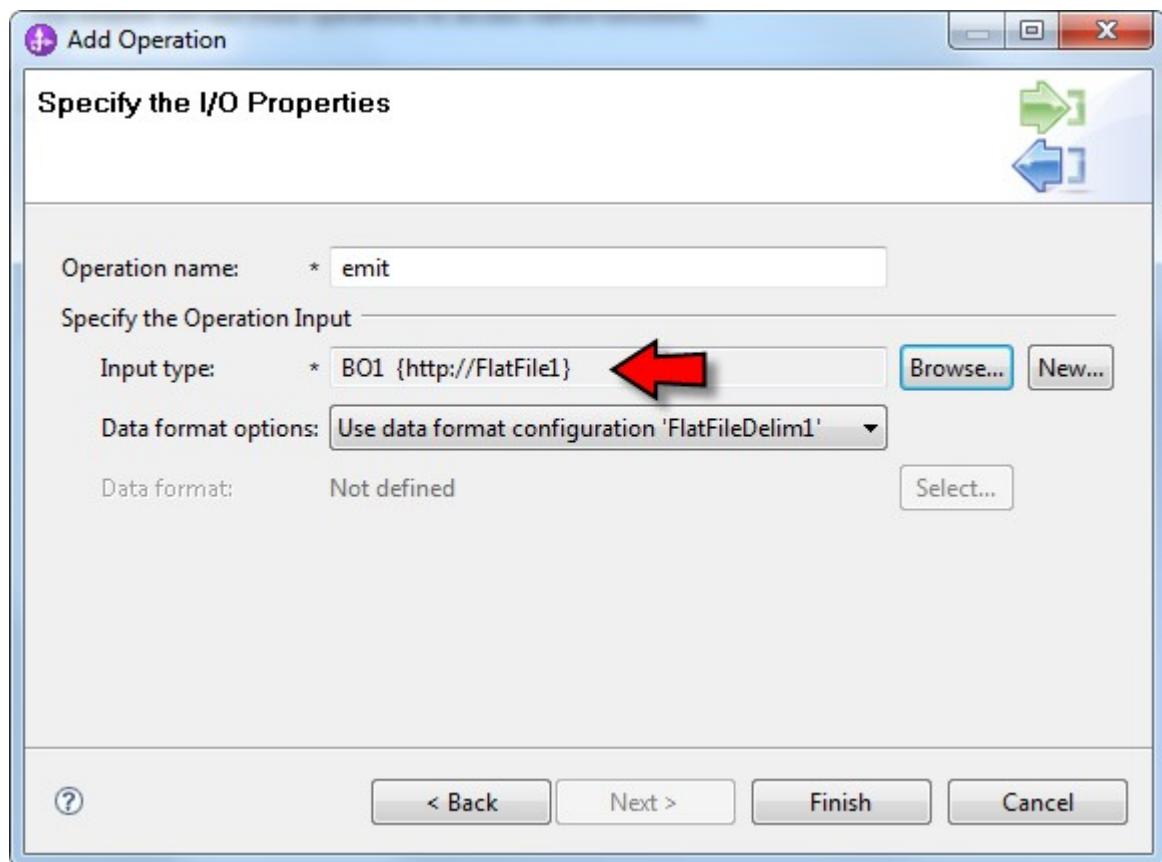
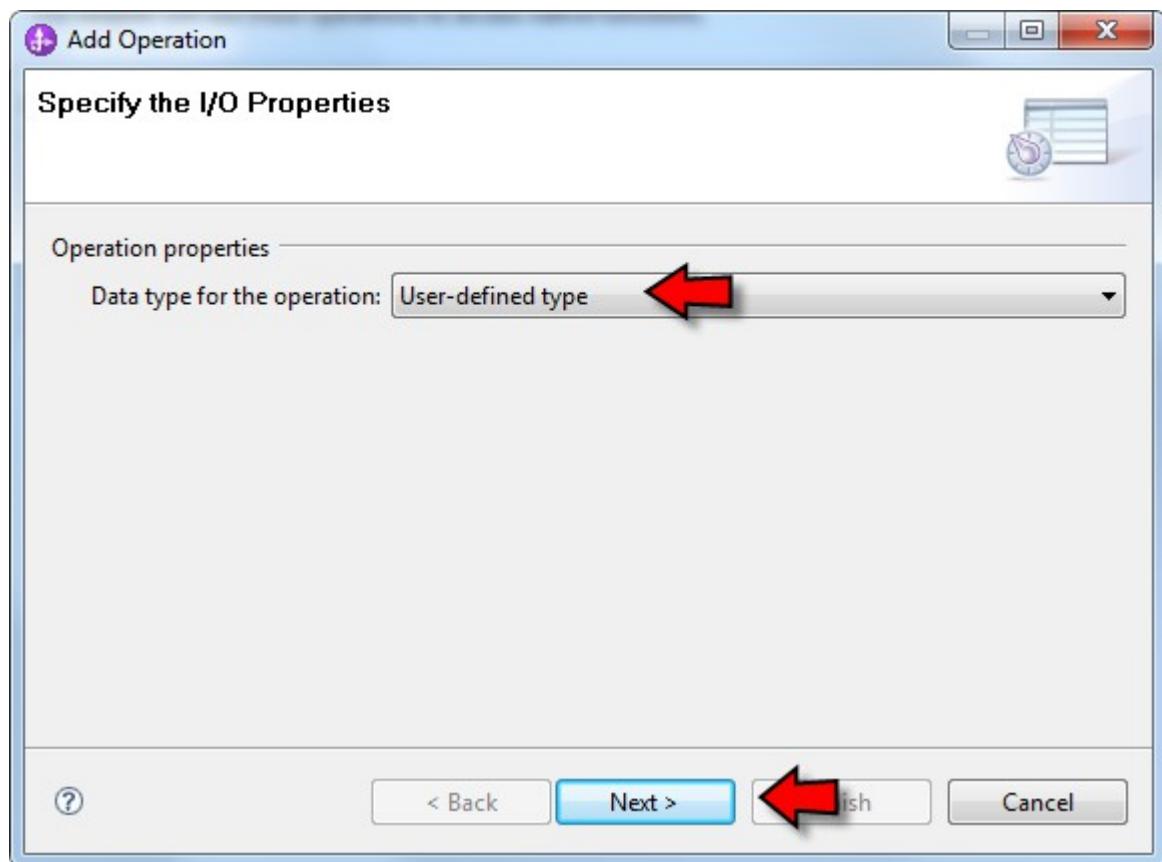


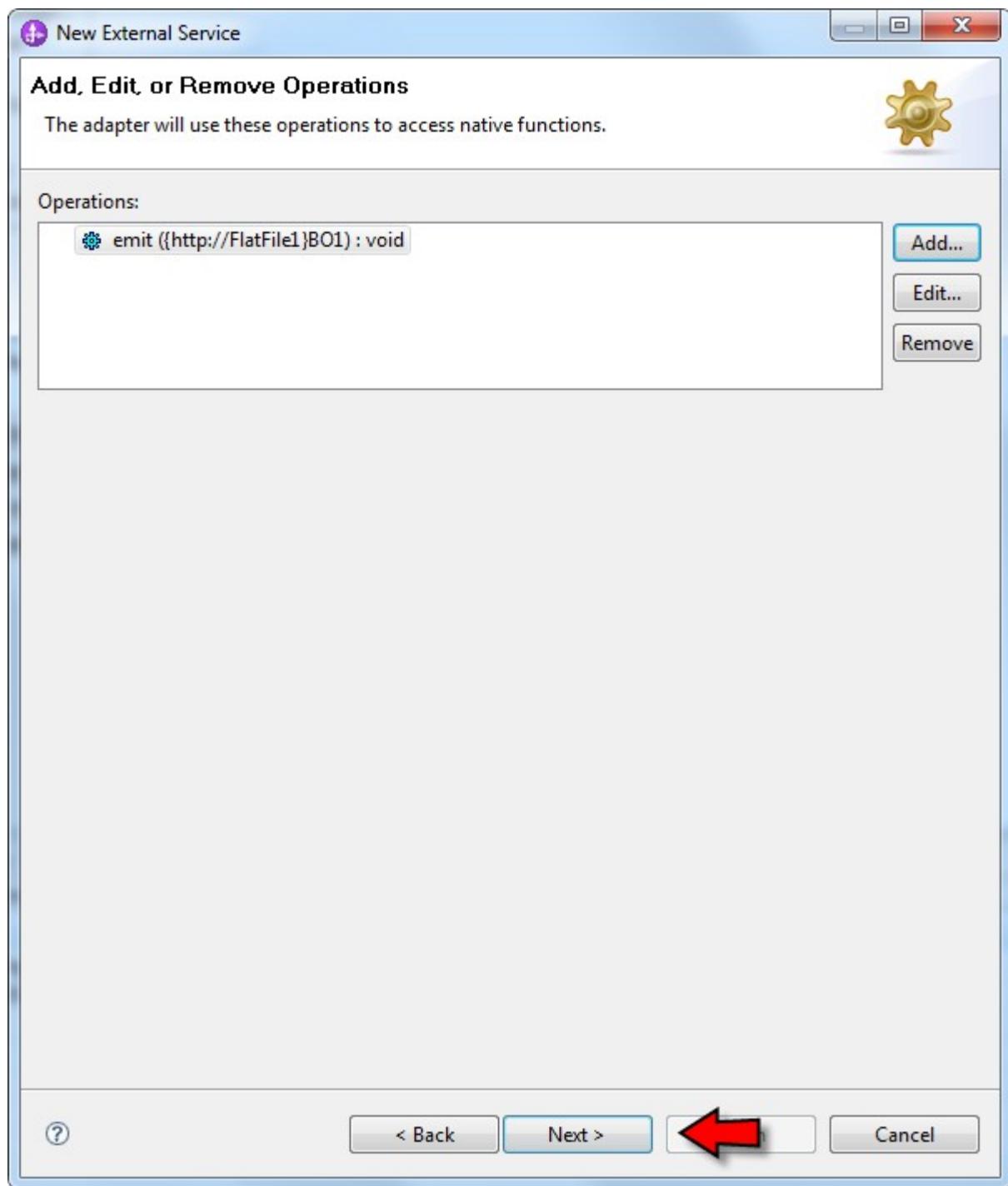


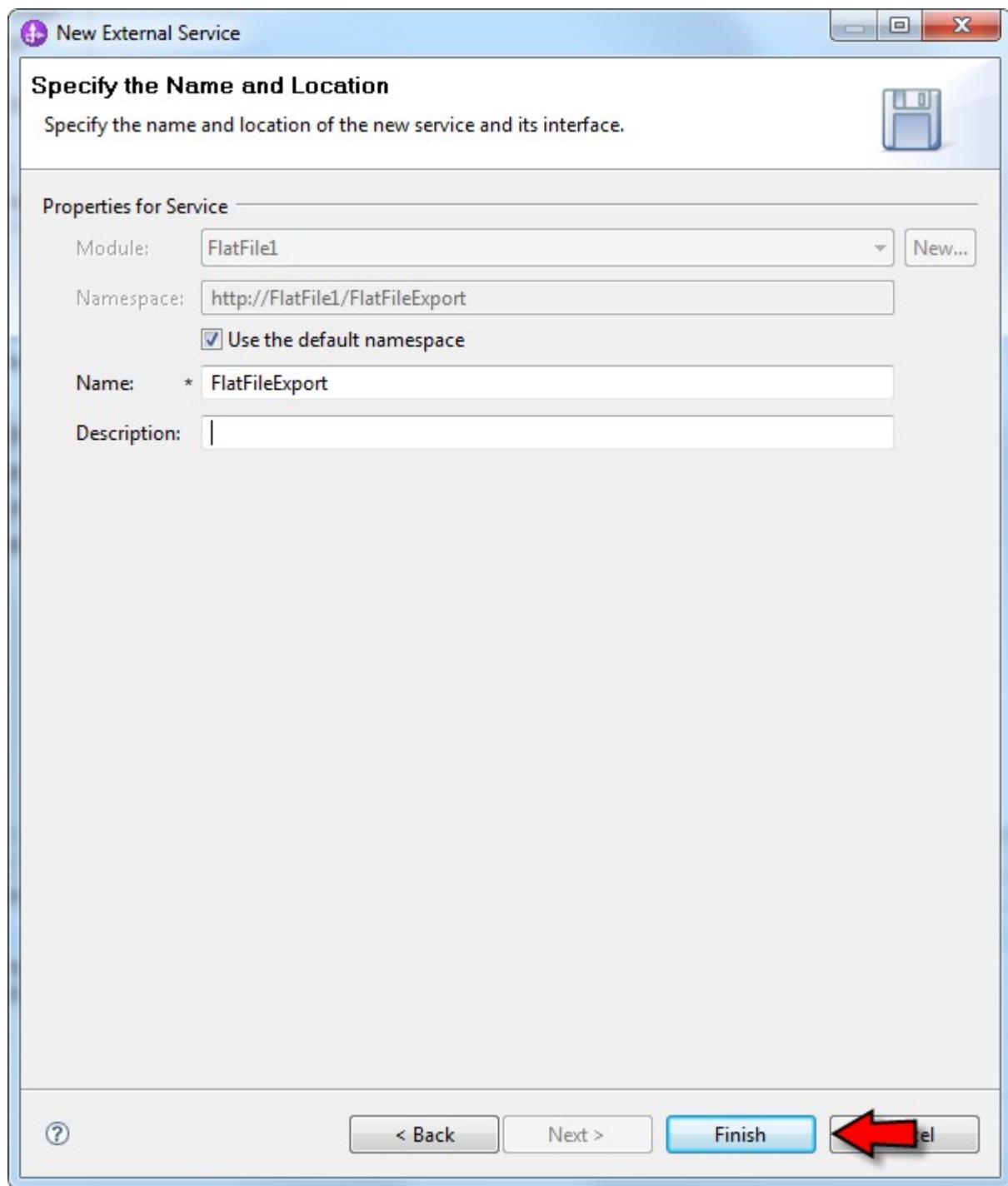












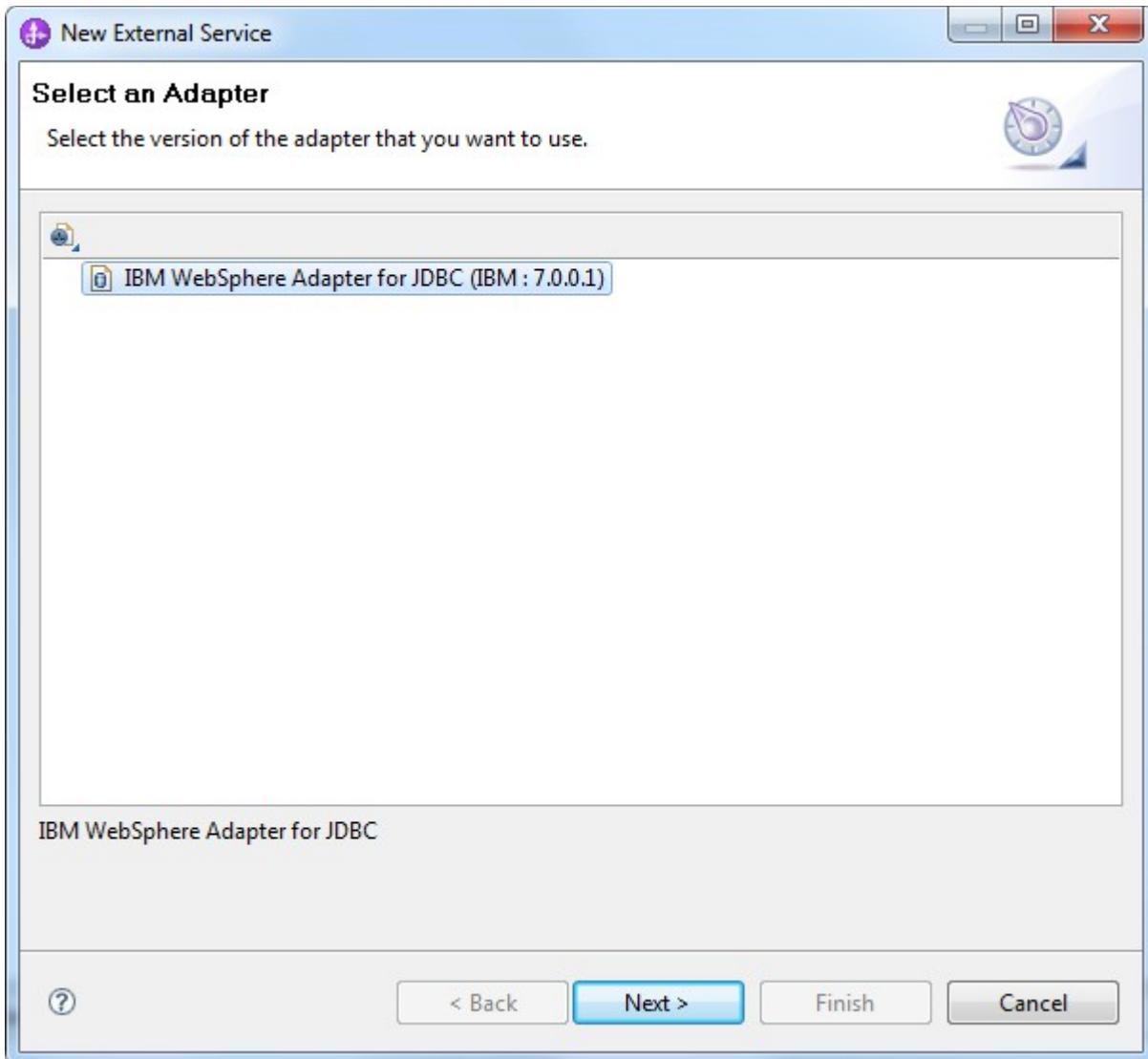
See also:

- DeveloperWorks - [SOA integration: Decouple service consumers from service providers over an ESB](#) - 2008-06-16
- DeveloperWorks - [Using WebSphere Integration Developer and WebSphere Flat File Adapter with custom data bindings](#) - 2007-10-10
- DeveloperWorks - [Getting connected with WebSphere Integration Developer adapters : Part 2, An introduction to the WebSphere Adapter for Flat Files](#) - 2007-05-09
- DeveloperWorks - [Building an ESB mediation to transform delimited files to service invocations](#) - 2006-10-04
- DeveloperWorks - [Use a generic data handler for inbound delimited flat files using WebSphere Process Server and WebSphere JCA adapter for flat files](#) - 2006-07-26
- DeveloperWorks - [Implementing custom databinding and custom function selectors in IBM WebSphere Adapters](#) – 2006-06-14

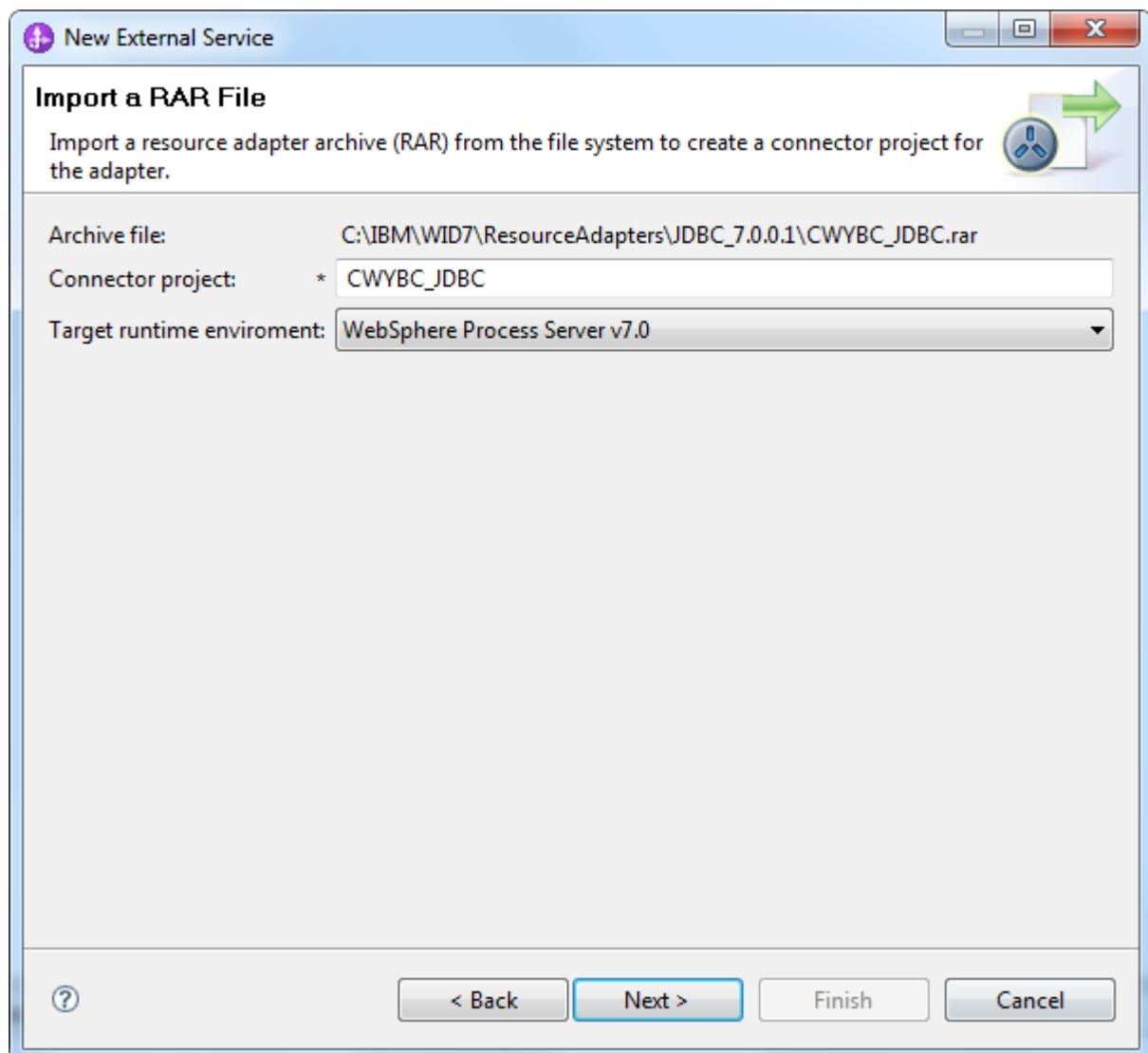
## **JDBC Adapter**

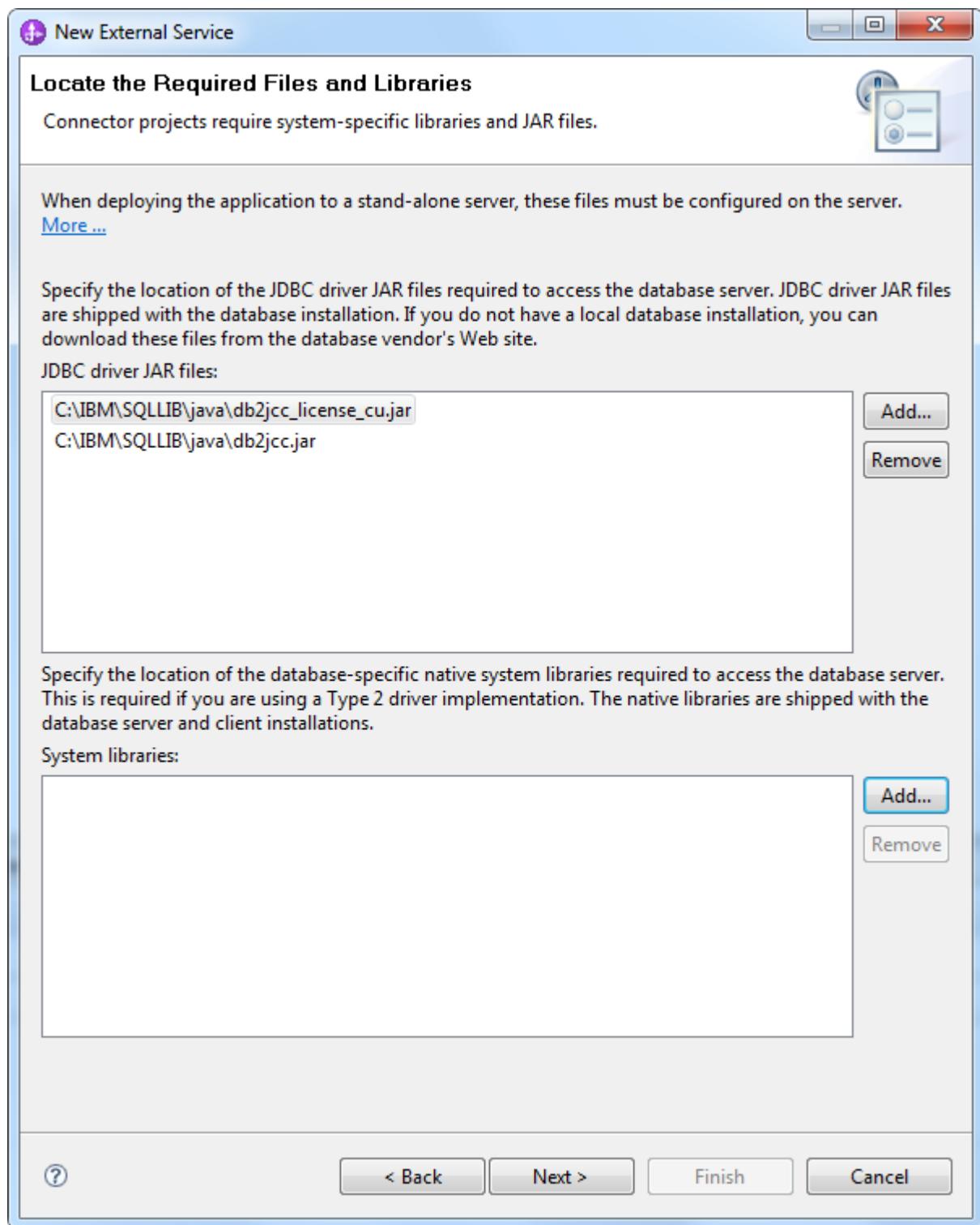
The JDBC adapter provides a service interface to database operations. The adapter utilizes the JDBC packages and hence the name of the adapter. In principle, any database that utilizes JDBC can be accessed through this adapter. This includes all the usual suspects.

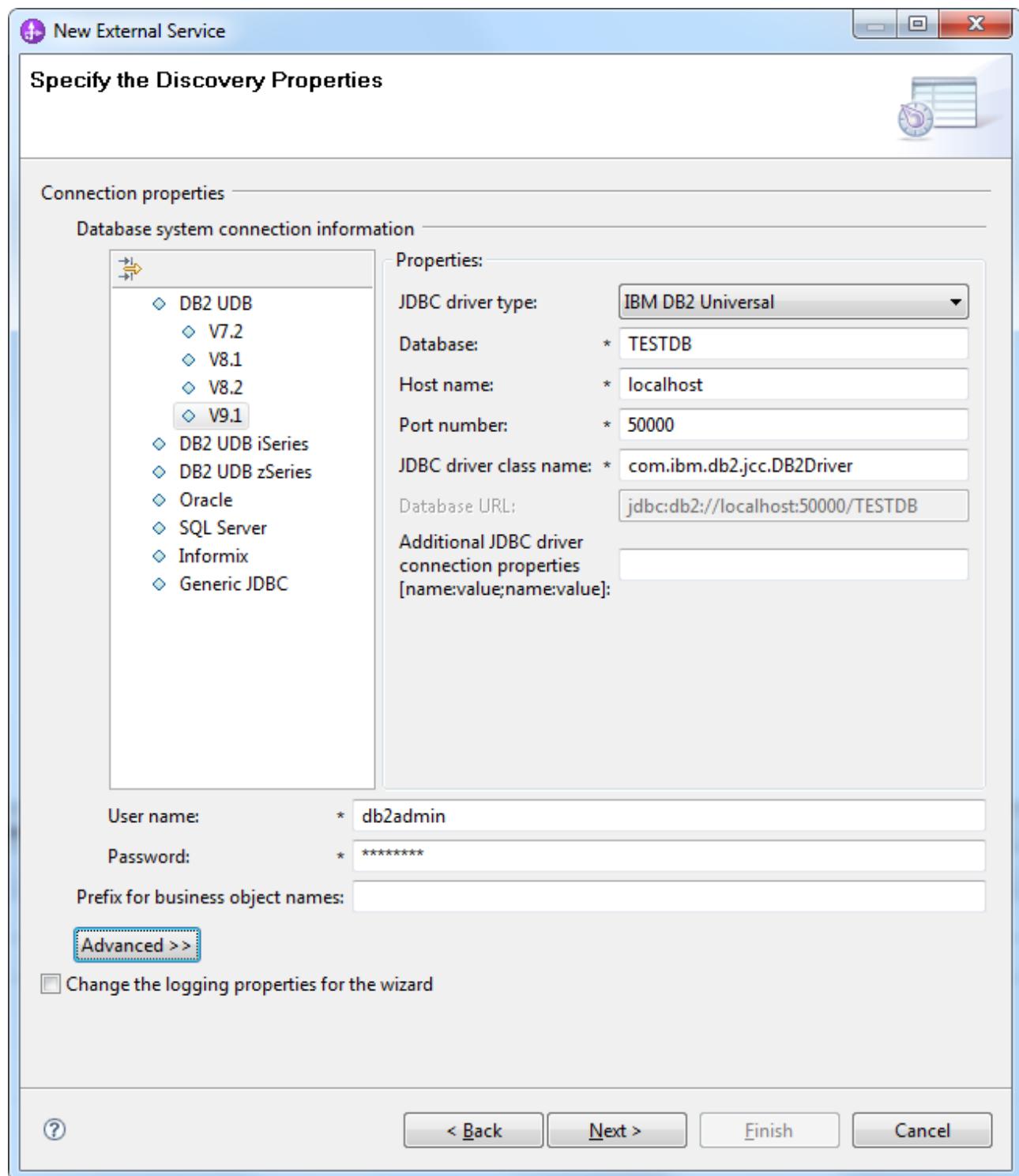
We create an outbound adapter by selecting the outbound JDBC adapter entry:



We are then asked for the name of the project to be created that will contain the JDBC configuration and code:







See also:

- DeveloperWorks - [Ensuring high availability and performance of events delivery with WebSphere Process Server and WebSphere JDBC Adapter](#) – 2009-12-02
- DeveloperWorks - [Getting connected with WebSphere Integration Developer adapters : Part 3, An introduction to the WebSphere Adapter for JDBC](#) - 2007-06-13

# Advanced Web Services

Beyond the basic concept of Web Services as used to invoke one service from another, there are advanced concepts that occasionally come into play. This section discusses some of these concepts and describes how they can be used in conjunction with IBPM.

## *WS-Notification*

WS-Notification is an open standard that overlays the concepts of Publish and Subscribe on top of Web Services. WebSphere Application Server implements v1.3 of the specification.

A NotificationProducer is a component that produces messages. Put another way, the NotificationProducer is a message publisher. In the WS-Notification specification, messages that are published are called “Notifications”. Another concept is the component known as the NotificationConsumer. Relating this back to the more common publish and subscribe terminology, this can be thought of as a subscriber. The NotificationProducer accepts requests from NotificationConsumers for subscriptions. When a subscription is made, the subscription includes information allowing the NotificationProducer to send messages to the subscriber and it also identifies which messages should be sent.

A NotificationConsumer sends a `SubscribeRequest` message to a NotificationConsumer when it wishes to register for a subscription. The subscriber is returned a reference to a `Subscription` that is managed by the `SubscriptionManager`. A `SubscriptionManager` is a service that implements the subscription management interface. The `SubscriptionManager` is considered to be a subordinate to the `NotificationProducer` and may be implemented as part of the `NotificationProducer`.

When a subscriber registers a desire for a subscription, it names the form of the subscription that should be taken. There are two possible options here. Raw notification and Notify Message notification. The Notify Message form includes additional metadata beyond the payload of the published message.

### Subscription Manager

- `GetResourceProperty`
- `Renew`
- `Unsubscribe`
- `PauseSubscription`
- `ResumeSubscription`
- `Destroy`
- `SetTerminationTime`

### Notification Broker

- `GetResourceProperty`
- `Notify`
- `Subscribe`
- `GetCurrentMessage`

- RegisterPublisher
- CreatePullPoint
- GetMessages
- DestroyPullPoint

#### Publisher Registration Manager

- DestroyRegistration
- GetResourceProperty
- Destroy
- SetTerminationTime

To subscribe to a topic, the subscriber sends a Subscribe message. The result from this is a Subscription.

A NotificationConsumer can itself be a Web Service provider. In this case, when the Notification Producer generates a message, that message is sent to the NotificationConsumer directly. This model of operation is called a “push-style” as the messages are pushed from the producer to the consumer. There may be cases where this is not possible such as the NotificationConsumer existing behind a firewall. In these circumstances, the consumer can utilize a “pull” model. The consumer can “poll” the provider to see if a message it is interested in is available for retrieval. To allow this “pull” model, a subscriber can ask the provider to create a “PullPoint”. Think of this as a location against which published messages are collected on behalf of the subscriber.

The concept of a NotificationBroker is that it relieves a publisher and a consumer from having to implement Web Services. The NotificationBroker performs the role of both a NotificationProducer and a NotificationConsumer. WAS 7.0 implements a NotificationBroker so we will focus on that for the remainder of the story. Associated with a NotificationBroker are one or more applications that are known as “Publishers”. A publisher sends messages to be published to the NotificationBroker.

### **Subscribe Request**

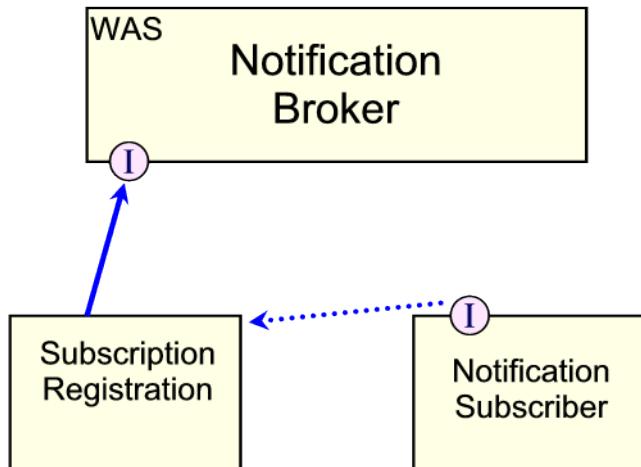
A Subscription Request is made against the Notification Broker. It may be made by the actual subscriber itself or on behalf of the subscriber. The subscription request names an Interface offered by the Subscriber such that when a publish is made, the notification broker will invoke the subscribers interface to deliver the published message. The interface offered by the subscriber **must** conform to the NotificationConsumer interface provided.

**Interface**

|               |                                     |                                    |
|---------------|-------------------------------------|------------------------------------|
| Name          | NotificationConsumer                | <a href="#">Refactor name</a>      |
| Namespace     | http://docs.oasis-open.org/wsn/bw-2 | <a href="#">Refactor namespace</a> |
| Binding Style | document literal non-wrapped        | <a href="#">More...</a>            |

**Operations**

|  | Name   | Type   |
|--|--------|--------|
|  | Notify | Notify |
|  | Inputs |        |



Here is a SOAP message that was mechanically captured for a successful subscription.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">

 <wsa:To>http://localhost:9081/weatherWSNServiceweatherWSNServicePointNB/Service</wsa:To>
 <wsa:MessageID>urn:uuid:8A37D7D3CF62D4990C1283181983897</wsa:MessageID>
 <wsa:Action>http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRequest</wsa:Action>
 </soapenv:Header>
 <soapenv:Body>
 <Subscribe
 xmlns="http://docs.oasis-open.org/wsn/b-2"
 xmlns:ns2="http://docs.oasis-open.org/wsn/t-1"
 xmlns:ns3="http://docs.oasis-open.org/wsrf/bf-2"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:ns5="http://docs.oasis-open.org/wsrf/rp-2"
 xmlns:ns6="http://docs.oasis-open.org/wsn/br-2"
 xmlns:ns7="http://docs.oasis-open.org/wsrf/r-2">
 <ConsumerReference>

 <wsa:Address>http://127.0.0.1:9081/WeatherWSNConsumerPushWeb/WeatherNotificationConsumer</wsa:Address>
 </ConsumerReference>
 <Filter>
 <TopicExpression
 Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"
 xmlns:tns="http://weather"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="TopicExpressionType">tns:daily-weather</TopicExpression>
 </Filter>

```

```

 </Subscribe>
</soapenv:Body>
</soapenv:Envelope>
```

Here is a sample response message

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<ns1:soapenv:Header xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<ns1:xsd="http://www.w3.org/2001/XMLSchema">
<ns1:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <wsa:Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeResponse</wsa:Action>
 <wsa:RelatesTo>urn:uuid:8A37D7D3CF62D4990C1283181983897</wsa:RelatesTo>
 </soapenv:Header>
 <soapenv:Body>
 <b2:SubscribeResponse xmlns:b2="http://docs.oasis-open.org/wsn/b-2">
 <b2:SubscriptionReference>
 <wsa:Address
 xmlns:wsa="http://www.w3.org/2005/08/addressing">http://win7-
x64:9080/weatherWSNServiceweatherWSNServicePointSM/Service</wsa:Address>
 <wsa:ReferenceParameters
 xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <subscriptionId
 xmlns="http://www.ibm.com/websphere/wsn/reference">sub_01ccc151d9a246c3d5c60ebc_
12ac39b203b_5</subscriptionId>
 <wsaucf:RoutingInformation
 xmlns:wsaucf="http://ucf.wsaddressing.ws.ibm.com">
 <wsaucf:HAClusterId>0000000020004747970650010575341465f5349425f4d455f5555494400
047575696400104644414345344441303303532303336</wsaucf:HAClusterId>
 <wsaucf:RoutingInformation>
 <messagingEngineUuid
 xmlns="http://www.ibm.com/websphere/wsn/reference">FDACE4DA03052036</messagingEn
gineUuid>
 <wsaucf:VirtualHostName
 xmlns:wsaucf="http://ucf.wsaddressing.ws.ibm.com">default_host</wsaucf:VirtualHo
stName>
 </wsa:ReferenceParameters>
 <wsa:Metadata xmlns:wsa="http://www.w3.org/2005/08/addressing">
 <ServiceName EndpointName="SubscriptionManagerPort"
 xmlns="http://www.w3.org/2006/05/addressing/wsdl">
 <ns1:ns1121979737="http://www.ibm.com/websphere/wsn/subscription-
manager">ns1121979737:weatherWSNServiceweatherWSNServicePointSM</ServiceName>
 </wsa:Metadata>
 </b2:SubscriptionReference>
 <b2:CurrentTime>2010-08-30T15:26:17.659Z</b2:CurrentTime>
 </b2:SubscribeResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

To make a subscription from a mediation flow, the SMO has to be constructed as follows:

```

=headers/SOAPHeader[1]/nameSpace
 http://www.w3.org/2005/08/addressing

=headers/SOAPHeader[1]/name
```

## Action

```
/headers/SOAPHeader[1]/value (Cast: anyURI)
 http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRequest

=headers/SOAPHeader[2]/nameSpace
 http://www.w3.org/2005/08/addressing

=headers/SOAPHeader[2]/name
 MessageID

=headers/SOAPHeader[2]/value
 someUUID

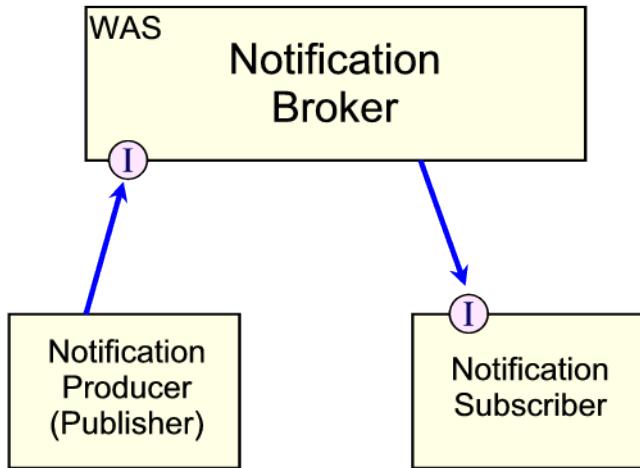
/body/Subscribe/ConsumerReference/Address/value
 WS-Hostname&Port

/body/Subscribe/Filter/TopicExpression/@Dialect (Cast: TopicExpression)
 http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple

/body/Subscribe/Filter/TopicExpression/*mixed
 topic value
```

## Publishing a message

In order to publish a message, the Notification Producer uses the Web Services interface provided by the Notification Broker and calls the Notify operation passing in the message to be published.



As of 2011-04, it appears that the notification message *must* be a single line of text. Errors are reported otherwise. This problem is claimed to be fixed in WAS 7.0.0.9 and above and there is an APAR with a fix for releases prior to that.

Here is a sample published message:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">

 <wsa:To>http://localhost:9081/weatherWSNServiceweatherWSNServicePointNB/Service</wsa:To>
 <wsa:ReplyTo>
 <wsa:Address>http://www.w3.org/2005/08/addressing/none</wsa:Address>
```

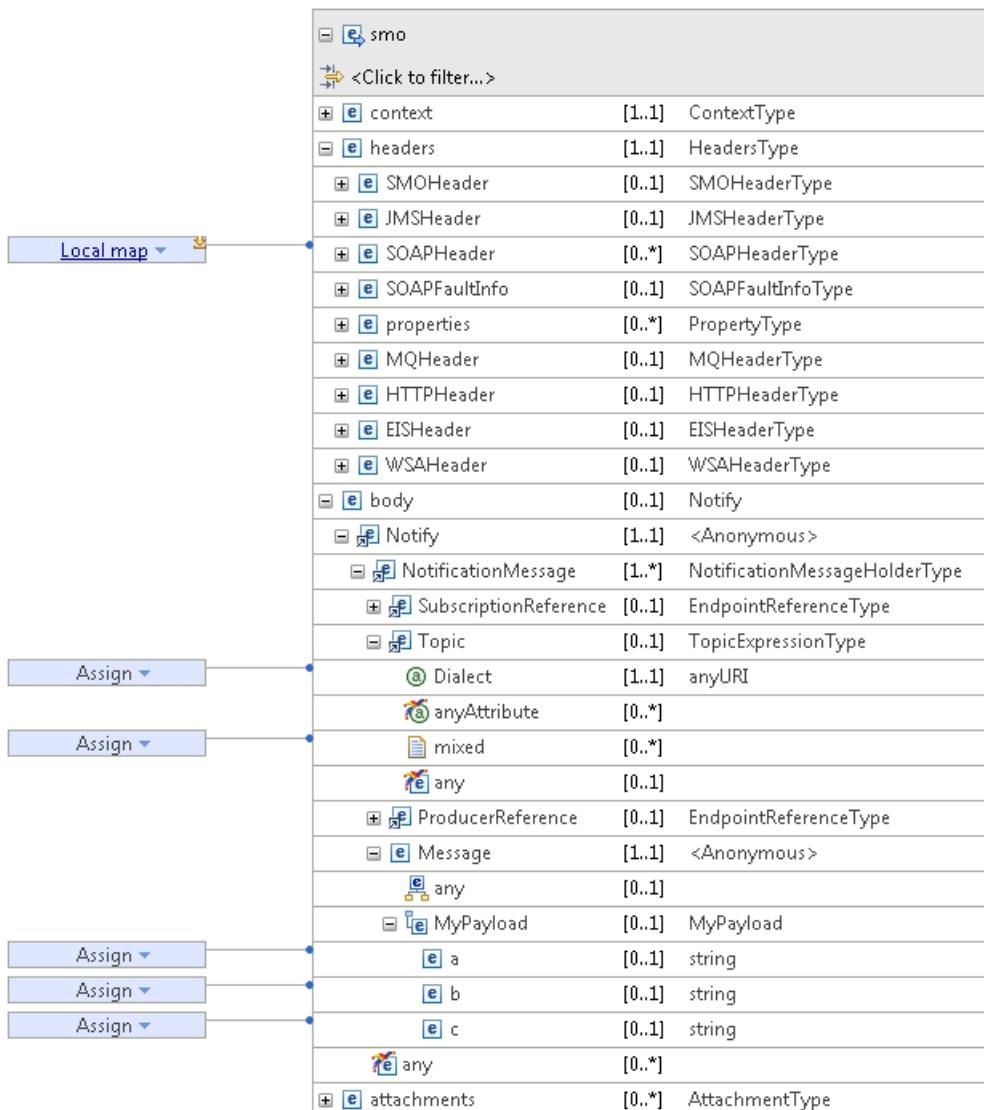
```

</wsa:ReplyTo>
<wsa:MessageID>urn:uuid:8A37D7D3CF62D4990C1283269176009</wsa:MessageID>
<wsa:Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationConsumer/Notify</wsa:Action>
</soapenv:Header>
<soapenv:Body>
<Notify xmlns="http://docs.oasis-open.org/wsn/b-2"
 xmlns:ns2="http://docs.oasis-open.org/wsrf/bf-2"
 xmlns:ns3="http://www.w3.org/2005/08/addressing"
 xmlns:ns4="http://docs.oasis-open.org/wsrf/rp-2"
 xmlns:ns5="http://docs.oasis-open.org/wsn/t-1"
 xmlns:ns6="http://docs.oasis-open.org/wsn/br-2"
 xmlns:ns7="http://docs.oasis-open.org/wsrf/r-2">
 <NotificationMessage>
 <Topic
 Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"
 xmlns:tns="http://weather"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="TopicExpressionType">tns:daily-weather</Topic>
 <Message>
 <weatherobj xmlns="">
 xmlns:ns8="http://docs.oasis-open.org/wsn/b-2">partly cloudy,-
5,SW,39,1283269151817</weatherobj>
 </Message>
 </NotificationMessage>
 </Notify>
</soapenv:Body>
</soapenv:Envelope>

```

### Using a mediation flow to build a WS-Notification message

The messages to be sent to the NotificationBroker can be built by a WESB mediation flow. This involves a little work but once understood isn't that complex. Here is an example of a Notify message:



To describe these mapings in this document, an XPath like notation will be used:

```
/headers/SOAPHeader[1]/nameSpace
http://www.w3.org/2005/08/addressing

=headers/SOAPHeader[1]/name
Action

=headers/SOAPHeader[1]/value (Cast: anyURI)
http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify

/body/Notify/NotificationMessage/Topic/@Dialect
http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple

/body/Notify/NotificationMessage/Topic/mixed
daily-weather

/body/Notify/NotificationMessage/Message/MyPayload (Cast: as appropriate)
data
```

## Topics for WS-Notification

When a subscription is made, it is made on a particular topic. This describes *which* publications that notifiers publish will be delivered to a given subscriber. A Topic is actually considered a topic expression and has rules that govern its structure.

Here is a simple example of the Topic section:

```
<Topic
 Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"
 xmlns:tns="http://weather"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="TopicExpressionType">tns:daily-weather</Topic>
```

In general, a Topic is contained within a <Topic> tag. This tag has a mandatory attribute called Dialect which will be discussed shortly. The Dialect attribute describes how the Topic should be interpreted. The body of the <Topic> tag identifies the topic.

There are three classes of topic expressions.

### Simple Topic Expression

The class known as Simple Topic Expression has the dialect attribute of the <Topic> tag set to:

Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple"

The topic value is an explicit value with no hierarchy descriptions nor wild cards.

### Concrete Topic Expression

This class known as Concrete Topic Expression has the dialog attribute of the <Topic> tag set to:

Dialect="<http://docs.oasis-open.org/wsn/t-1/TopicExpression/Concrete>"

The topic value is again an explicit value but with the addition of "/" characters to represent a hierarchy.

### Full Topic Expression

The class known as Full Topic Expression has the dialect attribute of the <Topic> tag set to:

Dialect="<http://docs.oasis-open.org/wsn/t-1/TopicExpression/Full>"

The topic value can use XPath notation to define the topic.

## Building a Notification Subscriber

A Notification Subscriber is a Web Service implementation that implements the Notification Consumer service interface. The Notification Subscriber is the “application” that is destined to receive the notifications that a publisher has published a message. This interface has a single operation on it called “Notify”:

|               |                                     |                                    |
|---------------|-------------------------------------|------------------------------------|
| Name          | NotificationConsumer                | <a href="#">Refactor name</a>      |
| Namespace     | http://docs.oasis-open.org/wsn/bw-2 | <a href="#">Refactor namespace</a> |
| Binding Style | document literal non-wrapped        | <a href="#">More...</a>            |



Operations and their parameters

| Name            | Type   |
|-----------------|--------|
| <b>▼ Notify</b> |        |
| Inputs          | Notify |

It is the service endpoint of the NotificationSubscriber that is supplied in a subscription request. This means that when a publisher publishes a message, the broker can look at its current set of subscriptions and for each subscription matching the publish, send a message to the subscriber by invoking its service interface.

A trivial implementation of the subscriber interface may look as follows:



The export is given the NotificationConsumer interface and a Web Services binding is defined. Here is an example of an SMO generated as a result of a published message received by a subscriber.

```

<?xml version="1.0" encoding="UTF-8"?>
<Se:smo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:Se="http://www.ibm.com/websphere/sibx/smox/v6.0.1"
 xmlns:_2="wsdl:http://docs.oasis-open.org/wsn/bw-2"
 xmlns:_2_1="http://docs.oasis-open.org/wsn/b-2"
 xmlns:ad="http://www.w3.org/2005/08/addressing"
 xmlns:ht="http://www.ibm.com/xmlns/prod/websphere/http/sca/6.1.0"
 xmlns:re="http://www.ibm.com/websphere/wsn/reference"
 xmlns:uc="http://ucf.wsaddressing.ws.ibm.com"
 xmlns:wsdl="http://www.w3.org/2006/05/addressing/wsdl">
 <context/>
 <headers>
 <SMOHeader>
 <MessageUUID>ED0819C0-012A-4000-E000-1910C0A8016F</MessageUUID>
 <Version>
 <Version>7</Version>
 <Release>0</Release>
 <Modification>0</Modification>
 </Version>
 <MessageType>Request</MessageType>
 <Operation>Notify</Operation>
 <Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationConsumer/Notify</Action>
 <SourceNode>MyConsumer</SourceNode>
 <SourceBindingType>WebService</SourceBindingType>
 <Interface>wsdl:http://docs.oasis-open.org/wsn/bw-2</Interface>
 </SMOHeader>
 <HTTPHeader>
 <control>
 <ht:URL>http://localhost:9080/WSNMod1Web/sca/MyConsumer</ht:URL>
 </control>
 </HTTPHeader>
 </headers>
</Se:smo>

```

```

<WSAHeader version="http://www.w3.org/2005/08/addressing">
 <ad:To>http://localhost:9081/WSNMod1Web/sca/MyConsumer</ad:To>
 <ad:ReplyTo>
 <ad:Address>http://www.w3.org/2005/08/addressing/none</ad:Address>
 </ad:ReplyTo>
 <ad:Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationConsumer/Notify</ad:Action>
 <ad:MessageID>urn:uuid:8B98FC5FF77DBF4EA81283876991522</ad:MessageID>
</WSAHeader>
</headers>
<body xsi:type="_2:Notify">
 <_2_1:Notify>
 <_2_1:NotificationMessage>
 <_2_1:SubscriptionReference>
 <ad:Address>http://win7-
x64:9080/weatherWSNService/weatherWSNServicePointSM/Service</ad:Address>
 <ad:ReferenceParameters>

<re:subscriptionId>sub_d0e1004de6f28244ac58deb0_12aecf2aee3_1</re:subscriptionId
>

<uc:RoutingInformation><uc:HAClusterId>00000000020004747970650010575341465f53494
25f4d455f55554944000475756964001041453745313633354531454645353134</uc:HAClusterI
d></uc:RoutingInformation>
 <re:messagingEngineUuid>AE7E1635E1EFE514</re:messagingEngineUuid>
 <uc:VirtualHostName>default_host</uc:VirtualHostName>
</ad:ReferenceParameters>
<ad:Metadata>
 <wsdl:ServiceName
EndpointName="SubscriptionManagerPort">ns1121979737:weatherWSNService/weatherWSNS
ervicePointSM</wsdl:ServiceName>
 <ad:Metadata>
 </_2_1:SubscriptionReference>
 <_2_1:Topic Dialect="http://docs.oasis-open.org/wsn/t-
1/TopicExpression/Simple">daily-weather</_2_1:Topic>
 <_2_1:Message>
 <MyPayload>
 <a>1
 2
 <c>3</c>
 </MyPayload>
 </_2_1:Message>
 </_2_1:NotificationMessage></_2_1:Notify>
</body>
</Se:smo>

```

## A WS-Notification Sample

Here we will build a complete WS-Notification sample. We will use the simple story domain of stock prices for our illustration. We will have a publisher that is publishing stock updates. We will also have a subscriber that is watching for such updates.

We will start by setting up a WS-Notification environment. WAS implements WS-Notification on top of the WAS SI Bus JMS technology. We could re-use an existing SI Bus but for this story, we will create a new bus as it is a relatively quick task.

First we create a new SI Bus to host our Stocks topic environment.

The screenshot shows the 'Buses' management interface. At the top, there's a header with the title 'Buses'. Below it is a section titled 'Buses' with a descriptive text about service integration buses. A red arrow points to the 'New' button in the toolbar, which is used to create a new bus. The main area displays a table of existing buses with columns for Select, Name, Description, and Security. One row is highlighted in blue, showing a bus named 'weatherWSNBus' with a security status of 'Disabled'.

The screenshot shows the 'Create a new Service Integration Bus' wizard. It is on Step 1: Create a new bus. The left sidebar shows 'Step 1: Create a new bus' and 'Step 2: Confirm creation of new bus'. The main panel is titled 'Create a new bus' and contains a form to configure the bus attributes. A red arrow points to the 'Enter the name for your new bus' input field, which contains the value 'StockBus'. There is also a checkbox for 'Bus security' which is unchecked. At the bottom, there are 'Next' and 'Cancel' buttons.

We have decided to switch off SI Bus security at this time as this only complicates the overall story. Once the bus has been created, at least one bus member must be added.

**Buses**

**Buses > StockBus**

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Configuration Local Topology

**General Properties**

Name: StockBus  
UUID: 86B76AD37B399B45  
Description:

**Topology**

- Bus members
- Messaging engines
- Foreign bus connections
- Bootstrap members

**Destination resources**

- Destinations
- Mediations

**Services**

Standard options were selected and the local server was added as a Bus Member.

Following the creation of a new SI Bus, we can now define a new WS-Notification environment. This can be found here:

Service integration

- Buses
- WSRR definitions
- SCM connectivity providers
- Common Event Infrastructure
- Web services
- WS-Notification
  - Services **←**
  - JAX-WS Handlers
  - JAX-WS Handler Lists
- Service Integration Bus Browser

**WS-Notification services**

**WS-Notification services**

A WS-Notification service provides access to service integration bus resources for Web services publish and subscribe clients.

Preferences

New **←**

Select Name Service integration bus name Type Description

You can administer the following resources:

|                          |                   |               |      |
|--------------------------|-------------------|---------------|------|
| <input type="checkbox"/> | weatherWSNService | weatherWSNBus | V7.0 |
|--------------------------|-------------------|---------------|------|

Total 1

**New WS-Notification service**

A WS-Notification service provides access to service integration bus resources for Web services publish and subscribe clients.

→ Step 1: Configure name, description, service integration bus and dynamic topic namespace settings

Step 2: Select WS-Notification service type

Step 3: Configure handler and web service policy settings

Step 4: Create WS-Notification service points

Step 5: Create permanent topic namespaces

Step 6: Summary

**Configure name, description, service integration bus and dynamic topic namespace settings**

Supply a name and description for the WS-Notification service

\* Name  

Description

Enable dynamic topic namespaces?

Requires registration

Service integration bus name  

**Next** | **Cancel**

**New WS-Notification service**

A WS-Notification service provides access to service integration bus resources for Web services publish and subscribe clients.

Step 1: Configure name, description, service integration bus and dynamic topic namespace settings

→ Step 2: Select WS-Notification service type

Step 3: Configure handler and web service policy settings

Step 4: Create WS-Notification service points

Step 5: Create permanent topic namespaces

Step 6: Summary

**Select WS-Notification service type**

Please select the type of WS-Notification service to create

**Version 7.0**

Select this type of service if you wish to use policy set to compose your new WS-Notification service with other web service standards, such as WS-ReliableMessaging.

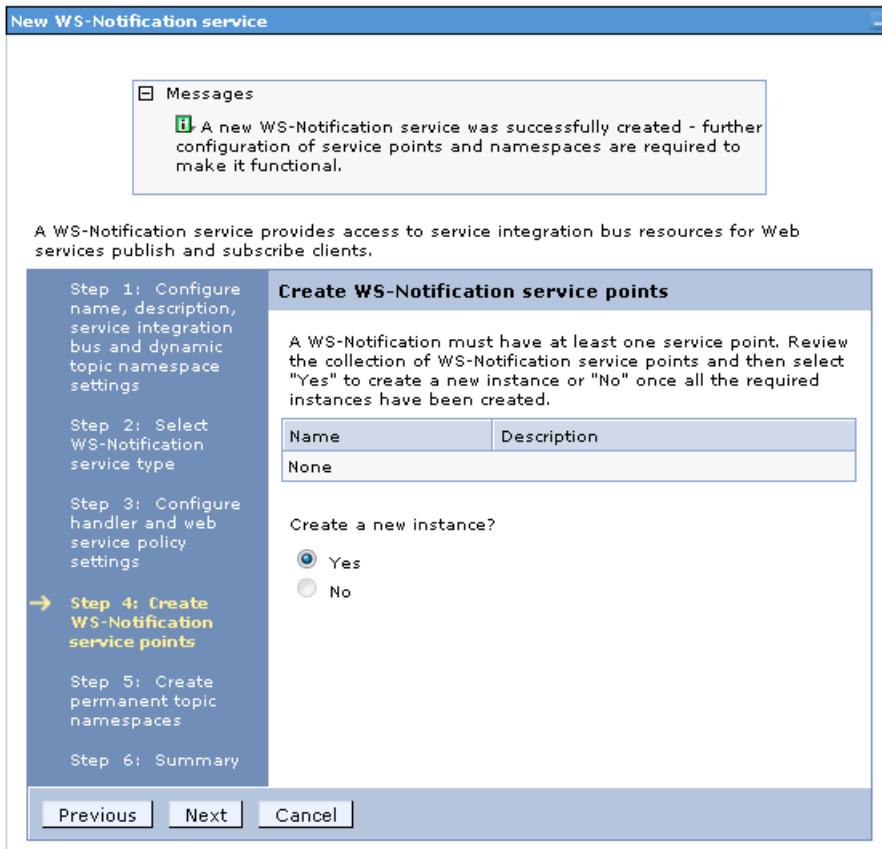
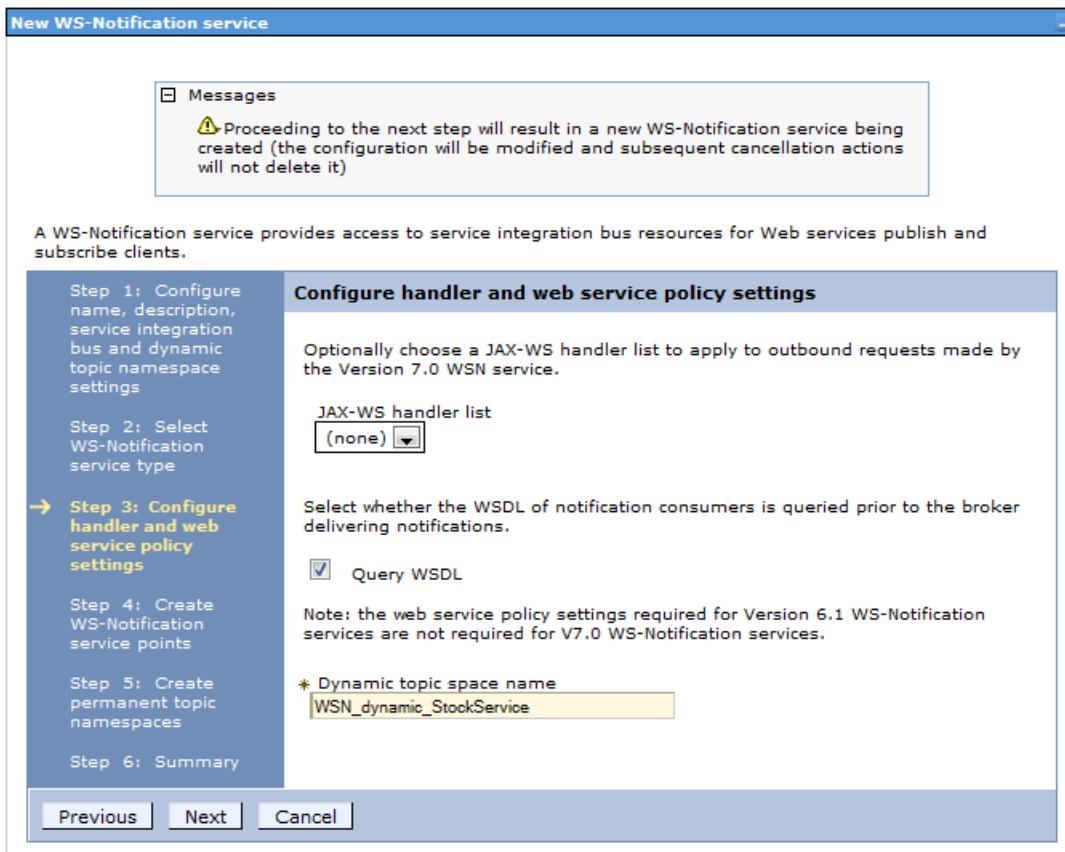
Additionally select this type of service if you wish to apply JAX-WS handlers to your new WS-Notification service.

This is the recommended type of service for new deployments.

**Version 6.1**

Select this type of service to expose the web service using the same technology provided in WebSphere Application Server version 6.1, including the ability to apply JAX-RPC handlers to the service.

**Previous** | **Next** | **Cancel**



New WS-Notification service point

Create a new WS-Notification service point.

**Configure name, description and select a bus member**

Supply a name and description and select a bus member for the WS-Notification service point. When creating a service point on a Version 7.0 WSN Service, only Websphere Application Server v7.0 and above bus members will be available for selection.

\* Name: StockServicePoint 

Description:

\* Select a bus member: win7-x64Node01:server1

Step 1: Configure name, description, service integration bus and dynamic topic namespace settings

Step 2: Select WS-Notification service type

Step 3: Configure handler and web service policy settings

Step 4: Create WS-Notification service points

→ Step 4.1: Configure name, description and select a bus member

Step 4.2: Define transport settings

Step 5: Create permanent topic namespaces

Step 6: Summary

Previous | Next | Cancel

New WS-Notification service point

Create a new WS-Notification service point.

**Define transport settings**

Select the transport settings for the new service point.

**SOAP over HTTP settings**

Service point accessed via HTTP proxy

\* HTTP proxy:

Select the SOAP version used by the service point: SOAP 1.1

**JAX-WS handler list settings**

NotificationBroker JAX-WS handler list: (none)

SubscriptionManager JAX-WS handler list: (none)

PublisherRegistrationManager JAX-WS handler list: (none)

Step 1: Configure name, description, service integration bus and dynamic topic namespace settings

Step 2: Select WS-Notification service type

Step 3: Configure handler and web service policy settings

Step 4: Create WS-Notification service points

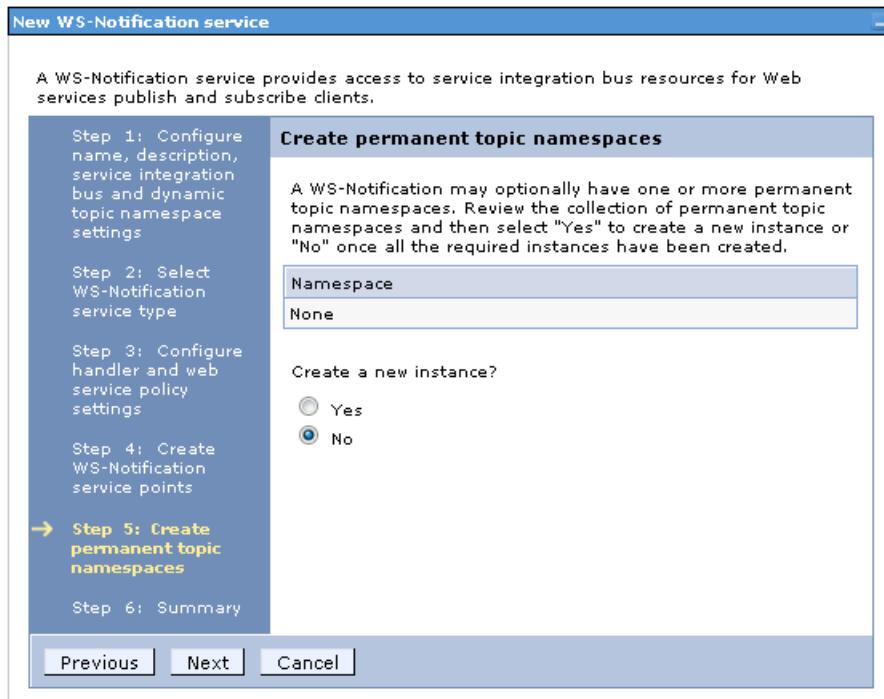
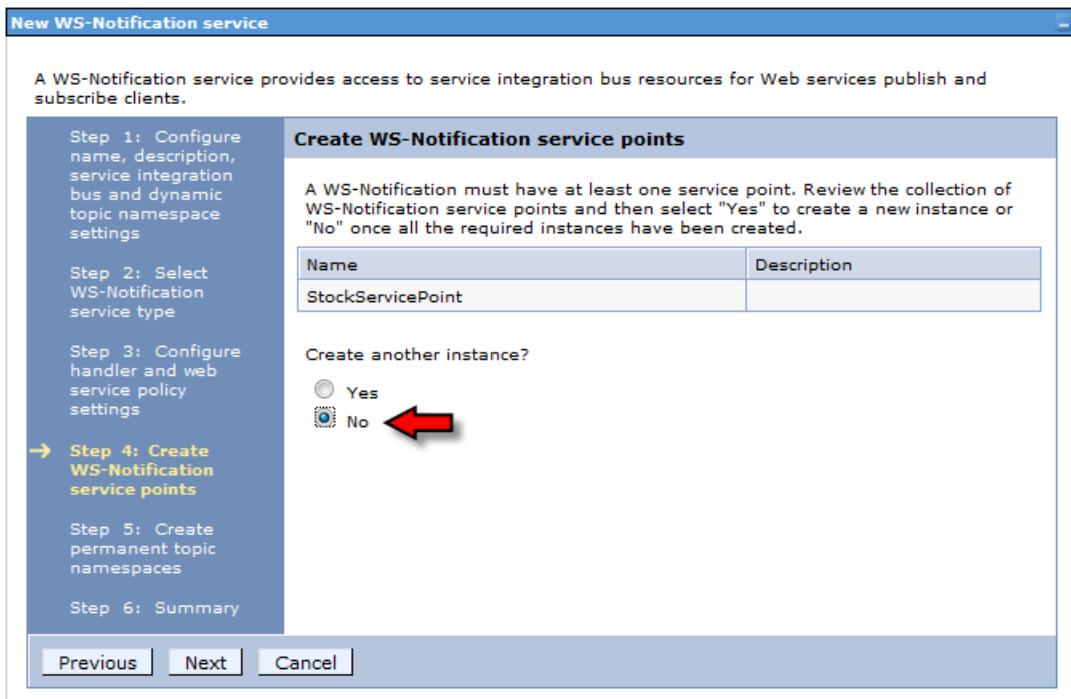
Step 4.1: Configure name, description and select a bus member

→ Step 4.2: Define transport settings

Step 5: Create permanent topic namespaces

Step 6: Summary

Previous | Next | Cancel



Once complete, the WS-Notification environment should be available. It is suggested to stop and restart the WAS server to be sure.

In order for applications to use the WS-Notification configuration just made, they must make Web Service calls to the server. To make a Web Service call, these applications will need WSDL that describes the target server and operations. The WSDL for these services can be dynamically retrieved from the new server configuration using the WAS admin console.

**WS-Notification services**

**WS-Notification services**

A WS-Notification service provides access to service integration bus resources for Web services publish and subscribe clients.

Preferences

| New | Delete |
|-----|--------|
|     |        |
|     |        |
|     |        |
|     |        |

| Select                   | Name              | Service integration bus name | Type | Description |
|--------------------------|-------------------|------------------------------|------|-------------|
| <input type="checkbox"/> | StockService      | StockBus                     | V7.0 |             |
| <input type="checkbox"/> | weatherWSNService | weatherWSNBus                | V7.0 |             |

Total 2

**WS-Notification services**

**WS-Notification services > StockService**

A WS-Notification service provides access to service integration bus resources for Web services publish and subscribe clients.

Configuration     Runtime

| General Properties           |                                                           | Additional Properties                          |
|------------------------------|-----------------------------------------------------------|------------------------------------------------|
| Name                         | StockService                                              |                                                |
| Service integration bus name | StockBus                                                  | <a href="#">WS-Notification service points</a> |
| Type                         | V7.0                                                      | <a href="#">Permanent topic namespaces</a>     |
| Description                  | <a href="#">Custom properties</a>                         |                                                |
|                              | <a href="#">Outbound request policy sets and bindings</a> |                                                |

**WS-Notification services**

**WS-Notification services > StockService > WS-Notification service points**

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified Web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

Preferences

| New | Delete |
|-----|--------|
|     |        |
|     |        |
|     |        |
|     |        |

| Select                   | Name              | Associated bus member  | Description |
|--------------------------|-------------------|------------------------|-------------|
| <input type="checkbox"/> | StockServicePoint | win7-x64Node01:server1 |             |

Total 1

**WS-Notification services**

[WS-Notification services](#) > [StockService](#) > [WS-Notification service points](#) > [StockServicePoint](#)

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified Web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

Configuration Runtime

**General Properties**

Name: StockServicePoint

Description:

Associated bus member: win7-x64Node01:server1

**Additional Properties**

- [Administered subscribers](#)
- [Custom properties](#)
- [Policy set configuration](#)
- [Publish WSDL files to zip](#)
- [Service point application](#)



**WS-Notification services**

[WS-Notification services](#) > [StockService](#) > [WS-Notification service points](#) > [StockServicePoint](#)

> [Publish WSDL files](#)

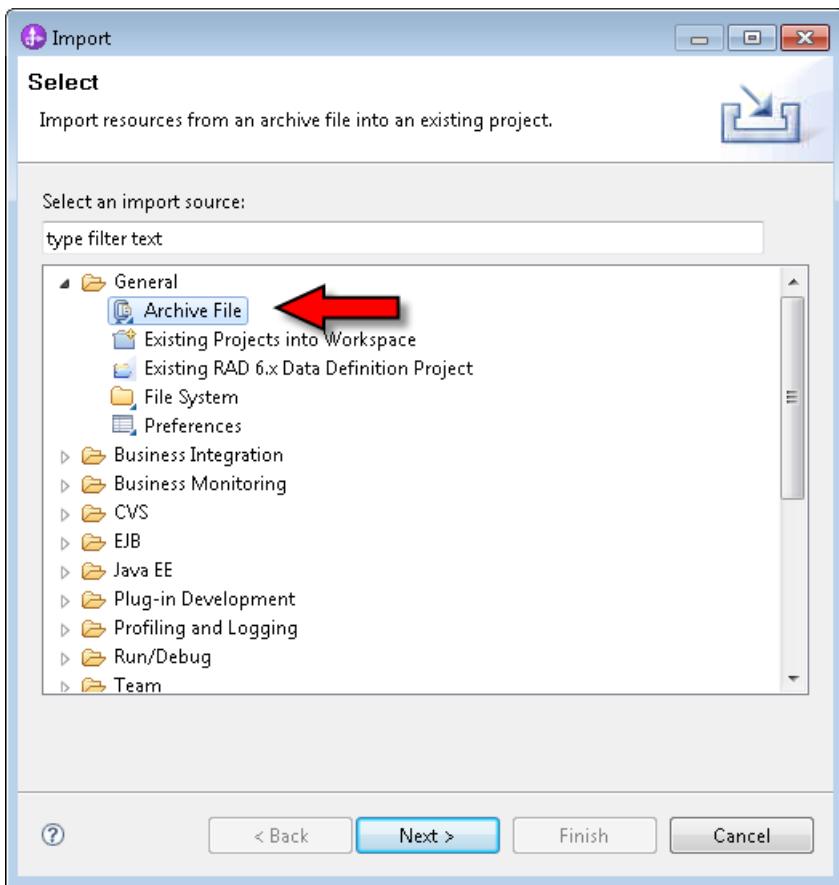
Click on the file name to download a zip file that contains the application's published WSDL files.

Publish WSDL files

[WSN\\_StockService\\_StockServicePoint.ear\\_WSDLFiles.zip](#)

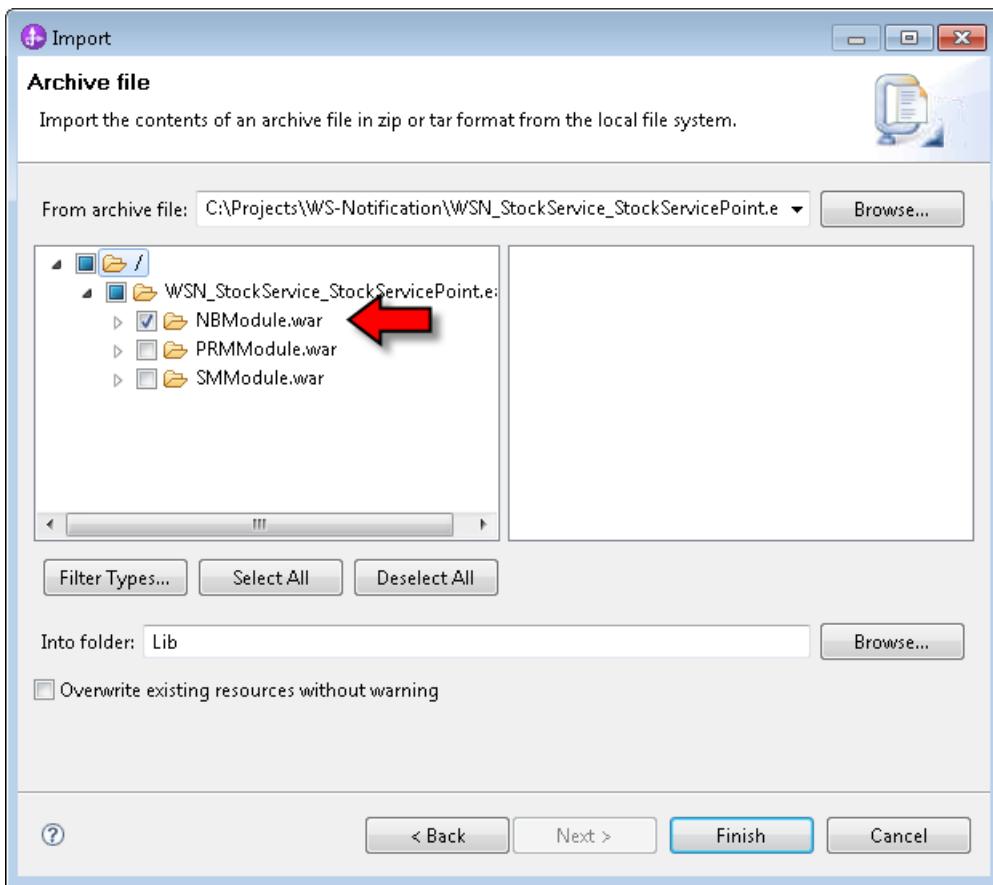
Back

Once the WSDL and XSD files have been exported, create a new Library project to hold the content. When the project has been created, import the ZIP file content that were exported from the WAS server.



Don't import all the content, just the NBModule. The NB acronym is Notification Broker.

**Warning!!:** When importing these artifacts, be sure to only import the files and not the folders. The folders contain the period character (".") which can **not** be used as folder names in WID as this conflicts with Java Packaging constructs.

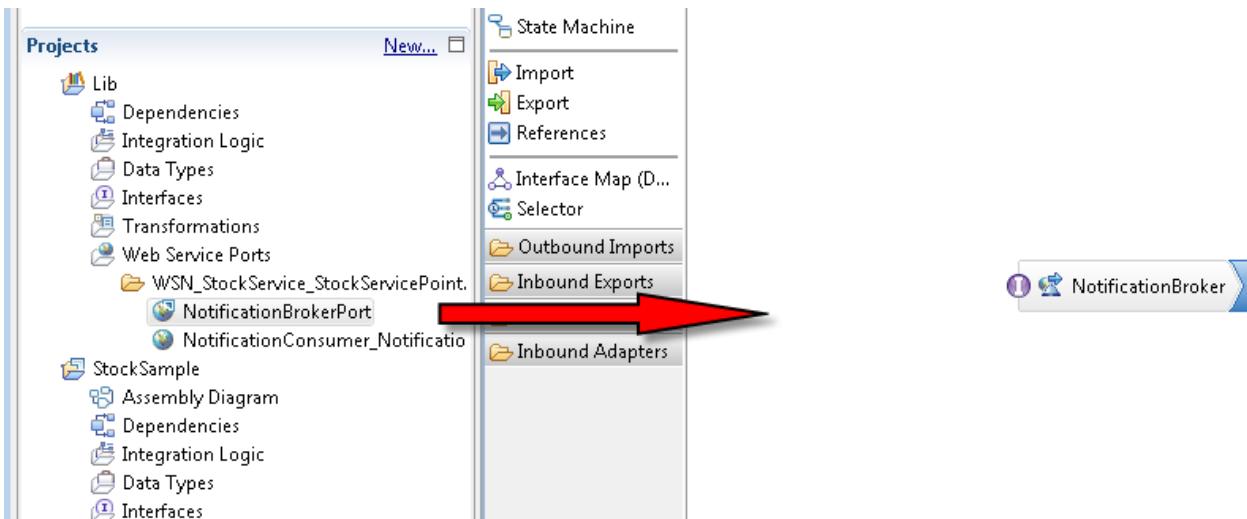


There are three services that we are going to be interested in. Two already exist and one we will build. The WS-Notification environment provides operations to register a subscription for a topic against a subscriber and also provides an operation to publish a notification. We will want to create a service that will act as a recipient for a subscription request.

We will start with that. We create a new module and associate the library we just built with that new module. In the module, we create an Export component and select the interface called `NotificationConsumer` as the interface of that export. We define a Web Service binding for that export. Finally we implement a trivial mediation flow for the export simply to show that it is called when we expect it to be called.



The next thing we wish to accomplish is to be able to register a subscription with the WS-Notification engine. WS-Notification provides us a service interface to be able to achieve exactly this. In the module, we add in the import for the WS-Notification broker.



### ***Implementation notes***

- As of 2010-09-03 it has been found that there are occasional problems with “cast” in the WESB XSLT mapping. Sometimes when a cast is requested in the WID editor, no list of possible data types is displayed. If the WID editor is stopped and restarted, the list of data types is once again shown.
- Payload data that is described as an XSD with a namespace appears to fail. A PMR has been raised. Currently use XSD data types that aren't associated with a namespace.
- When certain messages are sent to the broker in a multi-line format as opposed to a single line, an error is received in the console log. The error logged is:

```
[4/27/11 14:18:42:663 CDT] 0000005b WebServiceExc E
org.apache.axis2.jaxws.WebServiceExceptionLogger log A
javax.xml.ws.soap.SOAPFaultException throwable was caught. The detail message
is: actionMessageHolderType instance: unexpected Java class
com.ibm.ws.webservices.engine.xmlsoap.Text encountered for child object
at
com.ibm.ws.sib.wsn.webservices.utils.WSNMarshallerImpl.createSOAPFaultException(
WSNMarshallerImpl.java:748)
at
com.ibm.ws.sib.wsn.webservices.impl.inbound.provider.NBInvokerImpl.invoke(NBInvokerImpl.java:311)
at
com.ibm.ws.sib.wsn.webservices.impl.inbound.provider.NBProviderImpl.invokeTargetService(NBProviderImpl.java:142)
at
com.ibm.ws.sib.wsn.webservices.impl.inbound.provider.WSNProviderImpl.invoke(WSNProviderImpl.java:140)
at
com.ibm.ws.sib.wsn.webservices.impl.inbound.provider.NBProviderImpl.invoke(NBProviderImpl.java:108)
```

Changing the format to be single line seems to make the error disappear. A PMR has been raised to address this issue (2011-04-27 – 82736,7TD,000).

- IBM Suggests the following trace settings for debugging WS-Notification:

```
SIBWsn=all:com.ibm.ws.sib.webservices.*=all
```

## **Bugs and Fixes**

A problem was found where payload data with name spaces was not being handled correctly. A PMR was opened (00850,004,000) and a fix was provided. The fix requires the setting of a custom property. After applying the fix, a custom property need be set. Here is the doc on that:

To set this: Click Servers > Server Types > WebSphere application servers > server\_name, and then, in the Server Infrastructure section, click Java and process management > Process definition > Control > Java virtual machine > Custom properties Then define a new property and in the Name field, type the property name "sib.wsn.client.includeNSDecls". In the Value field, type the value "true". ( The default value is false ) .

## **References – WS-Notification**

- OASIS – [WS-Notification Technical Committee](#)
- [WS-Notification – Specification](#) - 2004-03-01
- DeveloperWorks - [WS-Notification in WebSphere Application Server V7: Part 2: Configuring JAX-WS applications with WS-Security for WS-Notification](#) – 2009-04-08
- DeveloperWorks - [WS-Notification in WebSphere Application Server V7: Part 1: Writing JAX-WS applications for WS-Notification](#) – 2008-11-12
- DeveloperWorks - [Leveraging Key WS-Notification Features in your Business Applications](#) - 2009-04-09
- DeveloperWorks - [How patterns shaped new WS-Notification functionality in IBM WebSphere Application Server 7.0](#) - 2008-12-23
- DeveloperWorks - [Optimal message processing with WS-Notification filters](#) – 2007-05-08
- DeveloperWorks - [Implement WS\\_Notification in WebSphere Application Server V6.1](#) - 2006-12-21
- DeveloperWorks - [WS-Notification in WebSphere Application Server Version 6.1](#) – 2006-11-15
- [WS Notification and WS Topics in the WS Resources Framework](#) – 2006-07-13
- IBM Education Assistant - [WS-Notification](#)
- RedBooks – [IBM WebSphere Application Server v7.0 Web Services Guide](#) – Chapter 9 - 2009-07-30

## **WS-Addressing**

WS-Addressing is the ability to describe Web Services endpoint information in a standard format. There are many uses for this capability.

Within an SCA Module we can define in the dependencies that the inclusion of WS-Addressing should be done.

▶ Version

---

▶ Libraries

---

▶ Java

---

▶ J2EE (Web, EJB, Connector, and so on)

---

▼ Predefined Resources

Select the resources to import into this module. [More...](#)

WS-Addressing 2005 schema file  
 WS-Addressing schema files (arrow pointing to this item)  
 WS-I attachment profile 1.0: swaRef schema file  
 WS-Security 1.0 schema files  
 WS-Security 1.1 schema files

Description:

▶ Ordering

---

▶ Unresolved Projects

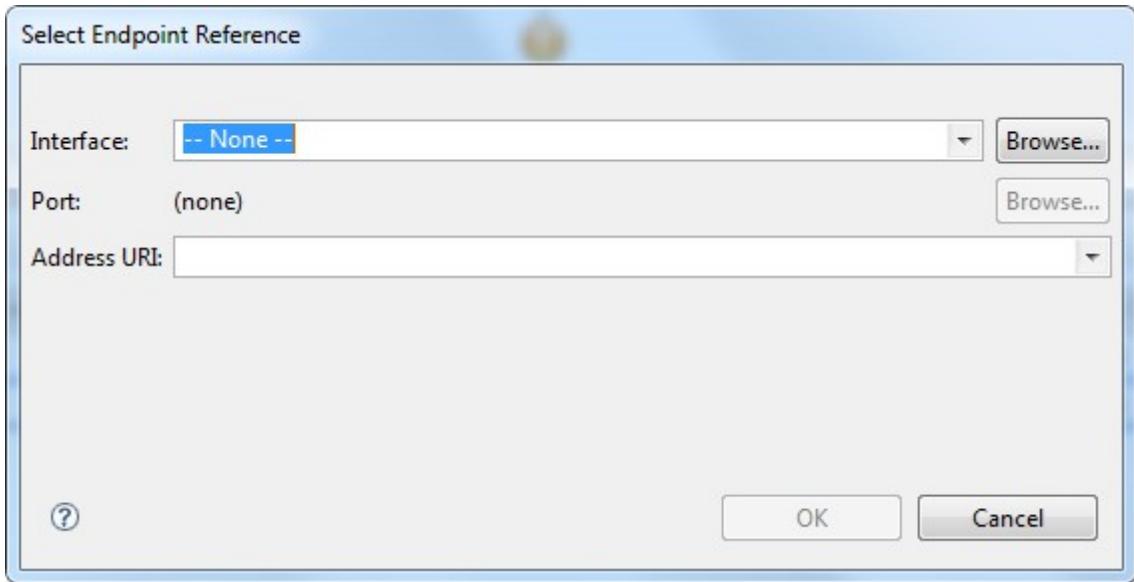
This adds a number of data types into the project.

- ▲ StandardImportFilesGen
  - AttributedQName
  - AttributedURI
  - EndpointReferenceType
  - FaultSubcodeValues
  - ReferenceParametersType
  - ReferencePropertiesType
  - Relationship
  - RelationshipTypeValues
  - ReplyAfterType
  - ServiceNameType
  - ServiceRefType

Within BPEL, we can perform assignments into a WS-Address by declaring a variable of type ServiceRefType. If we declare a variable of this type and then perform an assignment to it, we have two selection options. The first is selecting a Partner Reference. This will plug-in SCA bindings to the target.

- output1 : string
- srt1 : ServiceRefType
  - String (enter a value)
  - Number (enter a value)
  - True
  - False
  - Date
  - XPath Expression
  - Value Composer
  - Property of a Variable
  - ▲  Partner Reference (arrow pointing to this item)
    - I1
    - I2
    - I3
  - Endpoint Reference

The second option is an explicit Endpoint Reference. Using that technique, a wizard is displayed which prompts us for a variety of input parameters:



This will plug in the specified details.

If we look at a value of an example ServiceReferenceType we see that it has the following structure:

```
<p:ServiceRefType xsi:type="p:ServiceRefType"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ad="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:i3="http://Async1/I3"
 xmlns:p="http://schemas.xmlsoap.org/ws/2004/03/business-process/">
 <ad:EndpointReference>
 <ad:Address>http://localhost:9080/Async1Web/sca/I3Export</ad:Address>
 <ad:PortType>i3:I3</ad:PortType>
 <ad:ServiceName
 PortName="I3Export_I3HttpPort">i3:I3Export_I3HttpService</ad:ServiceName>
 </ad:EndpointReference>
</p:ServiceRefType>
```

Now let us give some thought to dynamic BPEL invokes. At a high level, we have the following notion. A BPEL process wishes to invoke a Partner service. Let us call this partner service "ABC". However, at development time, the endpoint address of "ABC" is not known. What we need to do is set the endpoint information dynamically at runtime. When we defined the BPEL process, we added a partner reference. Inside BPEL, this named partner reference is actually a variable of type ServiceReferenceType. What this means is that if we assign a ServiceReferenceType variable instance to the partner reference variable, the value of ServiceReferenceType will be honored. This specifically includes the addressing information and hence we have achieved dynamic invocation.

## WS-Security

Security associated with Web Service calls is based on the idea of Policy Sets. A policy set is a set of definitions of how security should be performed. Prior to policy sets, each service caller or provider would be explicitly coded or declared to have its own security settings. If a solution had many such services, each service would have to have the security settings replicated by hand which

was error prone and tedious. If a change was required to the security settings, all locations where the change needed to be applied had to be found and replaced. All in all, not a good situation. With the arrival of JAX-WS as a Java based Web Services engine/package, the notion of policy sets was introduced. Think of a policy set as a *container* of all the settings that one may wish to apply to a service. When a service is created, the named policy set can now be associated with that service. For additional web services, they can also be associated with the *same* named policy set. Because of this, the definitions for the security need only be created once and re-used when and where needed. A change to the policy set definitions results in an effective change to all the other services that referenced the policy set.

When an SCA Import is associated with a JAX-WS web service call, a named policy-set can then be associated with the SCA Import.

As such, the actual SCA import is not configured for security but instead a policy set is created and then the policy set associated with the import. If there are multiple SCA imports, each one can be re-use the same named policy set and thus provide loose coupling. If a security entry needs to be changed, changing it within the policy set will change it where ever that policy set is referenced.

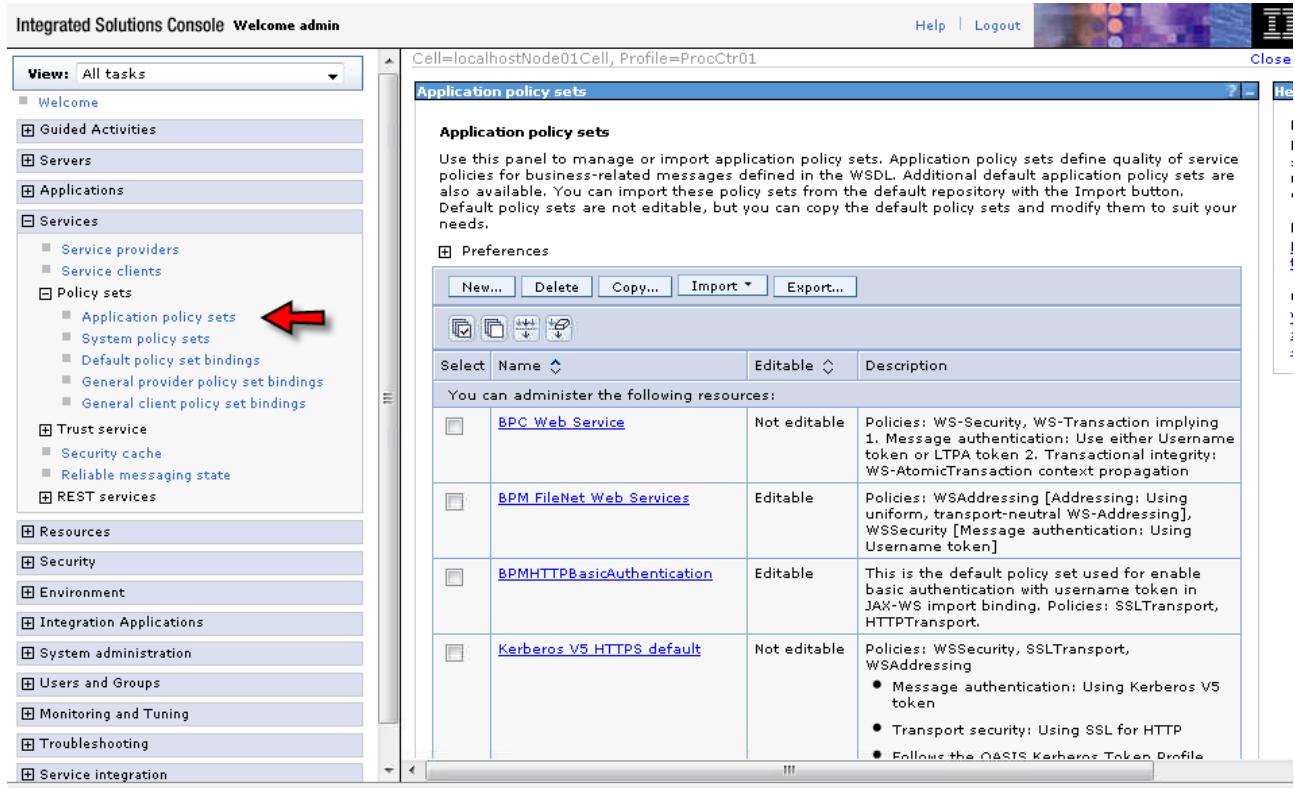
The same notion applied to SCA Export components:

To delve deeper into policy sets, we must set up some definitions.

- **Policy Type** – The Policy Type is a type of potential definitions for a type of policy. Think of it like a template of a group of setting that can be configured. Supplied Policy Types include:
  - WS-Security
  - WS-Addressing
  - WS-ReliableMessaging
  - HTTPS
  - WS-Transaction
- **Policy** – A policy is named instance of a Policy Type. It contains specific definitions for that Policy Type but explicitly does **not** include some specifics.

- Binding – A binding is associated with a Policy. The binding contains environment specific information. There are two types of binding:
  - Application-Specific binding
  - General bindings
- Policy Set – A collection of policies

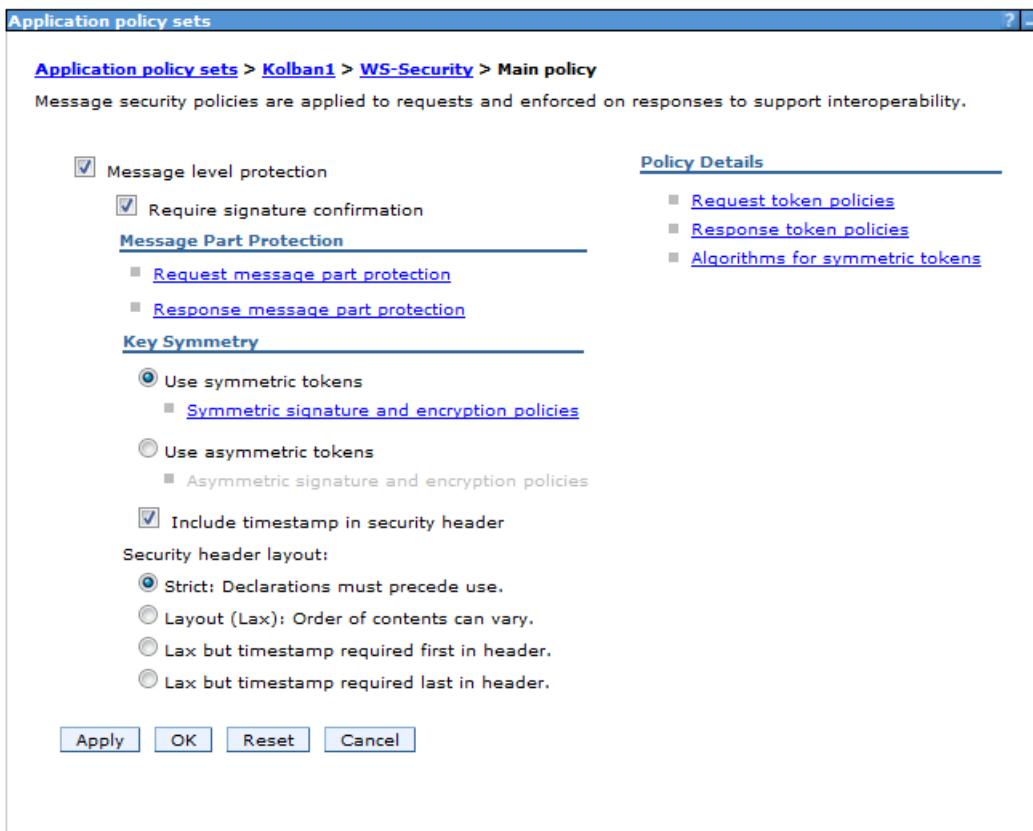
Policy set configurations are performed in the WAS admin console under Services > Policy sets:



The screenshot shows the Integrated Solutions Console interface. The left sidebar has a 'View: All tasks' dropdown and a tree view of various administrative categories. Under 'Services', the 'Policy sets' node is expanded, and its first child, 'Application policy sets', is highlighted with a red arrow. The main panel is titled 'Application policy sets' and contains a brief description of what it does. Below the description is a section titled 'Preferences' with buttons for 'New...', 'Delete', 'Copy...', 'Import...', and 'Export...'. A table lists four policy sets: 'BPC Web Service', 'BPM FileNet Web Services', 'BPMHTTPBasicAuthentication', and 'Kerberos V5 HTTPS default'. Each row in the table provides a link to the policy set details, its editability status (e.g., 'Not editable' or 'Editable'), and a detailed description of its policies.

| Select                   | Name                                       | Editable     | Description                                                                                                                                                                                                                                                |
|--------------------------|--------------------------------------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | <a href="#">BPC Web Service</a>            | Not editable | Policies: WS-Security, WS-Transaction implying 1. Message authentication: Use either Username token or LTPA token 2. Transactional integrity: WS-AtomicTransaction context propagation                                                                     |
| <input type="checkbox"/> | <a href="#">BPM FileNet Web Services</a>   | Editable     | Policies: WSAddressing [Addressing: Using uniform, transport-neutral WS-Addressing], WSSecurity [Message authentication: Using Username token]                                                                                                             |
| <input type="checkbox"/> | <a href="#">BPMHTTPBasicAuthentication</a> | Editable     | This is the default policy set used for enable basic authentication with username token in JAX-WS import binding. Policies: SSLTransport, HTTPTransport.                                                                                                   |
| <input type="checkbox"/> | <a href="#">Kerberos V5 HTTPS default</a>  | Not editable | Policies: WSSecurity, SSLTransport, WSAddressing <ul style="list-style-type: none"> <li>• Message authentication: Using Kerberos V5 token</li> <li>• Transport security: Using SSL for HTTP</li> <li>• Follows the OASIS Kerberos Token Profile</li> </ul> |

The security settings for a security policy look as follows:



In the Message Part Protection section we define how we want either the request or response messages protected. Protected in this case means two primary considerations. The first is encryption. If encryption is used then we define which parts of the message will be encrypted so that anyone examining the message in transit will not be able to decode all or part of the content. Note that this is not the same as SSL which encrypts at the transport level. Encryption in WS-Security encrypts parts of the body of the message.

The other part of message protection is called integrity. This is a technique to prevent tampering with a message once it has been sent but before it arrives. The integrity of a message is assured by digitally signing the request message and on receipt checking that the signature of the message was not changed. Think of this as performing a CRC or hash function over the content of the message and then including the resulting hash value as part of the message. If the message were changed, when the receiver performs the same hash function then the signature generated would be different than the one supplied with the message. The original signature is itself encoded with private keys so that a message can not simply have a new (but tampered with) signature added to the tampered message.

The parts of the message chosen for encryption and the parts for signing do not have to be the same.

**Application policy sets > Kolban1 > WS-Security > Main policy > Request message part protection**

Message part protection policies define parts to be protected and the nature of protection.

**Confidentiality protection**

Encrypted parts  
app\_encparts... ▾

Add Edit Delete

**Integrity protection**

Signed parts  
app\_signparts... ▾

Add Edit Delete

Done

## See also:

- RedBook - [Implementing Kerberos in a WebSphere Application Server Environment](#) - SG24-7771-00 - 2011-04-25
- RedBook – [IBM WebSphere Application Server v7.0 – Web Services Guide](#) – SG24-7758 – 2009-11-02 – (Chapter 6 – Policy Sets)
- RedBook - [Web Services Feature Pack for WebSphere Application Server V6.1](#) – SG24-7618
- DeveloperWorks - [Configuring WS-Security for JAX-RPC web services in WebSphere Process Server V7](#) - 2011-01-12
- DeveloperWorks - [Java web services: WS-Security without client certificates](#) – 2010-08-03
- DeveloperWorks - [WS-Policy security integration between DataPower and WebSphere Application Server](#) – 2009-11-18
- DeveloperWorks - [JAX-WS client APIs in the Web Services Feature Pack for WebSphere Application Server V6.1, Part 1: Creating a Dispatch client](#) – 2009-11
- DeveloperWorks - [JAX-WS client APIs in the Web Services Feature Pack for WebSphere Application Server V6.1, Part 2: Creating a proxy client](#) - 2007-09-19
- DeveloperWorks - [Message-level security with JAX-WS on WebSphere Application Server V7, Part 1: Using Rational Application Developer V7.5.2 to build secure JAX-WS Web services](#) – 2009-05-13
- DeveloperWorks - [Message-level security with JAX-WS on WebSphere Application Server V7, Part 2: Integrating JEE authorization](#) - 2010-01-27
- DeveloperWorks - [Message-level security with JAX-WS on WebSphere Application Server V7, Part 3: Programmatic client control using Web Services Security APIs](#) - 2010-08-31
- DeveloperWorks - [Using WSS4J/Axis2 API, Part 1: Sending WS-Security Signature and Encryption Profiles to Axis2/Rampart Web service](#) - 2009-11-24
- DeveloperWorks - [Java Web services: The high cost of \(WS-\)Security](#) – 2009-07-07
- DeveloperWorks - [Achieving Web services interoperability between the WebSphere Web Services Feature Pack and Windows Communication Foundation, Part 2: Configure and test WS-Security](#) – 2007-04-03
- DeveloperWorks - [Web Services Trust Language](#) - 2007-10-04
- DeveloperWorks - [Tackle WS-Security specification interoperability challenges, Part 4: Add a J2EE 1.3 provider endpoint to a J2EE 1.4 Web service](#) - 2007-09-13
- DeveloperWorks - [Tackle WS-Security specification interoperability challenges, Part 3: Using the EJB proxy](#) – 2007-09-04
- DeveloperWorks - [Tackle WS-Security specification interoperability challenges, Part 2: Using the WebSphere Web Services Gateway feature](#) - 2007-07-26
- DeveloperWorks - [Tackle WS-Security specification interoperability challenges, Part 1: Problem overview and four available workarounds](#) - 2007-05-03
- DeveloperWorks - [Understanding web services specifications, Part 4: WS-Security](#) – 2006-08-22
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6](#) -- Part 1 – 2006-04-19
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6](#) -- Part 2 – 2006-04-19
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6](#) -- Part 3 – 2006-05-30

- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6](#) -- Part 4 – 2006-07-26
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6](#) -- Part 5 - 2006-12-06
- DeveloperWorks – [Updated: Web Services Security Policy Language](#) – 2005-07-13
- DeveloperWorks - [Web Services Secure Conversation Language](#) - 2005-02-01
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Using Web Services Security in WebSphere Application Server](#) - 2004-04-14
- DeveloperWorks - [Web Services Security](#) - 2004-03-01
- DeveloperWorks – [Implementing WS-Security](#) – 2004-04-01
- DeveloperWorks - [Best Practices for web services, Part 11: Web services security, Part 1](#) – 2004-03-26
- DeveloperWorks - [Best Practices for web services, Part 12: Web services security, Part 2](#) - 2004-03-30
- DeveloperWorks - [Web services security, Part 1](#) - 2003-02-25
- DeveloperWorks - [Web Services Security: Moving up the stack](#) - 2002-12-01
- DeveloperWorks - [WS-Security Profile for XML-based Tokens](#) – 2002-08-28
- DeveloperWorks - [Security in a web services world: A proposed architecture and roadmap](#) - 2002-04-01

## **Building a secure Web Service**

When we wish to set up secure Web Service communications, we start by choosing a Policy Set and associated bindings. These will then be attached to the Web Services. When creating a new Policy Set, it is **always** recommended to start from one of the IBM supplied Policy Sets and copy that for use.

To view the existing Policy Sets, we can use the WAS admin console from Services → Policy Sets:

Cell=win7-x64Node01Cell, Profile=ProcCtr01

**Application policy sets**

Use this panel to manage or import application policy sets. Application policies for business-related messages defined in the WSDL. Additional details are available. You can import these policy sets from the default repository. Default policy sets are not editable, but you can copy the default policy set needs.

**Preferences**

| New...                                      | Delete                                                  | Copy...      | Import                                                                                                      | Export... |
|---------------------------------------------|---------------------------------------------------------|--------------|-------------------------------------------------------------------------------------------------------------|-----------|
|                                             |                                                         |              |                                                                                                             |           |
| Select                                      | Name                                                    | Editable     | Description                                                                                                 |           |
| You can administer the following resources: |                                                         |              |                                                                                                             |           |
| <input type="checkbox"/>                    | <a href="#">BPC Web Service</a>                         | Not editable | Policies enabled: 1 Policies: WS-Security, Message authentication token or LTPA token, WS-AtomicTransaction |           |
| <input type="checkbox"/>                    | <a href="#">BPM FileNet Web Services</a>                | Editable     | Policies enabled: 1 Policies: WSAddressing, uniform, transport, WSSecurity [Message, Username token]        |           |
| <input type="checkbox"/>                    | <a href="#">BPM SSO Policy Set</a>                      | Not editable | Policies enabled: 1 Policies enabled: 1                                                                     |           |
| <input type="checkbox"/>                    | <a href="#">BPM UNT Policy Set</a>                      | Not editable | Policies enabled: 1 Policies enabled: 1                                                                     |           |
| <input type="checkbox"/>                    | <a href="#">BPM Web Service API Callback Policy Set</a> | Editable     | Policies enabled: 1 WS-ReliableMessage                                                                      |           |
| <input type="checkbox"/>                    | <a href="#">BPM Web Service API Policy Set</a>          | Editable     | Policies enabled: 1 WS-ReliableMessage                                                                      |           |
| <input type="checkbox"/>                    | <a href="#">BPMHTTPBasicAuthentication</a>              | Editable     | Policies enabled: 1 This is the default basic authentication WS import binding HTTPTransport.               |           |
| <input type="checkbox"/>                    | <a href="#">Kerberos V5 HTTPS default</a>               | Not editable | Policies enabled: 1                                                                                         |           |

## Testing Web Services security with TCP/Mon

It seems like a sensible idea to test with TCP/Mon to visualize the Web Services traffic flowing between a service client and a service provider. In my testing I then placed TCP/Mon between the two but started to get some new errors. Specifically, the following fault was returned:

```

<soapenv:Body xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Fault>
 <faultcode>soapenv:FailedAuthentication</faultcode>
 <faultstring>security.wssecurity.WSSContextImpl.s02:
com.ibm.websphere.security.WSSecurityException: Exception
org.apache.axis2.AxisFault: CWWSS6521E: The Login failed because of an
exception: javax.security.auth.login.LoginException: CWWSS7218E: SCT is not
valid for the webservice endpoint
"http://localhost:9082/ProviderWeb/sca/ProviderI2". SCT is issued for
"http://localhost:9999/ProviderWeb/sca/ProviderI2". occurred while running
action:
com.ibm.ws.wssecurity.handler.WSSecurityConsumerHandler$1@21cb21cb</faultstring>
 </soapenv:Fault>
</soapenv:Body>
```

In my testing, my client was running on port 9081, my server on port 9082 with TCP/Mon proxying port 9999 to 9082. The client sent the request to 9999. At a guess, I'd say that the signature was built for a target of 9999 (where the client thought it was sending the request) while when it arrived at 9082 (the real target), the signature no longer matched. This seems to mean that we can't place

TCP/Mon in the path of signatures.

## JAX-WS

JAX-WS is the Java based way of either providing or invoking Web Services.

### JAX-WS Java SE stand alone Service Provider

With JAX-WS it is quite straightforward to build a Web Service provider that is nothing more than a stand-alone Java SE application. Here is a quick recipe.

Build a Java application that looks as follows:

```
package kolban;
import javax.jws.WebService;
import javax.xml.ws.Endpoint;

@WebService
public class CircleFunctions {

 public static void main(String[] args) {

 System.out.println("About to publish!");
 Endpoint.publish(
 "http://localhost:9998/WebServiceExample/circlefunctions",
 new CircleFunctions());
 System.out.println("Ending");

 }

 public double getArea(double r) {
 return java.lang.Math.PI * (r * r);
 }

 public double getCircumference(double r) {
 return 2 * java.lang.Math.PI * r;
 }
}
```

The key things to note are that it is annotated as being a Web Service.

Parameters for exposed methods should also be annotated with the "@WebParam" tags. This allows parameter names to be exposed to the callers. For example:

```
public void chargeCard(
 @WebParam(name="billingDetails") BillingDetails_srv billingDetails,
 @WebParam(name="chargeAmount") float chargeAmount)
```

See also:

- [Introducing JAX-WS 2.0 With the Java SE 6 Platform, Part 1](#) - 2006-09

# Business Space

Business Space is a web based end user interface portal for accessing applications hosted on IBPM Advanced. It is accessed at the URL:

<http://localhost:9060/BusinessSpace>

The screenshot shows the 'Getting Started' page of the Business Space portal. At the top, there's a navigation bar with links for 'Home', 'Go to Spaces', 'Manage Spaces', and 'Actions'. On the right, it shows 'admin | Help | Log out'. Below the navigation is a header 'Welcome to Business Space' with a house icon. A large banner features a woman's face and hands over a background of colored sticky notes. The banner text says 'Welcome to Business Space...' and 'where you can model, simulate, execute, rapidly change, monitor and optimize your core business processes in a collaborative environment. [More...](#)'. To the left, a blue button says 'Learn >>' and 'Read the documentation'. To the right, a green button says 'Start >>' and 'Open the Space Manager to work with business spaces'. On the far right, there's a sidebar with a 'See the features of Business Space' link and a 'Watch' button with a play icon. Below the sidebar, there's a list of links: 'Tour Business Space', 'How to open a space', 'How to create a space', 'How to add widgets', and 'How to get help'.

The model of Business Space is that it contains *Spaces* where a space is a collection of pages and each page can host one or more sections ("called widgets").

The widgets supplied by IBM are:

- Available Tasks
- Business Calendars
- Business Categories
- Business Category Information
- Business Rules
- Create Tasks
- Document
- Escalations
- Google Gadgets
- Human Workflow Diagram
- Mediation Policy Administration
- Module Administration

- Module Assembly
- Module Browser
- Module Health
- Module Properties
- My Tasks
- My Team's Tasks
- My Work Organizer
- Page Navigator
- Presentation
- Process Definitions
- Processes
- Proxy Gateway
- Script Adapter
- Security Roles
- Service Browser
- Service Monitor
- Spreadsheet
- Store and Forward
- Store and Forward Details
- System Health
- Task Definitions
- Task Information
- Tasks
- Tasks I created
- Team List
- Web Feed
- Web Site
- Work Basket Information
- Work Baskets
-

# Administering SCA Modules

SCA Modules are normally deployed to Process Servers as part of a Process Application. However, these can be manually administered through the Admin Console.

Within the Admin Console we will find Applications > SCA modules.

The screenshot shows the Integrated Solutions Console Admin Console interface. On the left, there is a navigation sidebar with a red arrow pointing to the 'SCA modules' link under the 'Applications' section. The main content area is titled 'SCA modules' and contains a table of installed SCA modules. The table has columns for Select, Module, Version, Application, and Status. Each row lists a module name, its version, the application it belongs to, and a status indicator. A total of 10 modules are listed.

| Select                   | Module                                                               | Version | Application                                                        | Status |
|--------------------------|----------------------------------------------------------------------|---------|--------------------------------------------------------------------|--------|
| <input type="checkbox"/> | <a href="#">App01_Implementation</a>                                 | SN2     | <a href="#">APP01-SN2-App01_ImplementationApp</a>                  |        |
| <input type="checkbox"/> | <a href="#">AssemblyService</a>                                      |         | <a href="#">AssemblyServiceApp</a>                                 |        |
| <input type="checkbox"/> | <a href="#">BFMIF_win7-x64Node01_server1</a>                         |         | <a href="#">BPEContainer_win7-x64Node01_server1</a>                |        |
| <input type="checkbox"/> | <a href="#">Caller</a>                                               |         | <a href="#">CallerApp</a>                                          |        |
| <input type="checkbox"/> | <a href="#">ComputerMonitorVendorService</a>                         |         | <a href="#">ComputerMonitorVendorServiceApp</a>                    |        |
| <input type="checkbox"/> | <a href="#">FactoryFloor</a>                                         |         | <a href="#">FactoryFloorApp</a>                                    |        |
| <input type="checkbox"/> | <a href="#">HTMIF_win7-x64Node01_server1</a>                         |         | <a href="#">TaskContainer_win7-x64Node01_server1</a>               |        |
| <input type="checkbox"/> | <a href="#">HTM_PredefinedTaskMsg_V7500_(win7-x64Node01_server1)</a> |         | <a href="#">HTM_PredefinedTaskMsg_V7500_win7-x64Node01_server1</a> |        |
| <input type="checkbox"/> | <a href="#">HTM_PredefinedTasks_V7500_(win7-x64Node01_server1)</a>   |         | <a href="#">HTM_PredefinedTasks_V7500_win7-x64Node01_server1</a>   |        |
| <input type="checkbox"/> | <a href="#">Messaging_Implementation</a>                             |         | <a href="#">Messaging_ImplementationApp</a>                        |        |

From there we can perform a variety of tasks including un-installation.

# Exercises and Case Studies

As the book comes to a close, we will explore some exercises and case studies. The exercises are sample business processes that are illustrative and you are encouraged to try and build them out yourself or simply follow along with the possible solution described. The case studies are the essence of real-world solutions seen in the wild. These processes have been sufficiently changed so that no process secrets are exposed.

## ***Exercise – Education class enrollment***

You have undoubtedly attended vendor education classes in the past. Think about what has to go on behind the scenes to host such a class. We will focus on a part of this process called student enrollment. We can imagine that a class on a particular topic is going to be taught at a location on a certain date. IBM education is going to teach the class and believes that there is sufficient demand in the topic in that particular city. An instance of a class, in a city and on a particular date is called a *class offering*. A few months before the class is to be delivered, the enrollment for the class begins. Enrollment closes three days before the delivery date. Anytime between the class being originally offered and two weeks prior to the delivery date, the class may be cancelled by the class administrators.

Challenge – Design and implement the enrollment process

When we think about any new process that has been handed to us for design, it is good to walk through the following generic questions:

- What is the purpose of the process?
- When does the process start?
- What is the input to the process?
- What are the possible completion states of the process?
- What is the output of the process?
- Who participates in the process?

Q: What is the purpose of the process?

This feels like one of those awkward questions but some reflection should show that we can't begin to design a process if we can't articulate an answer to this question. We don't have to describe *how* the process will achieve this task. After all, that is why we are going to design it because we don't know how it will work but we must be able to describe what its purpose is. For this scenario our process is to manage the enrollment of students for a class offering. And we should stop there. If we attempt to say more, we are probably in design.

Q: When does the process start?

Our process starts after the decision to offer a class has been made. How that particular decision is arrived at is not in scope and we can ignore it completely. We will assume that the class has been advertised (somehow) and that enrollment should begin.

Q: What is the input to the process?

Most if not all processes have some initial input data supplied to them when they start. It is this

input data that differentiates one process instance from another. Those processes which *don't* have input data quickly solicit some as part of the process to allow them to distinguish themselves from other processes. In our story, the input to this process is the details of a class offering. Here we could go off and perform interview after interview to meticulously pin down all the details or we could go to the other extreme and treat our input as a black box simply called a "class offering" and do nothing more than that. As always, the correct answer is some happy medium. I suggest making notes of the low hanging fruit. If we say that the process starts after a class is offered ... then ask about the attributes of an offered class. For our story, we will assume that a class offering is made up of the following information.

- Name of the class/Subject being taught
- Location of the class
- Date on which the class will start
- Duration of the class
- Cost to attend the class

Are all of these attributes of a class pertinent to our process? Probably not. Will there be other attributes that we will need to work with that haven't yet been captured? Almost certainly.

Determining the inputs to the process does not have to be a one-shot event. If it is later determined that in order to initiate a new process instance more information than we thought we needed is required then we can either negotiate to have those items supplied to the process when it starts or else we can include steps in our process to explicitly source that information from within the process itself.

Q: What are the possible completion states of the process?

Every process instance must eventually come to a conclusion. Determining when the process has finished is a key consideration and one that if captured will guide the design. Reviewing our process notion, we remember that it is the enrollment of students into a proposed class. Enrollment ends three days before the class is to be taught or if the class is cancelled. In this last sentence, we have the core of the process termination states. There are two of them. One is that the enrollment process ends because the class is ready to be taught and the other is that the enrollment process ends because the class has been cancelled.

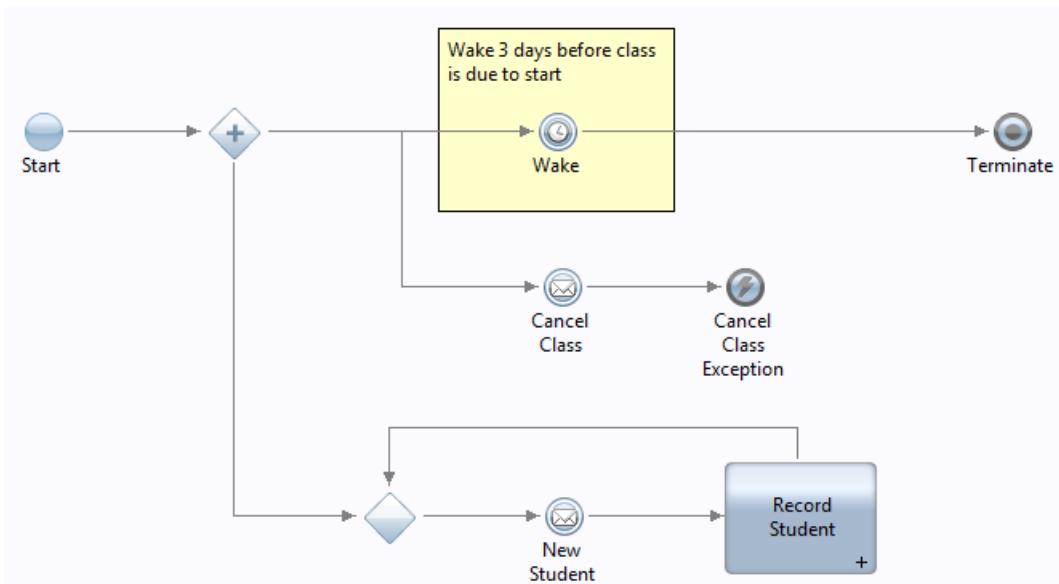
Q: What is the output of the process?

Unlike inputs, not all processes have distinct outputs from an IT data perspective. If the process we are following is that of making a cup of tea, there is no "data" at the end but is instead a sequence of steps that a human being follows. In our process however, we do have new output data. If we think about the purpose of the process, we will find that the output is a list of students enrolled for the class.

Q: Who participates in the process?

Most IBPM based processes have an element of staff interaction where human beings perform some role in the process. Identifying the roles of these people and how they interact can be key. The participants usually come to light during the interviews.

My first attempt at a solution was to use IBPM as a BPMN diagramming tool and I came up with the following:

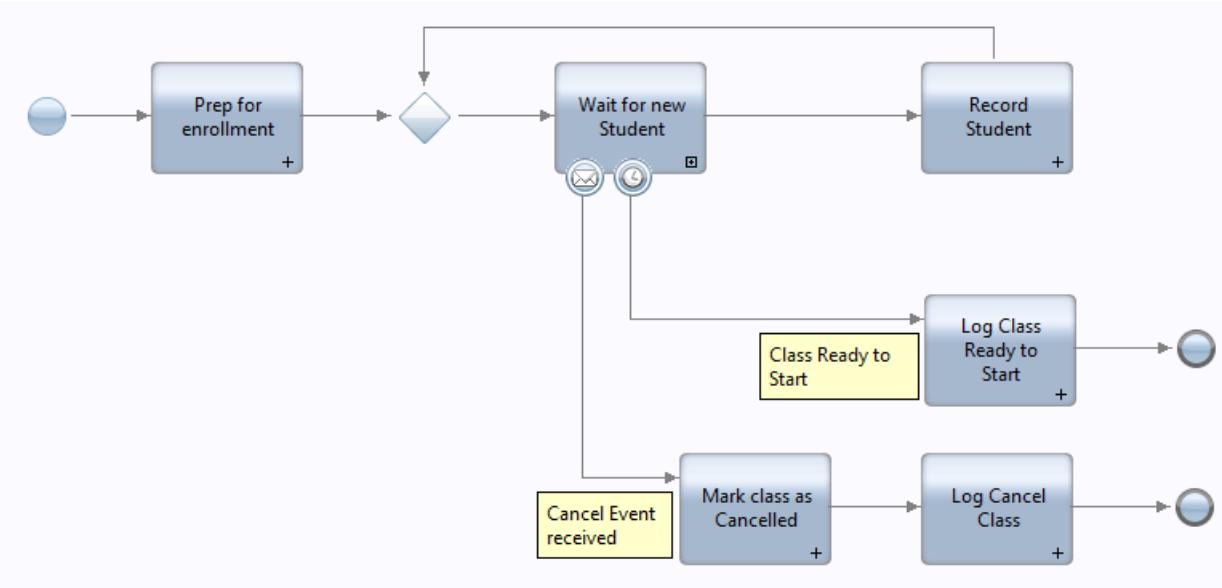


Let us walk through the story. It begins with a parallel gateway that splits the process into three parallel tokens. The first (top) reaches a timer that puts that thread to sleep until 3 days before the class is to start. The second (thread) waits for a Message Event that represents a request to cancel the class. The third (thread) waits for a Message Event that represents a request to enroll a new student. Notice that the new student event, once processed, returns back to waiting for another new student event. This appears to be an unbreakable loop which might cause the process to never end. Notice however that either a cancel event or a wake timer event will also occur. The result of these is an exception or a terminate event either of which will cause the process to terminate in its entirety. This process now looks pretty elegant ... unfortunately, we have some problems.

BPMN semantics state that when a terminate event is reached, the process is immediately terminated. This is what we expected. The BPMN semantics then go on to state that if the process that is terminated is a child process, the parent process regains control and continues. This too is goodness since we expect this process to be invoked as part of a larger process. Unfortunately, IBPM doesn't adhere to the BPMN semantics when it comes to the terminate event. If a terminate event is reached in IBPM, both the process **and** any parent chain that caused it to be reached are terminated and this is not what we want.

There is another flaw with this process as it stands. The cancel exception throws an exception (as per its design intent) but it does **not** pass any data (other than the nature of the exception) back to its caller. What this means is that the knowledge of which students are already enrolled will not be returned back to the caller in the event of a class cancellation and we said earlier that the outcome of student enrollment will be the students *currently* enrolled ... this should be true whether or not the enrollment process ended because the class was cancelled or whether the class is ready to be taught.

Now we approach our second level of refinement and after some thought, come up with the following:



Superficially, this process looks substantially different from our last attempt but actually follows the same principles. The core of the story is the activity called "Wait for new Student" which we have implemented as a sub-process. The process steps that it encapsulates looks as:



Ok ... this is obviously going to need some explanation. The activity called "Wait for new Student" will block until a new student event arrives. If one of these events arrives, we transition to "Record Student" and then go back to waiting for a new enrollment. This we can understand easily enough. Now notice the *attached* event handlers that are attached to the activity. There is one for a message event and one for a timer event. The message event will wake up when a cancel class event is received and the timer event will wake up when it is 3 days before the class. Either of these events causes the termination of the *activity* to which they are attached. Since this activity is the sub-process which is waiting for a new student, we have in effect found a way to break out of our loop.

Notice also that if the class is cancelled, we no longer throw an exception. Instead we now return simple control to the caller but in the data returned, we flag that the class has been cancelled and assume that the caller will handle that information appropriately.

Is this the only possible solution? Probably not ... and that is where the artistry of BPM comes into play. We have married together here a number of competing factors. We have attempted to capture the functional intent of the process, we have attempted to make the process as readable and easily understood as possible while and finally we have designed the process so that it can be actually executed on the IBPM runtime. Ideally all three of these goals can be satisfied but as shown, sometimes we have to pragmatically trade one for the other.

If we had used our first attempt at the process, we would have discovered the termination of all levels of the process and have to rethink our design (which we actually did). Only experience and diligent reading about (and experimentation with) the product will allow you to head-off such situations. The cancellation situation would ideally have been caught early enough to negotiate changes at the process level as a whole.

Now, back to the story ... as student enrollments arrives, where are these student enrollments being recorded? Ideally they are being recorded in a data store such as a database. Storing them within

the process itself is normally not a good idea. In general, unless it is control data, process data should be stored where other data normally lives and that is in a database or other external application.

Having created a BPD and drawn it out in AE unfortunately the process is not yet ready to run. Generally, less than 50% of the work has been accomplished in drawing the diagram till it is ready to be executed. As they say, the devil is in the detail and each element on the BPD canvas can have a lot of detail associated with it. A question of when the detail should be added arises. Should each activity be fleshed out as it is added to the diagram or should the diagram as a whole be drawn and then fleshed out with detail? There is nothing in the tool that forces you to work one way or another and each have strengths and weaknesses. Personally, I like to draw out the whole diagram and then go back and supply more details. One down side to this approach is to remember to actually add the details after creating the diagram. Too many times I have created a diagram and added the details I remembered to add only to find unexpected results during testing which were caused by other omitted details. One solution to this is to change the color of the activities *after* they have been implemented leaving their default colors to mean not implemented. Unfortunately not all activities can be colored. If adding the details later has this flaw why not add details as we go along? This itself can result in a different problem. It is very common to start building out a process only to make dramatic changes as we go along. If we fleshed out every step as we progress, we would be wasting effort discarding or dramatically changing much before the final process.

In our story, we have two event types. One for the cancellation of a process and one for the arrival of a new student registration. These types of events are used to indicate to the process something happening outside of the bounds of the process and for both of these, events are good models. We don't know if or when a class will be cancelled and we don't know when students will register or how many. Associated with an event is event payload data. This is information that travels with the event. For a student enrolling, we will assume this to be the details of the student. In our first pass at modeling, we don't have to pin down all the details of this data other than to know it exists. In IBPM, events are associated with UCAs. Because there are two types of events, it makes sense to define two UCAs.

## **Looking for home ....**

The following are sections that were written because I was working on that area at the time but haven't yet been found a proper *home* in the document yet.

# Human Task Manager

See Also – Human Task Manager:

- DeveloperWorks - [Using advanced human task patterns in WebSphere Process Server V7](#) - 2010-08-04

The Human Task Manager can be accessed through an API.

## Work Baskets

With the release of Feature Pack 1 for WPS 7.0, we saw the arrival of a new capability called Work Baskets. To understand Work Baskets, first let us review the nature of a standard human task. When a Human Task definition is created, a set of IBM defined roles for that task exist. For each role, the developer can define which users are to be associated with those roles. Let us consider the most commonly used role which is called Potential Owner.

Imagine we have a task definition called TaskDefinition1. When that task definition is created, we may assign Alice and Bob as potential owners. What this means is that when an instance of the task is created from the definition, at runtime, either Alice or Bob can claim and work upon the task. Charlie, who is not named in the task definition can not work on the task.

This is the basic model of Human Task authorization in WPS before the introduction of Work Baskets. Even though Work Baskets now exist, the original model still holds true. Work Baskets is an additional model that can be used if desired or, conversely, be completely ignored.

Let's use an analogy to understand Human Task Work Baskets. Imagine fictitious workers in an office. Each worker sits at their desk to perform their job. When they are ready for some work to do they stand up from their chair and walk to a communal area where there are "work baskets".



Each work basket has a "name" associated with it. This name describes the type of work that can be found in the folders in that basket. Examples of work basket names at an office may be:

- Bills to be paid – Bills that the company has to pay
- Customer complaints – Concerns from customers that have to be addressed
- Sales leads – Potential new customers that the company should follow up upon

Depending on the job role of a worker, the worker can take work from the appropriate basket. To make the story more interesting, imagine that there is a security guard beside the work baskets. In order for a worker to see work in a basket, he must tell the guard who he is. The guard then checks the employee's name against a list of people who are authorized to take work from the basket. Beside each basket there is a clipboard with names on it. If the worker's name is not in the list, he doesn't get to work with the folders.

## Work Basket Roles

- Reader – Role members can examine the definition of the work basket
- Opener – Role members can see the content of the work in the work basket
- Distributor – ??
- Transfer initiator – Role members can transfer work **out** of the work basket
- Appender – Role members can add tasks to the work basket
- Task reader – Members of this role can see the properties of a task but can't claim or complete it
- Task editor – Role members can work on the content of the task but can't claim it or complete it
- Task potential owner – Role members can claim a task
- Task administrator – Role members can administer a task

## Work Basket Business Space widgets

Work Baskets are created from the Work Baskets list widget. The Actions pull-down has a Create option to create a new Work Basket.

|                          | Identifier | Name | Type       | Owner | Description |
|--------------------------|------------|------|------------|-------|-------------|
| <input type="checkbox"/> | WB1        | WB1  | Individual | admin |             |
| <input type="checkbox"/> | WB2        | WB2  | Individual | admin |             |

Actions ▾

- Create**
- Create copy
- Edit
- Remove as target
- Delete

1 - 2    2

When selected, the companion Business Space Widget called Work Basket Information will be updated (via Widget to Widget events) to allow the user to define the settings for the new basket.

**Work Basket Information**

Save Cancel New work basket

General | Access | Distribution targets

|              |                                                             |         |                      |
|--------------|-------------------------------------------------------------|---------|----------------------|
| Identifier * | <input type="text"/>                                        | Name *  | <input type="text"/> |
| Description  | <input type="text"/>                                        |         |                      |
| Type *       | Individual                                                  | Owner * | <input type="text"/> |
| Task list    | <input type="text"/> <input type="button" value="Look up"/> |         |                      |

The settings of a Work Basket are broken down into three categories:

#### General

- Identifier – An identifier for the work basket
- Name – A name for the work basket
- Description – A description of the work basket
- Type
  - Individual
  - Group
  - Clearance
  - Topic
- Owner – The owner of the work basket
- Task list – Unknown ???

#### Access

**Business Space - Page1**

Save Cancel New work basket

General | Access | Distribution targets

| Entity        | Work basket roles                   |                                     |                                     |                                     |                                     |                                     | Task roles                          |                                     |                                     |                                     |                                     |
|---------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
|               | Reader                              | Opener                              | Appender                            | Transfer initiator                  | Distributor                         | Reader                              | Administrator                       | Editor                              | Potential owner                     |                                     |                                     |
| Bonnie        | <input checked="" type="checkbox"/> |
| User or group | <input type="text"/>                |                                     |                                     |                                     |                                     | <input type="button" value="Add"/>  |                                     |                                     |                                     |                                     |                                     |

In the Access category, we get to name a set of users and groups and then associate the authorities of this work basket against those entities. We can define two sets of attributes ... those for the work basket roles and those for the task roles.

#### Distribution Targets

Work Basket Information

Save Cancel New work basket

General | Access | Distribution targets

Selected distribution targets

| Identifier | Name | Type       | Owner | Description |
|------------|------|------------|-------|-------------|
| WB1        | WB1  | Individual | admin |             |

Available distribution targets

| Identifier | Name | Type       | Owner | Description |
|------------|------|------------|-------|-------------|
| WB2        | WB2  | Individual | admin |             |
| aaa        | bbb  | Individual | admin |             |

In the last category known as Distribution targets, we get to list the potential other work baskets to which tasks in this work basket can be routed.

## Human Task Definition for Work Baskets

In the Human Task template definition, the name of the work basket associated with the task can be supplied. Either a hard-coded work basket name or a variable whose value will be the work basket name can be supplied.

To-do Task - WBHT1

|                       |                                                                                     |                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| Description           | People directory:                                                                   | Virtual Member Manager                                                                  |
| <b>Details</b>        | Task priority:                                                                      | 5                                                                                       |
| Propagation           | Work basket identifier:                                                             | WB1  |
| Interface             | Business category:                                                                  |                                                                                         |
| Duration              | Default language: *                                                                 | English - United States                                                                 |
| Event Monitor         | Event handler name:                                                                 |                                                                                         |
| Global Event Settings | Substitution policy:                                                                | No substitution                                                                         |
| Environment           | Date when task becomes valid:                                                       | None Select Date Time zone: America/Chicago Time: 0 : 0 : 0                             |
|                       | <input type="checkbox"/> Business-relevant                                          | <input checked="" type="checkbox"/> Enable subtask creation                             |
|                       | <input type="checkbox"/> Task can be claimed when it is suspended                   | <input checked="" type="checkbox"/> Task can be delegated                               |
|                       | <input checked="" type="checkbox"/> Enable follow-on task creation                  | <input type="checkbox"/> Bind the life cycle to the invoking component                  |
|                       | <input type="checkbox"/> Give owner read access to surrounding process context data |                                                                                         |

## Technical Notes

The default Query Table for work baskets is called “WORK\_BASKET”

Maybe IBM.DEFAULTALLWORKBASKETSLST

To obtain a list of all work baskets, the following REST request can be used:

```
/rest/bpm/htm/v1/workBaskets/query
queryTableName=IBM.DEFAULTWORKBASLETSLST
admin=true
```

The attributes available in the response includes:

- DESCRIPTION
- NAME
- DISPLAY\_NAME
- OWNER
- TYPE
- WBID
- WORK\_BASKET.WBID

A new attribute has been added to a task instance called “workBasketName”

See Also – Work Baskets:

- [InfoCenter - Using work baskets and business categories](#) – WPS 7.0
- [Documentation for Work Basket API \(IBM Internal\)](#)

## ***Business Categories***

Human Tasks can have attributes associated with them called Business Categories. This allows tasks to be categorized for efficient selection and filtering.

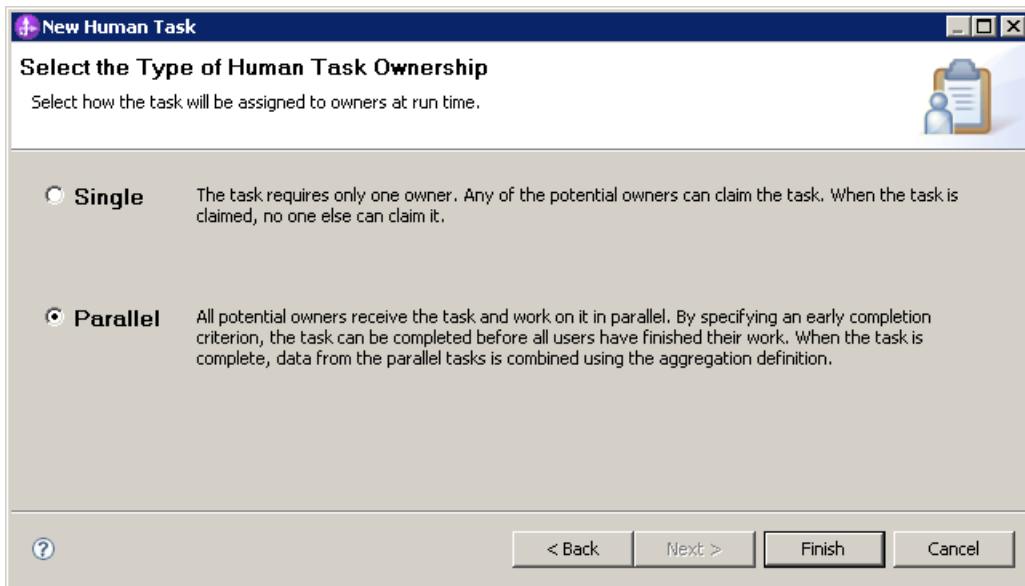
## ***Parallel Tasks***

When a Human Task is created, we normally think of that task as having one owner and it not completing until that single owner completes it. In WPS 7 and beyond, a feature was added which IBM called "Parallel Tasks". The idea behind this is that a Human Task can be created (just as normal) but the Human Task is defined as being parallel in its template definition in WID. When an instance is created, instead of just a single owner being able to claim it, **all** potential owners get to claim the same task. Each potential owner sees their own copy of the task and get to work upon it. When each potential owner completes the task then the original task is flagged as completed and the caller of the task gets to move onwards. Although interesting, there are two more facets to this feature that really bring it to life.

The first is that not all of the child parallel tasks need to be completed before the parent is allowed to complete. An early completion configuration may be defined such that the parent task completes when only some of the child tasks have been completed. If the parent is allowed to complete and there are still other child tasks, these are immediately terminated.

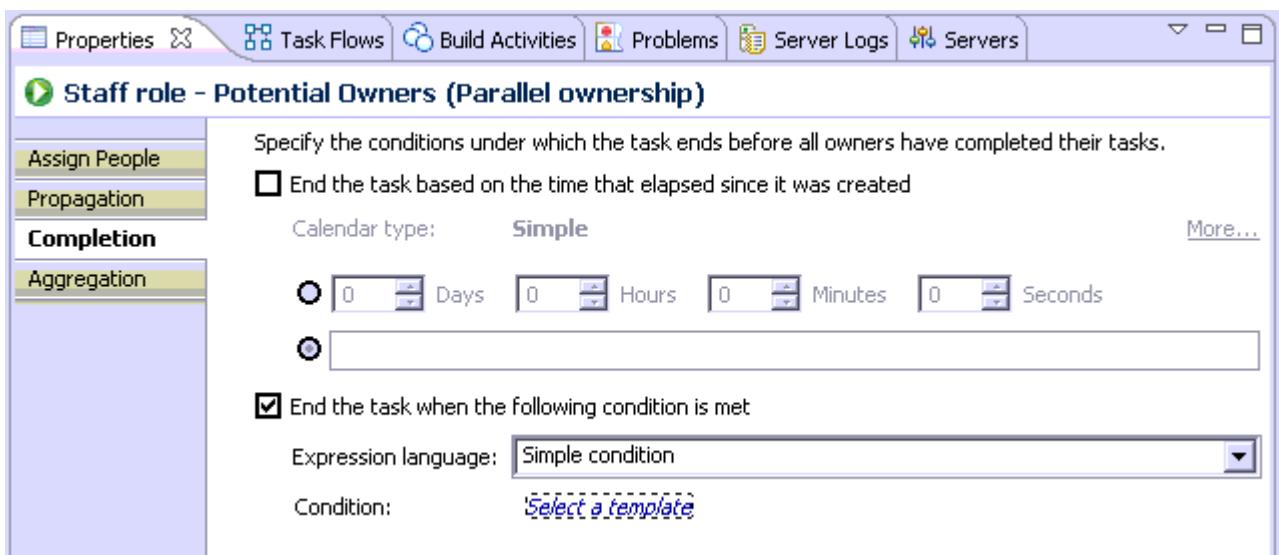
The second important feature is how the configuration of the final response of the parent task is determined. With a simple Human Task, the response from the sole user that claimed the task is the final response but with parallel tasks, there are many different responses from many different users. As such, the ultimate final response will be an aggregate (accumulation) of the responses from all the different users. The decisions of how the aggregations are formed is under the control of the Human Task template designer.

During the creation of a new Human Task template, a panel is shown asking if this task is "single" or "parallel". If single is selected, it is standard human task with one final owner. If Parallel is selected, then the function described in this section becomes active.



## Completion Selection

When a task is defined as parallel, when the Potential Owners selection is selected, the Properties view contains a Completion tab. Within this area the attributes that define the completion criteria for the parent task are defined. This is where we specify under what circumstances the parent can complete before all the children have completed.



Completion can be configured based on either a time interval from when the parent task was created or from examination of the child tasks that have completed. The later is the one that we will focus on as it is most interesting.

The condition that causes completion of the parent task can be specified by either a custom XPath expression or by using one of the templates supplied by IBM. The supplied IBM templates are:

- Percentage of workers – Completes if the number of completed children (of any outcome) has

exceeded a supplied percentage. For example, if we only need 50% of the possible respondents.

- One disapproval – If a single response signals disapproval (a vote) then if our logic says that we need unanimous acceptance, then end the parent task and all its children. If one person disapproves, there is no need to wait for the outcome of the other people.
- One approval – If a single response signals approval (a vote) then we can complete the parent human task.
- Percentage of approvals – This template is used to complete the parent task if a percentage of approvals has been exceeded.
- Majority reached – This template is used to complete the parent task if a majority of outcomes has been reached.

## ***Work Items***

When a Human Task instance is created, a set of potential owners must be determined. This results in a set of "work items" being created. A work item is a relationship between a person and a human task to be worked upon. One way to think of a work item is that it has an attribute called "who" which is the identity of the staff that can work on that item. The "who" attribute can take one of three possible values:

- A named user
- A named group
- A flag that says everybody is eligible to work on that item

When the Human Task is created, a set of such work items are associated with the task. This means that there could be more than one work item which gives us a result that a task can be associated with a combination of users and/or a combination of groups.

## ***Authorization and Staff Resolution***

When a human task instance is created, we want to restrict the set of people who are allowed to work on such a task. This means that we need to determine (resolve) a set of staff members who will be granted the privilege of working upon or owning a task. This is termed staff resolution.

Let us now introduce a concept called "People Assignment Criteria". In previous versions of WPS this was known as staff verbs. Imagine you fall sick and need some medical attention. You want someone to look at you and make a diagnosis. The person that should perform this role can't be just anyone ... they must fulfill a requirement. Specifically, they should be doctors. If you fall ill on an airplane, the flight attendants ask if there is a doctor on board. What they are doing is executing a rule looking for the correct people to fulfill the task.

In WPS, this is implemented by parameterized queries that can be executed against staff repositories that will return a set of userids that can perform a human task. Generically, executing a staff query that results in a set of people is called a staff resolution. The staff query is executed against a staff repository. In reality, the staff repository is commonly an LDAP directory.

## See Also:

- DeveloperWorks – [IBM WebSphere Developer Technical Journal: Expand your user registry options with a federated repository in WebSphere Application Server V6.1](#) – 2007-01-24

## ***Staff Resolution Criteria (aka Verbs)***

Each staff query has associated with it:

- name
- description
- set of parameters
- 

WPS provides some pre-built staff queries:

- Everybody
- Users by user ID
- Users
- Group Members
- Group
- Person Search
- Group Search
- Native Query
- Manager of Employee
- Manager of Employee by user ID
- User Records by user ID
- Users by user ID without Named Users
- Group Members without Named Users
- Group Members without Filtered Users
- User Records by user ID without Named Users
- Department Members
- Role Members
- Nobody

## **Verb Descriptions**

### **Group Search**

This verb retrieves the members of a group where the group is identified by an attribute (not by a name). The attributes of a group that can be matched are:

- type
- industryType
- businessType
- geographicLocation
- affiliates
- displayName
- secretary
- assistant
- manager
- businessCategory
- parentCompany

### **Group Members**

This verb queries a group and adds the set of all users in that group into the result set. It is important to realize that this verb expands the group set into members when the query is executed. This means that if the members of the group are subsequently changed by an administrator, the result set will **still** consist of the original expanded group which may be different than what one would expect if the group were expanded later. This query results in a work item for every possible member of the group and hence if the group is large, it may not be the preferred solution. See the group verb for maintaining a group name until the last possible moment.

|                       |                                                                                  |
|-----------------------|----------------------------------------------------------------------------------|
| GroupName             | The name of the group to query for members                                       |
| AlternativeGroupName1 | An optional group name to query for members that will be added to the result set |
| AlternativeGroupName2 | An optional group name to query for members that will be added to the result set |
| IncludeSubgroups      |                                                                                  |

For LDAP, the default implementation searches for objects of type groupOfNames and from these returns their member attributes. Each member injects a new user into the result set. This object is not the only possible grouping mechanism and groupOfUniqueNames may be desired in which case the XSLT file should be modified.

```
<sldap:staffQueries threshold="$Threshold">
 <sldap:usersOfGroup groupDN='GroupName DN'>
 <sldap:resultObject objectclass='$DefaultGroupClass' />
 <sldap:resultAttribute name="$DefaultGroupClassMemberAttribute"
 destination="intermediate" />
 </sldap:resultObject>
 <sldap:resultObject>
 <sldap:resultAttribute name="$DefaultUserIdAttribute"
 destination="UserID"/>
 </sldap:resultObject>
 </sldap:usersOfGroup>
```

```
</slapd:staffQueries>
```

## Dynamic user selection

A BPEL process can programmatically assign a set of users to a Human Task. This is known as dynamic user selection as the result is never a static definition but rather an algorithmically generated user set dynamically selected at runtime.

One way to achieve this is to use the verb Users by user ID. This can take as a parameter a variable which is of type array of Strings.

## Staff Resolution plugins

Plugins for WPS are available that determine staff sets by contacting directory services. These plugins are specific to the directory type they service.

When a developer builds a staff query, that query is described in abstract terms. For example, "Members of a group". However, the execution of a request against a particular provider is provider specific. The query to return members of a group is one thing for LDAP but another way for VMM. The staff resolution plugins "even out" this disparity. However, the abstract staff query request must be transformed into the correct parameters for the staff resolution plugin. What happens is that the staff query is passed through an XSLT transform prior to being passed to the staff resolution plugin.

See Also:

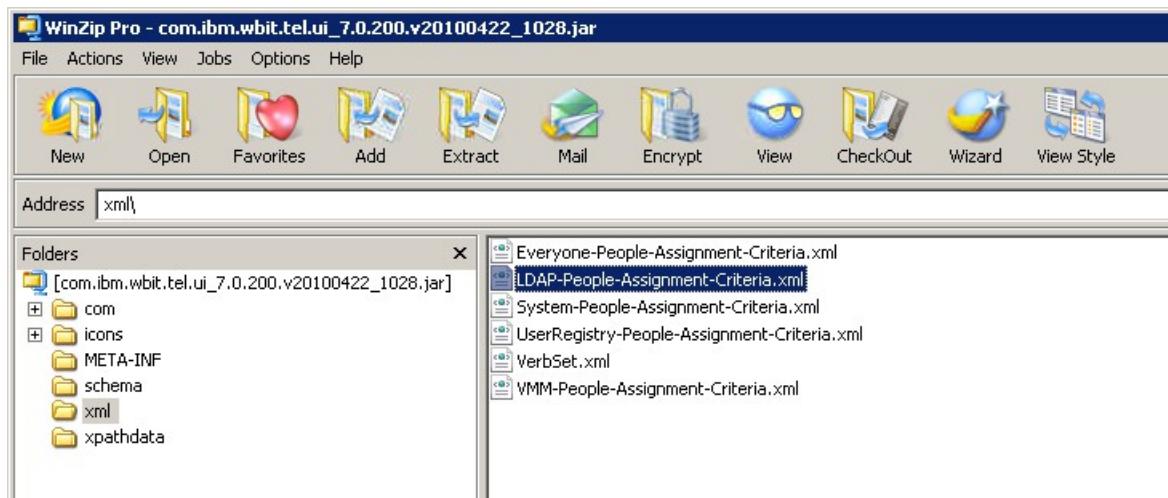
- DeveloperWorks - [Building human tasks with the Virtual Member Manager and LDAP](#) - 2010-11-03

## Custom Human Task staff resolutions

Control files that define the verb sets available for a particular provider can be found in an IBM supplied JAR file. In the directory C:\Program Files\IBM\SDPShared\plugins, a file can be found called:

```
com.ibm.wbit.tel.ui_<version>.jar
```

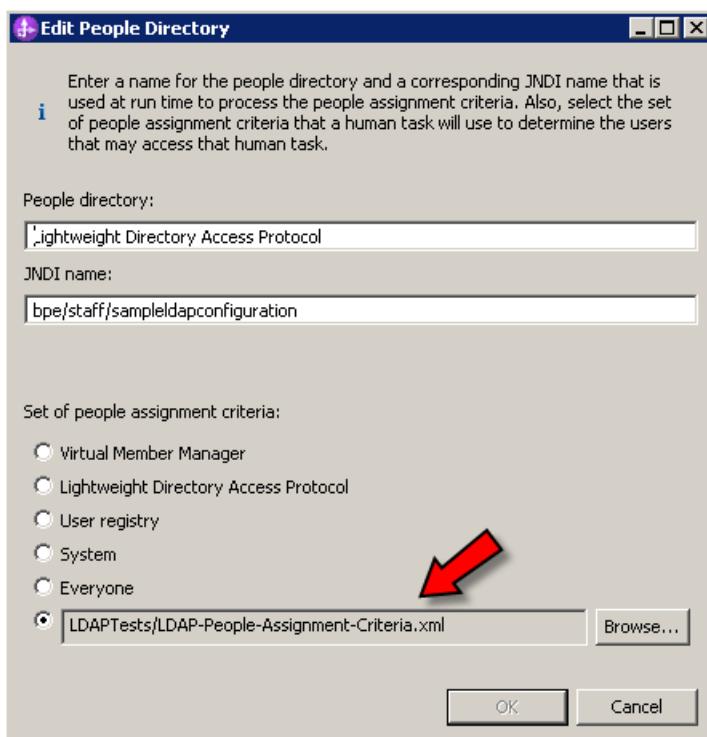
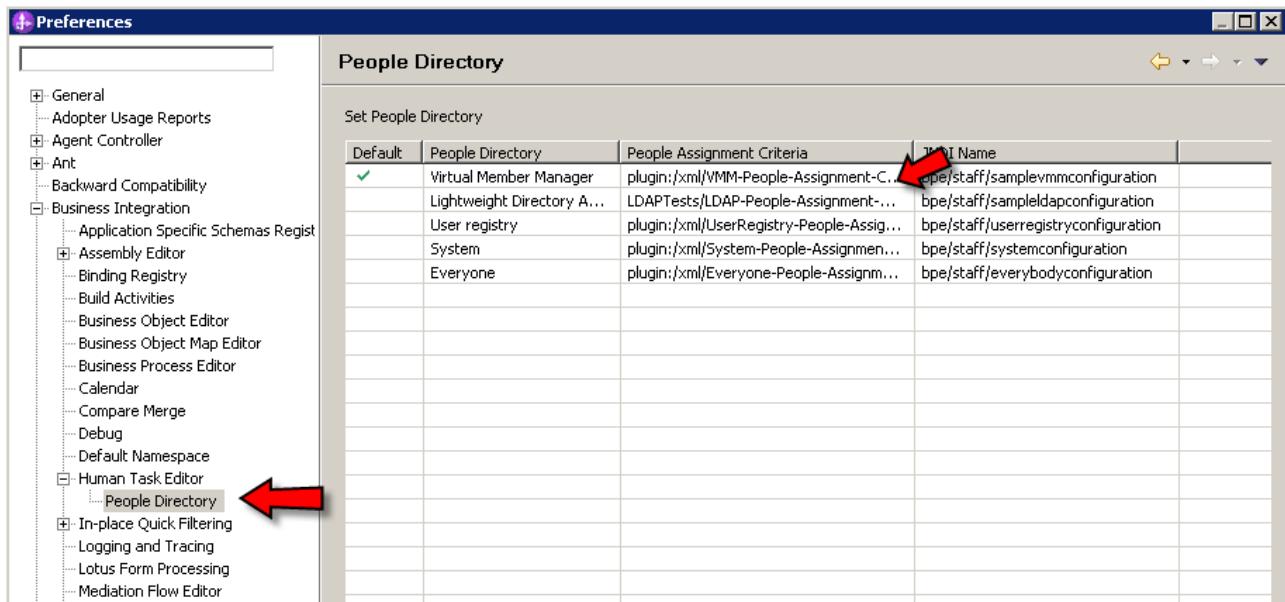
This file can be opened with a ZIP editor. In the JAR file, a folder can be found at /xml. This folder contains a series of XML files that describe the people assignment criteria available in the WID editor.



A copy of the XML file to be modified can be extracted from the ZIP. This XML file contains the definitions of the people selection criteria available for use. To start working with a modified version, add a copy of that file into your module that you are building. Once done, in WID, go to the Windows > Preferences dialog and navigate to:

Business Integration > Human Task Editor > People Directory

Edit the people directory that you are working with and change the People Assignment Criteria to point to the copy of the file:



This file contains a series of verb definitions where each verb is a people assignment criteria.

The format of an entry is as follows:

```
<vs:DefineVerb name='name'>
```

```

<vs:Description>description</vs:Description>
<vs:Mandatory>
 <vs:Parameter>
 <vs:Name>name</vs:Name>
 <vs:type>type</vs:type>
 <vs:hint>hint</vs:Hint>
...
 <vs:hint>hint</vs:Hint>
 </vs:Parameter>
</vs:Mandatory>
<vs:Optional>
 <vs:Parameter>
 <vs:Name>name</vs:Name>
 <vs:type>type</vs:type>
 <vs:hint>hint</vs:Hint>
...
 <vs:hint>hint</vs:Hint>
 </vs:Parameter>
</vs:Optional>
</vs:DefineVerb>

```

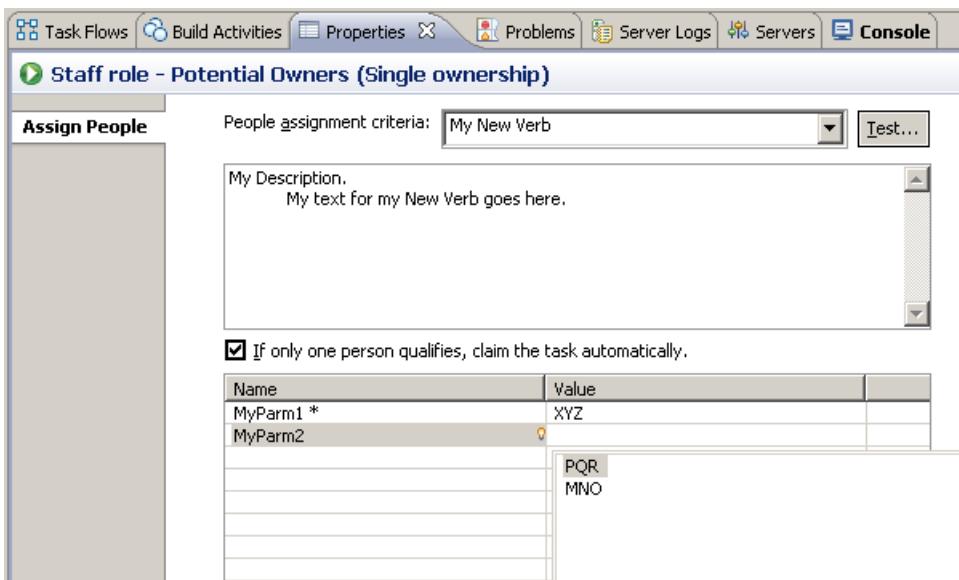
The `DefineVerb` tag has a name which is what is shown in the pull down list in the WID editor. The following `Description` tag defines the text shown in the description area in WID.

The mandatory and optional sections defines zero or more parameters that can be supplied. A parameter is described by its name and type. The type must be taken from one of the following types:

- boolean
- date
- dateTime
- decimal
- double
- duration
- float
- gDay
- gMonth
- gMonthDay
- gYear
- gYearMonth
- xsd:integer
- xsd:string
- xsd:time

Optional hints for the parameters can also be defined. These hints are available using entry assist and simply provide pre-entered values if desired.

Here is an example:



This is generated from:

```
<vs:DefineVerb name='My New Verb'>
 <vs:Description>My Description.
 My text for my New Verb goes here.
 </vs:Description>
 <vs:Mandatory>
 <vs:Parameter>
 <vs:Name>MyParm1</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint>XYZ</vs:Hint>
 <vs:Hint>ABC</vs:Hint>
 </vs:Parameter>
 </vs:Mandatory>
 <vs:Optional>
 <vs:Parameter>
 <vs:Name>MyParm2</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint>PQR</vs:Hint>
 <vs:Hint>MNO</vs:Hint>
 </vs:Parameter>
 </vs:Optional>
</vs:DefineVerb>
```

When a user has entered data in the WID editor, that data for the verb is saved in the Human Task definition file (a .tel file). The format of the entry in this file is:

```
<staff:verb>
 <staff:name>name</staff:name>
 <staff:parameter id='parmName'>value</staff:parameter>
...
 <staff:parameter id='parmName'>value</staff:parameter>
</staff:verb>
```

The staff prefix is bound to the namespace:

```
xmlns:staff="http://www.ibm.com/schemas/workflow/wswf/plugins/staff"
```

In a ".tel" file, the prefix may be "tel" but the namespace is the same.

From this we see that the XML file that describes the verbs is a template that is used solely by the WID editor to describe what the user can enter. Once entered, it becomes data for the Human Task.

During deployment, the human task entries must be mapped or converted to a format suitable for

the staff resolution plugin. This is achieved with an XSLT mapping file. Creation of new verbs requires new XSLT mapping files to accommodate the new verbs. The original files should never be modified, instead new copies of these files should be taken and changed as needed.

## Staff resolution plugin language – The Staff Query

At the conclusion of all of these shenanigans, a request is made to a staff resolution plugin to determine the set of users that are to be allowed to work on the human task. That request is made by sending an XML document. Loosely, this looks as follows:

```
<staff:staffQueries threshold="integer">
 staff query element *
</staff:staffQueries>
```

Here is a more complex example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<sldap:staffQueries
 xmlns:sldap="http://www.ibm.com/schemas/workflow/wsrf/plugins/staff/ldap"
 xmlns:staff="http://www.ibm.com/schemas/workflow/wsrf/plugins/staff"
 xmlns:xalan="http://xml.apache.org/xslt" threshold="1000000">
 <sldap:usersOfGroup groupDN="cn=g1,cn=myRealm,o=k1" recursive="yes">
 <sldap:resultObject objectclass="groupOfNames" usage="recursive">
 <sldap:resultAttribute name="member" destination="intermediate" />
 </sldap:resultObject>
 <sldap:resultObject objectclass="inetOrgPerson" usage="simple">
 <sldap:resultAttribute name="uid" destination="userID" />
 <sldap:resultAttribute name="mail" destination="eMailAddress" />
 <sldap:resultAttribute name="preferredLanguage"
 destination="preferredLocale" />
 </sldap:resultObject>
 </sldap:usersOfGroup>
</sldap:staffQueries>
```

With the exception of `<everybody/>`, `<groupID/>` and `<nobody/>` the `staffQueries` document can contain a variety of children. Each child will add one or more users to the resulting set of users.

The staff query elements define the core rules. The defined rules are:

`everybody`

- `groupID`
- `nobody`
- `remove`
- `userID`

There are others but those are specific to the type of repository against which the request is being made. Multiple queries can be executed within a single staff query and the resulting set of authorized people is the union of all the individual queries.

Simple staff queries.

`everybody`

```
<staff:staffQueries>
 <staff:everybody/>
</staff:staffQueries>
```

Selects everybody. No other queries are allowed in conjunction with this request. This makes sense as everybody really does mean everybody.

### groupID

```
<staff:staffQueries>
 <staff:groupID name="group name"/>
</staff:staffQueries>
```

Selects a named group.

### nobody

```
<staff:staffQueries>
 <staff:nobody/>
</staff:staffQueries>
```

Selects nobody.

### remove

```
<staff:staffQueries>
 <staff:remove value="user ID to remove"/>
</staff:staffQueries>
```

Removes a named userid from the unioned set of userids that have already been built.

### userID

```
<staff:staffQueries>
 <staff:userID name="valid WebSphere user ID"/>
</staff:staffQueries>
```

A named userid.

## ***User registry queries***

```
<sur:user .../>
<sur:usersOfGroup .../>
<sur:search .../>
```

## ***LDAP queries***

When working with the LDAP staff resolution plugin, the following additional functions are available:

- user
- usersOfGroup
- search
- intermediateResult

## user

```
<sldap:user dn="distinguished name of the user"
 attribute="attribute to evaluate"
 objectclass="LDAP objectclass of the queried object"/>
```

The user query is used to lookup an explicit LDAP entry by explicit DN. The "dn" attribute is required and acts as the lookup key in the directory. The "objectclass" attribute is required and names the class of object that is to be returned/found. The "attribute" attribute defines the LDAP entry attribute who's value is to be used for the returned user identity.

## usersOfGroup

```
<sldap:usersOfGroup groupDN="distinguished name of the group"
 recursive="yes"|"no">
 attribute evaluation specification 1+
</sldap:usersOfGroup>
```

## search

```
<sldap:search baseDN="search root"
 filter="valid LDAP filter"
 searchScope="subtreeScope" | "onelevelScope" | "objectScope"
 recursive="yes"|"no">
 attribute evaluation specification 1+
</sldap:search>
```

The baseDN is an optional attribute that supplies an LDAP subtree for the start of the search. If omitted, the baseDN of the plugin configuration will be use. The "filter" attribute is mandatory and used to constrain the search results returned.

## attribute

\*\*\* DEPRECATED \*\*\* - replaced by resultObject

As seen by the last two queries, LDAP entries can be returned. An LDAP entry is composed of a variety of attributes. Somewhere we need to supply the information as to which of the entries attributes should be used and for what purpose. This is where the next tag comes in to play.

```
<sldap:attribute
 name="attribute name"
 objectclass="LDAP object class"
 usage="simple|recursive" />
```

"name" is the required attribute that names the LDAP attribute to be used. "objectclass" is the required attribute that names the object class associated with the entry.

## resultObject

This is a replacement for the old <sldap:attribute> instruction.

```
<sldap:resultObject objectclass="LDAP object class" usage="simple|recursive">
 <sldap:resultAttribute
 name="attribute name"
 destination="userID|eMailAddress|preferredLocale|intermediate" />*
</sldap:resultObject>
```

What this does is look for results that match the LDAP object and, extracts the named attribute value and sets that to be either the userid, email address or locale for that result. If the usage type is

recursive, then the destination can be intermediate. What this seems to mean is that the results are DNs that are themselves retrieved and the rules re-applied.

#### intermediateResult

```
<sldap:intermediateResult name="variable name"
 threshold="positive nonzero integer number"
 query of type user, usersOfGroup or search
</sldap:intermediateResult>
```

"name" is a mandatory parameter and is a variable name. It will be used to temporarily hold the results of the query. This variable can then be used in subsequent queries.

#### intersect

```
<sldap:intersect value="variable name">
</sldap:intersect>
```

This processing instruction expects a variable as input. The variable contains a list of users/identities. When the instruction is reached, an intersection set operation is performed against the identities in the variable and the identities built so far by previous instructions (the result set). The set produced from this intersection becomes the new current result set.

### Examples

#### LDAP group lookup

```
<sldap:staffQueries
 xmlns:sldap="http://www.ibm.com/schemas/workflow/wswf/plugins/staff/ldap"
 xmlns:staff="http://www.ibm.com/schemas/workflow/wswf/plugins/staff"
 xmlns:xalan="http://xml.apache.org/xslt">
 <sldap:usersOfGroup groupDN="cn=nurses,cn=myRealm,o=k1"
 recursive="no">

 <sldap:resultObject objectclass="groupOfNames" usage="recursive">
 <sldap:resultAttribute destination="intermediate" name="member" />
 </sldap:resultObject>

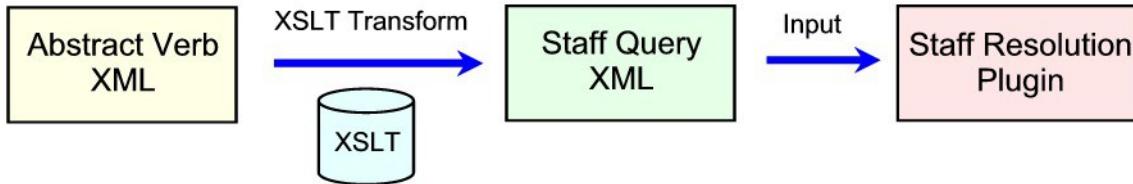
 <sldap:resultObject objectclass="inetOrgPerson" usage="simple">
 <sldap:resultAttribute destination="userID" name="uid" />
 <sldap:resultAttribute destination="eMailAddress" name="mail" />
 <sldap:resultAttribute destination="preferredLocale"
 name="preferredLanguage" />
 </sldap:resultObject>
 </sldap:usersOfGroup>
</sldap:staffQueries>
```

### XSLT Mapping to Staff Resolution plugin language

If you have followed the story so far you will have found that when a staff query is entered in WID, it is transformed into a verb format in the human task .tel file that looks as follows:

```
<staff:verb>
 <staff:name>name</staff:name>
 <staff:parameter id='parmName'>value</staff:parameter>
...
 <staff:parameter id='parmName'>value</staff:parameter>
</staff:verb>
```

This encoding of the staff query is generic. It is not the encoding expected by any one of the IBM supplied staff resolution plugins. During deployment, this format of XML has to be transformed/mapped into the staff resolution plugin language which is also an XML document. WPS does this by executing an XSLT transformation against the verb format XML using an XSLT file that is specific to the Staff Resolution plugin.



These XSLT files can be found in the directory:

<WPS Install>/ProcessChoreographer/Staff

The files currently supplied include:

- EverybodyTransformation.xsl
- LDAPTransformation.xsl
- SystemTransformation.xsl
- UserRegistryTransformation.xsl
- VMMTransformation.xsl

Testing of these XSL files can be performed in WID. We can run an XML document through an XSLT and view the resulting output XML.

Here is an example of a transformed query. First, the original verb XML file:

```

<?xml version="1.0" encoding="UTF-8"?>
<staff:verb xmlns:staff="http://www.ibm.com/schemas/workflow/wsrf/plugins/staff"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <staff:name>Group Members</staff:name>
 <staff:parameter id="GroupName">cn=g1,cn=myRealm,o=k1</staff:parameter>
 <staff:parameter id="IncludeSubgroups">true</staff:parameter>
</staff:verb>

```

and the result after having been passed through an XSLT transform:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<sldap:staffQueries
 xmlns:sldap="http://www.ibm.com/schemas/workflow/wsrf/plugins/staff/ldap"
 xmlns:staff="http://www.ibm.com/schemas/workflow/wsrf/plugins/staff"
 xmlns:xalan="http://xml.apache.org/xslt" threshold="1000000">
 <sldap:usersOfGroup groupDN="cn=g1,cn=myRealm,o=k1" recursive="yes">
 <sldap:resultObject objectclass="groupOfNames" usage="recursive">
 <sldap:resultAttribute name="member" destination="intermediate" />
 </sldap:resultObject>
 <sldap:resultObject objectclass="inetOrgPerson" usage="simple">
 <sldap:resultAttribute name="uid" destination="userID" />
 <sldap:resultAttribute name="mail" destination="eMailAddress" />
 <sldap:resultAttribute name="preferredLanguage"
 destination="preferredLocale" />
 </sldap:resultObject>
 </sldap:usersOfGroup>
</sldap:staffQueries>

```

```
</sldap:usersOfGroup>
</sldap:staffQueries>
```

If a new verb is introduced, the XSLT must be changed to accommodate that new verb. This means the designer must have skills to code and test XSLT as well as knowledge of the design of the existing XSLT program.

The design of IBM's supplied XSLT is roughly as follows.

A template is provided which matches of "/staff:verb". This will find the <verb> elements in the input XML document. The name child element of the verb is saved in a variable called \$verb.

Next there is a large <xsl:choose> block which is a switch on the name of the verb. For each verb, there is a call to a template.

For example, if we have defined a new verb called "supervisors", then an example verb XML might look like:

```
<staff:verb>
 <staff:name>Supervisors</staff:name>
 <staff:parameter id='parmName'>value</staff:parameter>
 ...
 <staff:parameter id='parmName'>value</staff:parameter>
</staff:verb>
```

In the master template of the XSLT, we might then add the following:

```
<xsl:choose>
 ...
 <xsl:when test="$verb='Supervisors'">
 <xsl:call-template name="Supervisors" />
 </xsl:when>
 ...
</xsl:choose>
```

A matching template must also be added to the file.

```
<xsl:template name="Supervisors">
 <sldap:staffQueries>
 ...
 </sldap:staffQueries>
</xsl:template>
```

The addition of a completely new staff resolution entry can be done as follows:

Launch the WAS admin console. Navigate to Resources > People Directory.

Select the type of directory provider to be enhanced (eg. LDAP People Directory Provider).

**People directory provider**

**People directory provider**

People directory providers are responsible for retrieving user information. Each people directory provider is registered with the runtime by specifying a name and a JAR file that contains the people directory provider.

Scope: Cell=Node03Cell, Node=Node01

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#).

Node=Node01

Preferences

| Name                                    | Scope       | Description                                                                                 |
|-----------------------------------------|-------------|---------------------------------------------------------------------------------------------|
| LDAP People Directory Provider          | Node=Node01 | This people directory provider can be used for LDAP based people queries.                   |
| System People Directory Provider        | Node=Node01 | This people directory provider can be used for System based people queries.                 |
| User Registry People Directory Provider | Node=Node01 | This people directory provider can be used for User Registry based people queries.          |
| VMM People Directory Provider           | Node=Node01 | This people directory provider can be used for Virtual Member Manager based people queries. |

Total 4

**People directory provider**

**People directory provider > LDAP People Directory Provider**

People directory providers are responsible for retrieving user information. Each people directory provider is registered with the runtime by specifying a name and a JAR file that contains the people directory provider.

Configuration

| General Properties                  |                                                                           | Additional Properties                          |
|-------------------------------------|---------------------------------------------------------------------------|------------------------------------------------|
| * Scope                             | cells:Node03Cell:nodes:Node01                                             | <a href="#">People directory configuration</a> |
| * Name                              | LDAP People Directory Provider                                            |                                                |
| Description                         | This people directory provider can be used for LDAP based people queries. |                                                |
| * People directory plug-in JAR file | \${WBI_INSTALL_ROOT}/lib/taskldap.jar                                     |                                                |

Apply OK Reset Cancel

**People directory provider**

**People directory provider > LDAP People Directory Provider > People directory configuration**

A People directory configuration is defined for a People directory provider. A People directory provider can have multiple People directory configurations. The configuration contains the JNDI name used to reference it, the XSL mapping file and the values for the custom properties.

Preferences

**New** 



| Select                   | Name                                                       | Description                                                                           | JNDI name                          | XSL Transformation File                                                  |
|--------------------------|------------------------------------------------------------|---------------------------------------------------------------------------------------|------------------------------------|--------------------------------------------------------------------------|
| <input type="checkbox"/> | <a href="#">LDAP People Directory Configuration sample</a> | This sample people directory configuration can be used for LDAP based people queries. | bpe/staff /sampleldapconfiguration | `\${WBI_INSTALL_ROOT}/ProcessChoreographer /Staff/LDAPTransformation.xsl |

Total 1

**XSL Transformation File**

The file name, including the absolute path, of the XSL transformation file.

Path to the new application  **MyCustomTransformation.xsl**

**People directory provider**

**People directory provider > LDAP People Directory Provider > People directory configuration > New**

A People directory configuration is defined for a People directory provider. A People directory provider can have multiple People directory configurations. The configuration contains the JNDI name used to reference it, the XSL mapping file and the values for the custom properties.

Configuration

**General Properties**

\* Scope

\* Name

Description

\* JNDI name

\* XSL Transformation File

The additional properties will not be available until the general properties for this item are applied or saved.

**Additional Properties**

Custom properties

The output of the transformation can be seen in WAS trace with the trace flag:

com.ibm.ws.staffsupport.\*=all

It should be remembered that the XSLT transformation happens **once** during deployment. The result of the transformation is then kept by the WPS runtime. This means that we don't have to perform an expensive XSLT transformation every time we need a staff resolution. It is tempting to place conditional logic in the XSLT that we *think* will execute at runtime for every task but this is not how the architecture works and will lead to unexpected results. Please keep in mind that the XSL transform happens **only** during application deployment and at no other time.

### **Programming tips**

If the value of a Verb parameter determines whether or not to perform processing, consider the following recipe:

```
<xsl:variable name="ParmName">
 <xsl:value-of select="staff:parameter[@id='ParmName']"/>
</xsl:variable>
<xsl:if test="$ParmName != ''">
 <!-- Do something -->
</xsl:if>
```

When testing/developing, **always** test the transformations using the WID XSLT engine. Do not try and debug an XSLT in place in the Human Task engine. It may not report XSLT errors.

After having changed an XSLT file, re-start the WPS server. It appears that is required in order for the resolution to be tried again.

### **Caching of staff resolutions**

When a staff resolution is executed, its results are cached such that if a subsequent query is made, the relatively expensive LDAP lookup need not occur again and the previous result reused. Obviously this can result in stale data being used. HTM provides the ability to clear that cache either manually (immediately) through the admin console or at periodic scheduled times through a component known as the refresh demon.

Through the admin console, the path is:

```
Servers > WebSphere Application Servers > server > "Business Integration" >
"Business Process Choreographer" > Human Task Manager
```

The runtime tab contains an option to refresh

### **Tracing Staff resolution**

Switching on the WAS trace flag `com.ibm.task.*=all` will produce Human Task trace in the trace output data. An additional trace flag that might also be useful is

```
com.ibm.ws.staffsupport.*=all
```

This results in a combined trace flag of:

```
com.ibm.ws.staffsupport.*=all:com.ibm.task.*=all
```

Looking for trace records that start:

```
[5/24/10 14:59:45:603 CDT] 00000040 TraceHTM >
com.ibm.task.staff.StaffQueryManager.getStaffResult(StaffQueryManager.java:1662) ENTRY
StaffQuery =
```

will show the XML passed as input to the XSLT transform for the staff resolution plugin.

If the staff resolution system is LDAP, then it can occasionally be useful to examine the requests and responses flowing to and from the LDAP server. TDS has a an audit log where one can see requests being sent to it but there is no obvious way to see the responses being returned. In addition, other LDAP providers may be used that you are not familiar with. A solution to tracing the LDAP flows is to use the package called [SLAMD](#). As part of this package, a tool called `ldap-decoder` is provided. This tool acts as a proxy/interceptor between an LDAP client and an LDAP server and displays LDAP queries and responses.

Here is a quick cheat sheet on using `ldap-decoder`. First, understand that it is a DOS command window tool and hence must be run from the DOS prompt. It sits *between* a real LDAP provider and a client (such as WPS) that is making LDAP requests. When WPS makes a query against LDAP, it is really making the request to `ldap-provider`. `ldap-decoder` then forwards the request to the real LDAP provider which in turn sends a response back to the `ldap-decoder` tool. This in turn sends the response back to WPS. This means that `ldap-decoder` sees both the requests to LDAP and the responses returned from LDAP. `ldap-decoder` knows how to parse the LDAP data stream (which is binary) and displays the requests and corresponding responses in the DOS console window.

The `ldap-decoder` tool can be found in the tools sub folder of the `slamd` installation directory. Before starting the tool, edit the file called `set-java-home.bat` and change the `JAVA_HOME` environment variable to point to a Java runtime.

To run `ldap-decoder`, execute the following:

```
ldap-decoder -h LDAPProviderHostname -p LDAPProviderPort -L LocalListenPort
```

`ldap-decoder` will then start listening on the `LocalListenPort` and when it receives a request, it will be forwarded to the `LDAPProviderHostname` at port `LDAPProviderPort`. With `ldap-decoder` running, go to human task provider properties and change the URL where the LDAP provider goes to perform lookups. Point this to the `ldap-decoder` tool's port.

See Also – Custom staff resolution:

- [DeveloperWorks - Authorization and staff resolution in Business Process Choreographer: Part 1: Understanding the concepts and components of staff resolution](#)
- [DeveloperWorks - Authorization and staff resolution in Business Process Choreographer: Part 2: Understanding the programming model for staff resolution](#)
- [DeveloperWorks - Authorization and staff resolution in Business Process Choreographer: Part 3: Customization options for staff resolution](#)
- [DeveloperWorks - Authorization and staff resolution in Business Process Choreographer: Part 4: Staff resolution specifications and reference guide](#)
- DeveloperWorks – [WebSphere Application Server Enterprise Process Choreographer: Staff Resolution Architecture](#) – 2003-04-16
- DeveloperWorks – [WebSphere Application Server Enterprise Process Choreographer: Staff Resolution Parameter Reference](#) – 2003-02-21
- [Using set intersection operations in people assignment criteria](#)

## Worked Examples

This section contains some detailed worked examples illustrating how to create custom verbs.

### **Group and User List intersection**

Consider a group that defines the set of all building inspectors in Dallas. Now consider a list of all inspectors in Texas that are certified to work on high-rise buildings. This list will include inspectors in Houston and San Antonio. If we want to assign work to a building inspector in Dallas that is certified to work on high-rise buildings, we need a staff resolution expression that can

build the intersection between a named group of users and a supplied list of users.

When we design such a solution, first we give some thought to how it would appear to an end user. It would show up in WID in the Human Task editor as a new type of selectable Verb for staff resolution.

We decide to call the new verb "Group an Users Intersection".

For most verbs, there are defined parameters and this is no exception. We see that we need two parameters:

- Group Name – The name of a group a users
- Users – A list of named users passed as input

With our knowledge of what must be done to achieve this task, we build an XML fragment that is to be made known to WID to allow us select the verb. A suitable XML fragment would be as follows:

```
<vs:DefineVerb name="Group and Users Intersection">
 <vs:Description>Result set is the intersection between named users and a group</vs:Description>
 <vs:Mandatory>
 <vs:Parameter>
 <vs:Name>GroupName</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint></vs:Hint>
 </vs:Parameter>
 <vs:Parameter>
 <vs:Name>Users</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint></vs:Hint>
 </vs:Parameter>
 </vs:Mandatory>
 <vs:Optional></vs:Optional>
</vs:DefineVerb>
```

This says that there is a new verb called Group and Users Intersection which has two mandatory parameters called GroupName and Users both of data type string.

This XML fragment should be added to the existing XML for the other verbs and made known to WID.

The next thing to do is to modify the XSLT file used to build the staff query language.

```
<xsl:template name="GroupAndUsersIntersection">
 <xsl:variable name="Users">
 <xsl:value-of select="staff:parameter[@id='Users']"/>
 </xsl:variable>
 <sldap:staffQueries>
 <sldap:usersOfGroup recursive="no">
 <xsl:attribute name="groupDN">
 <xsl:value-of select="staff:parameter[@id='GroupName']"/>
 </xsl:attribute>
 <!-- the object class can be either groupOfNames or groupOfUniqueNames -->
 <sldap:resultObject objectclass="groupOfUniqueNames" usage="recursive">
 <!-- The name must be either member or uniqueMember depending on the
Object Class being used -->
 <sldap:resultAttribute destination="intermediate"
 name="uniqueMember" />
 </sldap:resultObject>
 </sldap:staffQueries>
</xsl:template>
```

```

<sldap:resultObject objectclass="inetOrgPerson" usage="simple">
 <sldap:resultAttribute destination="userID" name="uid" />
 <sldap:resultAttribute destination="eMailAddress" name="mail" />
 <sldap:resultAttribute destination="preferredLocale"
 name="preferredLanguage" />
</sldap:resultObject>
</sldap:usersOfGroup>
<sldap:intersect>
 <xsl:attribute name="value">
 <xsl:value-of select="$Users"/>
 </xsl:attribute>
</sldap:intersect>
</sldap:staffQueries>
</xsl:template>

```

### **Group and Group intersection**

In this example we are going to create a verb for group and group intersection. The result set will be the set of users in **both** groups.

The WID definition looks as follows:

```

<vs:DefineVerb name="Multiple Group Intersection">
 <vs:Description>Result set is the intersection between multiple
groups</vs:Description>
 <vs:Mandatory>
 <vs:Parameter>
 <vs:Name>FirstGroupName</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint></vs:Hint>
 </vs:Parameter>
 <vs:Parameter>
 <vs:Name>SecondGroupName</vs:Name>
 <vs>Type>xsd:string</vs>Type>
 <vs:Hint></vs:Hint>
 </vs:Parameter>
 </vs:Mandatory>
 <vs:Optional></vs:Optional>
</vs:DefineVerb>

```

The corresponding XSLT looks as follows:

```

<xsl:template name="MultipleGroupIntersection">
 <xsl:variable name="Group1">
 <xsl:value-of select="staff:parameter[@id='FirstGroupName']"/>
 </xsl:variable>
 <xsl:variable name="Group2">
 <xsl:value-of select="staff:parameter[@id='SecondGroupName']"/>
 </xsl:variable>
 <sldap:staffQueries>
 <sldap:usersOfGroup recursive="no">
 <xsl:attribute name="groupDN">
 <xsl:value-of select="$Group1"/>
 </xsl:attribute>
 <!-- the object class can be either groupOfNames or groupOfUniqueNames -->
 <sldap:resultObject objectclass="groupOfUniqueNames" usage="recursive">
 <!-- The name must be either member or uniqueMember depending on the
Object Class being used -->
 <sldap:resultAttribute destination="intermediate"
 name="uniqueMember" />
 </sldap:resultObject>
 <sldap:resultObject objectclass="inetOrgPerson" usage="simple">

```

```

<sldap:resultAttribute destination="userID" name="uid" />
<sldap:resultAttribute destination="eMailAddress" name="mail" />
<sldap:resultAttribute destination="preferredLocale"
 name="preferredLanguage" />
</sldap:resultObject>
</sldap:usersOfGroup>
<sldap:intermediateResult>
 <xsl:attribute name="name">group2Users</xsl:attribute>
 <sldap:usersOfGroup recursive="no">
 <xsl:attribute name="groupDN">
 <xsl:value-of select="$Group2"/>
 </xsl:attribute>
 <!-- the object class can be either groupOfNames or groupOfUniqueNames -->
 <sldap:resultObject objectclass="groupOfUniqueNames"
usage="recursive">
 <!-- The name must be either member or uniqueMember depending on the
Object Class being used -->
 <sldap:resultAttribute destination="intermediate"
 name="uniqueMember" />
 </sldap:resultObject>
 <sldap:resultObject objectclass="inetOrgPerson" usage="simple">
 <sldap:resultAttribute destination="intermediate" name="uid" />
 </sldap:resultObject>
 </sldap:usersOfGroup>
</sldap:intermediateResult>
 <sldap:intersect value="%group2Users%">
 </sldap:intersect>
</sldap:staffQueries>
</xsl:template>

```

See also – Staff Resolution:

- DeveloperWorks - [Concurrent human task assignment in WebSphere Process Server V7](#) - 2010-06-30

## ***StaffQueryResultPostProcessorPlugin***

When a Human Task is reach and a set of users for a task is determined, the resulting set of users is known as a Staff Query Result. When the query for the users is completed but *before* the result is used, a custom user exit can be called which is supplied the result set that will be used. This exit can modify the result set to add or remove users. This means that the user set can be completely controlled by application logic.

Before we discuss the plugin and its interface, let us first examine the StaffQueryResult interface.

This object represents a result from a staff query. The result can be one of four possible types:

- Everybody – Everyone is authorized in this result
- Nobody – No-one is authorized in this result
- Group – The members of a named group (only 1 group may be named) is authorized
- Userids – A collection of named users is authorized

An instance of a StaffQueryResult can hold data for **one** of these types. A StaffQueryResult can **not** name a group and a set of users. To determine what type of StaffQueryResult is present, the method called `getResultType()` is available. This returns one off:

- `StaffQueryResult.RESULT_TYPE_EVERYBODY`
- `StaffQueryResult.RESULT_TYPE_GROUPIDS`
- `StaffQueryResult.RESULT_TYPE_NOBODY`

- `StaffQueryResult.RESULT_TYPE_USERIDS`

If the type is Userids, then the userid data can be retrieved from the StaffQueryResult with **one of** either of the two methods:

- `Collection getUserData()`
- `Map getUserDataMap()`

Which ever one is called first prevents the other from being used (odd semantics that). The returned data structures (the Collection or the Map) can be modified and will reflect changes to the StaffQueryResult. Another way of saying this is that the returned data structures are references to the real data represented by the StaffQueryResult. The entries in these data collections are of type `UserData` which is a wrapper of three fields: The `UserName`, the `Locale` and the `EmailAddress`.

Instances of `UserData` objects and `StaffQueryResult` objects are created from a `StaffQueryResultFactory` instance.

See Also – Post Processing Plugins:

- DeveloperWorks – [Authorization and staff resolution in Business Process Choreographer: Part 3: Customization options for staff resolution](#) – 2007-12-12

## ***Replacement Expressions***

At runtime, parameters to verbs can be taken from the context in which the process is running. This allows variables to be named in the verb definitions and the content of those variables will be used to supply data to the verb evaluations.

Here are some simple examples:

- The visual description of a human task instance can be set as a function of a parameter in the call to that task
- The potential owner for a task can be set to be the owner of a previously completed task
- The allowable set of expressions vary depending on whether the task is in-line or stand-alone.

## **Stand-alone Expressions**

The following table lists all expressions, that are available for stand-alone human tasks (i.e. tasks that are not embedded inline in a process). These expressions can be used inside descriptions as well as staff assignments. The expressions must be enclosed in '%' characters.

|                                       |                               |
|---------------------------------------|-------------------------------|
| <code>htm:task.originator</code>      | task originator name          |
| <code>htm:task.owner</code>           | task owner name               |
| <code>htm:task.starter</code>         | task starter name             |
| <code>htm:task.administrators</code>  | list of task administrators   |
| <code>htm:task.potentialOwners</code> | list of potential task owners |
| <code>htm:task.editors</code>         | list of task editors          |

|                            |                                 |
|----------------------------|---------------------------------|
| htm:task.readers           | list of task readers            |
| htm:task.potentialStarters | list of potential task starters |

## Staff Variables

|                           |                                              |
|---------------------------|----------------------------------------------|
| htm:property.CustomProp   | value of task's custom property 'CustomProp' |
| htm:task.displayName      | default task display name                    |
| htm:task.description      | default task description                     |
| htm:task.instanceID       | task instance id                             |
| htm:task.BPCEexplorerURL  | BPC Explorer URL to task/escalation details  |
| htm:input.[part][\XPath]  | data from task's input message               |
| htm:output.[part][\XPath] | data from task's output message              |

The htm:input notation may be difficult to get right at the first try. If you open the Interface in the WSDL editor (not the usual Interface editor) then you can find all the pieces of the expression

(click for full size)

The corresponding replacement expression is then:

```
%htm:input.requestContentCommentParameters\request/user%
```

Note that the syntax here is easy to mistype. The XPath expression can be arbitrarily complex. We have seen reports that for some styles of wrapped data it is easiest to do a tree search by beginning the XPath expression with '//', so this expression is usable instead of previous example:

```
%htm:input.requestContentCommentParameters//request/user%
```

Another, and possibly simpler recipe seems to be:

```
%htm:input.<operationName>Parameters\<operationParameterName>/<fieldName>%
```

## Escalation variables

|                                          |                                  |
|------------------------------------------|----------------------------------|
| htm:escalation.description               | default escalation description   |
| htm:escalation.expectedTaskState         | escalation's expected task state |
| htm:escalation.instanceID                | escalation instance id           |
| htm:escalation(escalationName).receivers | escalation receivers             |

## Inline expressions

The following tables illustrate the expressions available to in-line human tasks.

### Process variables

|                                         |                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wf:variable.VariableName.[part][\XPath] | 'VariableName[part][\XPath]' is passed to BFM to resolve the variable/expression. Note: Recent testing seems to be showing that this syntax may no longer be valid. It seems that simply naming the variable in % characters is all this required. For example %myStringVar% or %myBO\myfield% works. |
| wf:property.customPropertyName          | The 'customPropertyName' is passed to the BFM to resolve the custom property.                                                                                                                                                                                                                         |

### Inline staff variables

|                                           |                                                                                       |
|-------------------------------------------|---------------------------------------------------------------------------------------|
| wf:process.starter                        | The starter of the process.                                                           |
| wf:process.administrators                 | The administrators of the process.                                                    |
| wf:process.readers                        | The readers of the process.                                                           |
| wf:activity(activityName).potentialOwners | The potential owners of either the current activity or the activity named in brackets |
| wf:activity(activityName).owner           | The owner of either the activity named in                                             |

|                                   |                                                                              |
|-----------------------------------|------------------------------------------------------------------------------|
|                                   | brackets                                                                     |
| wf:activity(activityName).editors | The editors of either the current activity or the activity named in brackets |
| wf:activity(activityName).readers | The readers of either the current activity or the activity named in brackets |

Note that inline staff variables which refer to sibling activities can only be resolved, if the referred activity is also an inline human task and is already started at the point in time when the staff resolution happens.

### **Getting data from Tasks**

The replacement expressions allow data to be set into tasks, but what about getting data from tasks? The solution to this is the realization that even when a Human Task completes, but before the end of a process, the data associated with that task is not deleted and is available through API. Within a Java Snippet within a BPEL process, the following code can be used:

```
ActivityInstanceData aid = activityInstance("ActivityName");
```

From the ActivityInstanceData object, various properties can be retrieved such as the owner of the activity with the `getOwner()` method.

See Also – Replacement Expression:

- DeveloperWorks – [Dynamic configuration of the Human Task Manager in WebSphere Process Server](#) – 2007-08-22

### **Human Task User Interfaces**

See Also:

- TechNote - [How to embed a custom client type into the human task editor](#) - 2006-10-18

### **Human Task API Event Handlers**

WPS provides extensions that can be used to implement Human Task functionality.

An interface called `APIEventHandlerPlugin5` can be implemented. A default version is already implemented called `APIEventHandler` which can be used to merely over-ride the functions desired.

The interface provides two methods of each function. These are prefixed with pre and post (for example: `preCallTask` and `postCallTask`). The following table lists the functions available to be caught.

| Function    | When Called  |
|-------------|--------------|
| CallTask    | CALL TASK    |
| CancelClaim | CANCEL CLAIM |
| Claim       | CLAIM        |

|                             |                                  |
|-----------------------------|----------------------------------|
| Complete                    | COMPLETE                         |
| CompleteWithFollowOnTask    | COMPLETE WITH FOLLOW-ON TASK     |
| CompleteWithNewFollowOnTask | COMPLETE WITH NEW FOLLOW-ON TASK |
| CreateAndCallTask           | CREATE AND CALL TASK             |
| CreateAndStartTask          | CREATE AND START TASK            |
| CreateAndStartTaskAsSubTask | CREATE AND START TASK AS SUBTASK |
| CreateTask                  | CREATE TASK                      |
| CreateWorkItem              | CREATE WORKITEM                  |
| DeleteTask                  | DELETE TASK                      |
| DeleteWorkItem              | DELETE WORKITEM                  |
| GetTaskAndMarkRead          | GET TASK AND MARK READ           |
| ReplaceWorkItem             | REPLACE WORKITEM                 |
| RestartTask                 | RESTART TASK                     |
| ResumeTask                  | RESUME TASK                      |
| SetBinaryCustomProperty     | SET BINARY CUSTOM PROPERTY       |
| SetCustomProperty           | SET CUSTOM PROPERTY              |
| SetFaultMessage             | SET FAULT MESSAGE                |
| SetInputMessage             | SET INPUT MESSAGE                |
| SetOutputMessage            | SET OUTPUT MESSAGE               |
| SetTaskRead                 | SET TASK READ                    |
| StartTask                   | START TASK                       |
| StartTaskAsSubTask          | START TASK AS SUBTASK            |
| SuspendTask                 | SUSPEND TASK                     |
| SuspendTaskUntil            | SUSPEND TASK UNTIL               |
| SuspendTaskWithCancelClaim  | SUSPEND TASK WITH CANCEL CLAIM   |
| TerminateTask               | TERMINATE TASK                   |
| TransferToWorkBasket        | TRANSFER TO WORK BASKET          |
| TransferWorkItem            | TRANSFER WORKITEM                |

|                    |                      |
|--------------------|----------------------|
|                    |                      |
| TriggerEscalation  | TRIGGER ESCALATION   |
| UpdateInactiveTask | UPDATE INACTIVE TASK |
| UpdateTask         | UPDATE TASK          |

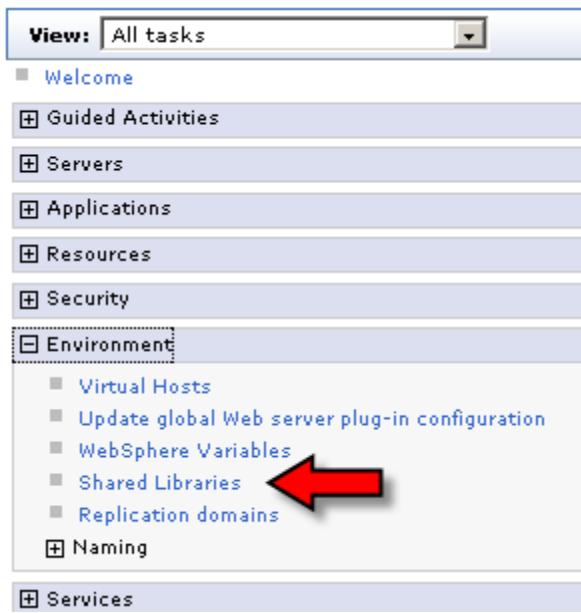
Experimentation seems to show that if the plugin JAR is added to the TaskContainer application as opposed to the system class path then if the TaskContainer is bounced, a new version of the plugin will be loaded. This can be useful during development.

WebSphere Process Server (WPS) provides Human Task Management through the HTM functions. When a Human Task is created, a set of eligible users is chosen for being potential owners of that task and for other roles also related to the task. This set of users is defined in the Task Template during development time in WID. At runtime, we have the option of changing the set of selected users through the concept of a custom plugin written in Java. This plugin is executed during the processing of the operation and can change the result.

To assist with the creation of such a plugin, a sample WID 6.2 Project Interchange has been written which is a skeleton/placeholder for the functions. It is provided as a companion to this document.

Once the plugin has been implemented, it must be installed into the WPS server. The InfoCenter documentation on this is vague and assumes some prerequisite knowledge on the part of the reader. The following illustrates the steps required to install a plugin.

After building the plugin and exporting it as a JAR file, this JAR needs to be registered as a WAS shared library. From within the Admin Console of WAS, create a new shared library.



Create a new shared library entry.

Shared Libraries

**Shared Libraries**

Use this page to define a container-wide shared library that can be used by multiple applications.

Scope: Cell=**WBMonSrv\_wps\_Cell**, Node=**WBMonSrv\_wps\_N**

Scope specifies the level at which the resource definition is visible. For more information about scope and how it works, [see the scope settings help](#).

Node=**WBMonSrv\_wps\_Node**, Server=**server1**

**Preferences**

New 



| Select                   | Name                   |
|--------------------------|------------------------|
| <input type="checkbox"/> | <a href="#">JWLlib</a> |

Give the new shared library a logical name and set the Classpath attribute to point to the absolute location on the local file system of the exported Plugin JAR file.

**Shared Libraries**

**Shared Libraries > PostProcessor**

Use this page to define a container-wide shared library that

Configuration

**General Properties**

\* Scope  
cells:WBMonSrv\_wps\_Cell:nodes:WBMonSrv\_wps\_Node

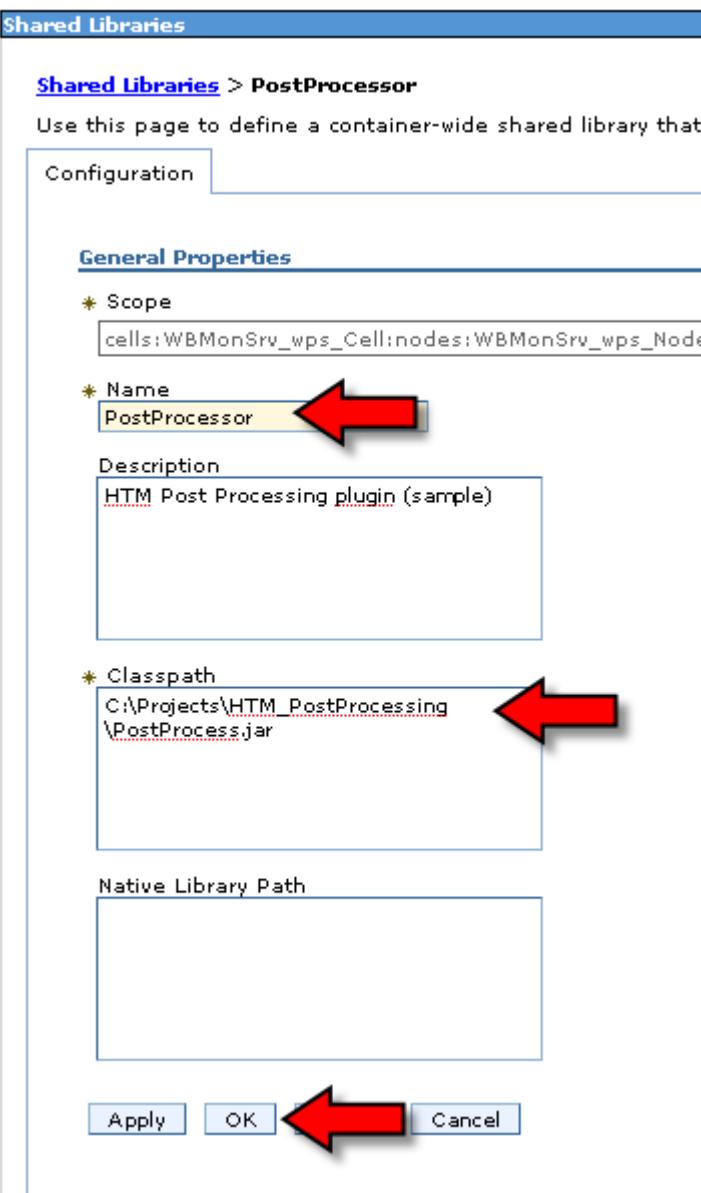
\* Name  
**PostProcessor** 

Description  
HTM Post Processing plugin (sample)

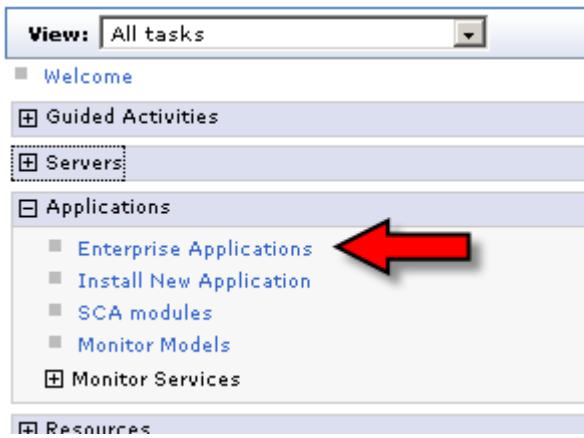
\* Classpath  
C:\Projects\HTM\_PostProcessing  
\PostProcess.jar 

Native Library Path

Apply   OK    Cancel



Next we need to tell the WPS provided Human Task Manager component to include the newly defined shared library in its own classpath. Again from the Admin Console, drill down to the Human Task Manager application.



Select the Human Task Manager application which has a name which starts with TaskContainer.....

|                          |                                                         |  |
|--------------------------|---------------------------------------------------------|--|
| <input type="checkbox"/> | <a href="#">RemoteAL61</a>                              |  |
| <input type="checkbox"/> | <a href="#">Sender1App</a>                              |  |
| <input type="checkbox"/> | <a href="#">TaskContainer_WBMonSrv_wps_Node_server1</a> |  |
| <input type="checkbox"/> | <a href="#">TestController62</a>                        |  |
| <input type="checkbox"/> | <a href="#">ivtApp</a>                                  |  |

A large red arrow points to the application name 'TaskContainer\_WBMonSrv\_wps\_Node\_server1'.

In the References section, click on Shared library references.



Select the complete application (not the modules) by checking the box and then click the Reference shared libraries button.

**Enterprise Applications**

**Enterprise Applications > TaskContainer\_WBMonSrv\_wps\_Node\_server1**

Shared Library Mapping for Modules

Specify shared libraries that the application or individual modules require. You can define configuration at the appropriate scope.

| Select                              | Application                             | URI     |
|-------------------------------------|-----------------------------------------|---------|
| <input checked="" type="checkbox"/> | TaskContainer_WBMonSrv_wps_Node_server1 | -I      |
| Select                              | Module                                  | URI     |
| <input type="checkbox"/>            | HTMIF_WBMonSrv_wps_Node_server1Web      | h.war/w |
| <input type="checkbox"/>            | HTMRESTAPI                              | taskres |

**Reference shared libraries**

**OK**   **Cancel**

In the list of available shared libraries, select the library we added earlier and add it to the list of libraries associated with the application.

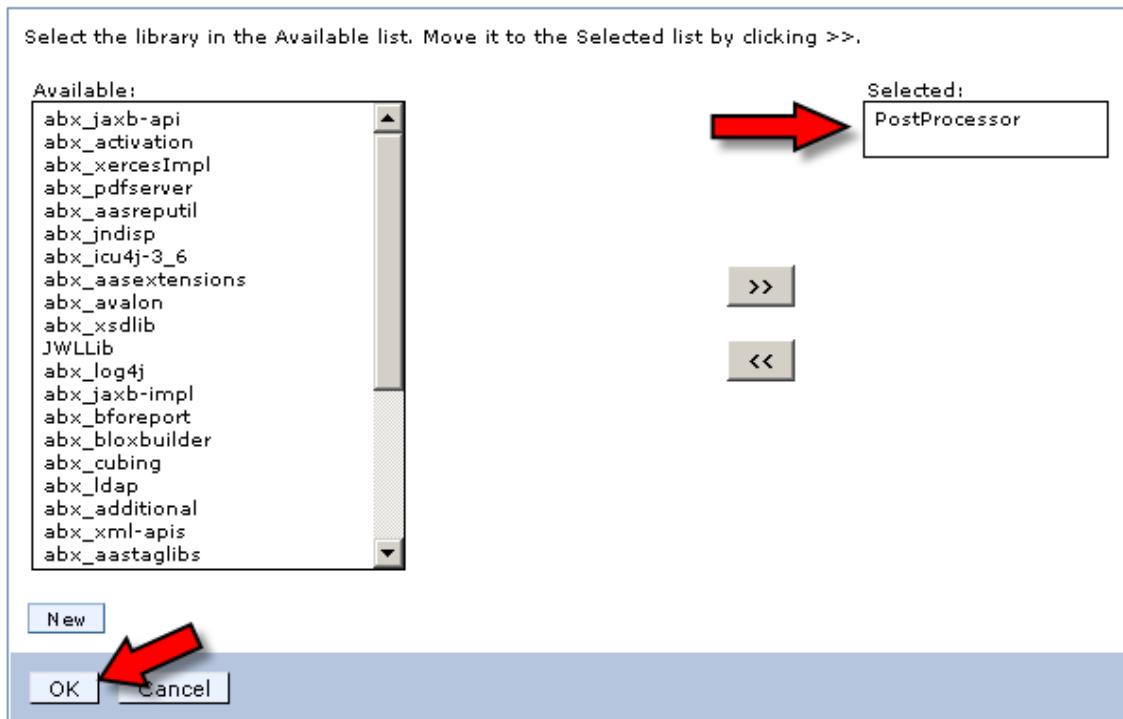
Select the library in the Available list. Move it to the Selected list by clicking >>

**Available:**

- abx\_jAXB-api
- abx\_activation
- abx\_xercesImpl
- abx\_pdfserver
- abx\_aasreputil
- abx\_jndisp
- abx\_icu4j-3\_6
- abx\_aasextensions
- abx\_avalon
- abx\_xsdlib
- JWLLib
- abx\_log4j
- abx\_jAXB-impl
- abx\_bforeport
- PostProcessor**
- abx\_bloxbuilder
- abx\_cubing
- abx\_Idap
- abx\_additional
- abx\_xml-apis

**>>** **<<**

The result will be that the application now references the shared library. Press OK to complete the task.



The shared library has now been associated with the task and we can complete this step with the OK button.

**Enterprise Applications**

[Enterprise Applications](#) > [TaskContainer\\_WBMonSrv\\_wps\\_Node\\_server1](#) > Shared library references

Shared Library Mapping for Modules

Specify shared libraries that the application or individual modules reference. These libraries must be defined in the configuration at the appropriate scope.

| Reference shared libraries |                                         |                                 |                  |
|----------------------------|-----------------------------------------|---------------------------------|------------------|
| Select                     | Application                             | URI                             | Shared Libraries |
| <input type="checkbox"/>   | TaskContainer_WBMonSrv_wps_Node_server1 | META-INF/application.xml        | PostProcessor    |
| Select                     | Module                                  | URI                             | Shared Libraries |
| <input type="checkbox"/>   | HTMIF_WBMonSrv_wps_Node_server1Web      | h.war,WEB-INF/web.xml           |                  |
| <input type="checkbox"/>   | HTMRestAPI                              | taskrestapi.war,WEB-INF/web.xml |                  |

OK      Cancel

We have one last step ahead of us and this is to tell the Human Task Manager to use the plugin during its runtime processing. From the Admin Console, select the Application Server details and in the Business Integration > Business Process Choreographer section, select Human Task Manager.

**Application servers**

**Application servers > server1**

Use this page to configure an application server. An application server is a server that provides services required enterprise applications.

**Runtime** Configuration

---

#### General Properties

Name: server1

Node Name: WBMonSrv\_wps\_Node

Run in development mode

Parallel start

Access to internal server classes: Allow

**Server-specific Application Settings**

Classloader policy: Multiple

Class loading mode: Parent first

#### Container Settings

- Session management
- SIP Container Settings
- Web Container Settings
- Portlet Container Settings
- EJB Container Settings
- Container Services
- Business Process Services

#### Applications

- Default policy set bindings
- Installed applications

#### Server messaging

- Messaging engines
- Messaging engine inbound transports
- WebSphere MQ link inbound transports
- SIB service

#### Business Integration

- Business Integration Configuration
- Business Space Configuration
- System REST Service Endpoints
- Service Component Architecture
- Common Event Infrastructure
- Business Process Choreographer
  - Business Process Choreographer Containers
  - Business Flow Manager
  - Human Task Manager
  - Business Process Choreographer Explorer
  - Business Process Choreographer Event Collector
  - People Assignment Service
- Business Rules
- Custom Properties



Select Custom Properties.

**Application servers**

**Messages**

The Human Task Manager is currently installed. Restart the server for any changes to take effect.

**Application servers > server1 > Human Task Manager**

The human task manager expands the reach of WS-BPEL to include activities that require human interaction as steps in an automated business process. Business processes involving human interaction are interruptible and persistent (a person may take a long time to complete the task) and resume when the person completes the task.

Configuration    Runtime

**General Properties**

**E-Mail Service**

- E-mail session JNDI name: mail/HTMNNotification\_WBMonSrv\_wps\_Node\_server1
- Sender e-mail address: taskmanager.emailservice@htmcompanydomain
- Escalation URL prefix:
- Task URL prefix:
- Administrator URL prefix:
- Process Explorer URL prefix:

**Containers**

- [Business Process Choreographer Containers](#)
- [Deferred Configuration](#)

**Related Items**

- [Business Flow Manager](#)
- [Business Process Choreographer Explorer](#)
- [Business Process Choreographer Event Collector](#)

**Additional Properties**

- [REST Service Endpoint](#)
- [Cleanup Service Jobs](#)
- [People Assignment Service](#)
- [Custom Properties](#)

**People Resolution**

People query refresh schedule: 0 0 1 \* \* ?

Timeout for people query result [sec]: 3600

Enable group work items

**Related Resources**

In the Custom Properties for the Human Task Manager application, click on the property called Staff.PostProcessorPlugin. This will allow us to edit/change/set the value of this property.

| Application servers > server1 > Human Task Manager > Custom Properties                                 |                                                                                                                                                                                                                                                                                                                                         |                                                                 |
|--------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| Specifies an arbitrary name-value pair. The value is a string that can set internal system properties. |                                                                                                                                                                                                                                                                                                                                         |                                                                 |
| + Preferences                                                                                          |                                                                                                                                                                                                                                                                                                                                         |                                                                 |
|                                                                                                        | <input type="button" value="New"/> <input type="button" value="Delete"/>                                                                                                                                                                                                                                                                |                                                                 |
|                                                                                                        |     |                                                                 |
| Select                                                                                                 | Name ▾                                                                                                                                                                                                                                                                                                                                  | Value ▾                                                         |
| <input type="checkbox"/>                                                                               | <a href="#">EscalationEmail.ClientDetailURL</a>                                                                                                                                                                                                                                                                                         |                                                                 |
| <input type="checkbox"/>                                                                               | <a href="#">EscalationEmail.MaxRetries</a>                                                                                                                                                                                                                                                                                              | 4                                                               |
| <input type="checkbox"/>                                                                               | <a href="#">EscalationEmail.RetryTimeout</a>                                                                                                                                                                                                                                                                                            | 3600                                                            |
| <input type="checkbox"/>                                                                               | <a href="#">EscalationEmail.Subject</a>                                                                                                                                                                                                                                                                                                 | The task '%htm:task.displayName' escalated                      |
| <input type="checkbox"/>                                                                               | <a href="#">EscalationEmail.Template</a>                                                                                                                                                                                                                                                                                                | file:\${WAS_INSTALL_ROOT}/Pro<br>/sample/emailNotification.html |
| <input type="checkbox"/>                                                                               | <a href="#">Staff.Diagnosis</a>                                                                                                                                                                                                                                                                                                         | development_mode                                                |
| <input type="checkbox"/>                                                                               | <a href="#">Staff.PostProcessorPlugin</a>                                                                                                                                                                                                                                                                                               | none                                                            |
| <input type="checkbox"/>                                                                               | <a href="#">Staff.PostProcessorPlugin.EnableResultSharing</a>                                                                                                                                                                                                                                                                           | false                                                           |

Set the value of this property to be the logical name of the plugin. Note that this need not be the same name as the Java class that implements the plugin so take care here.

**Application servers**

**Application servers > server1 > Human Task Manager > Custom**

Specifies an arbitrary name-value pair. The value is a string that

Configuration

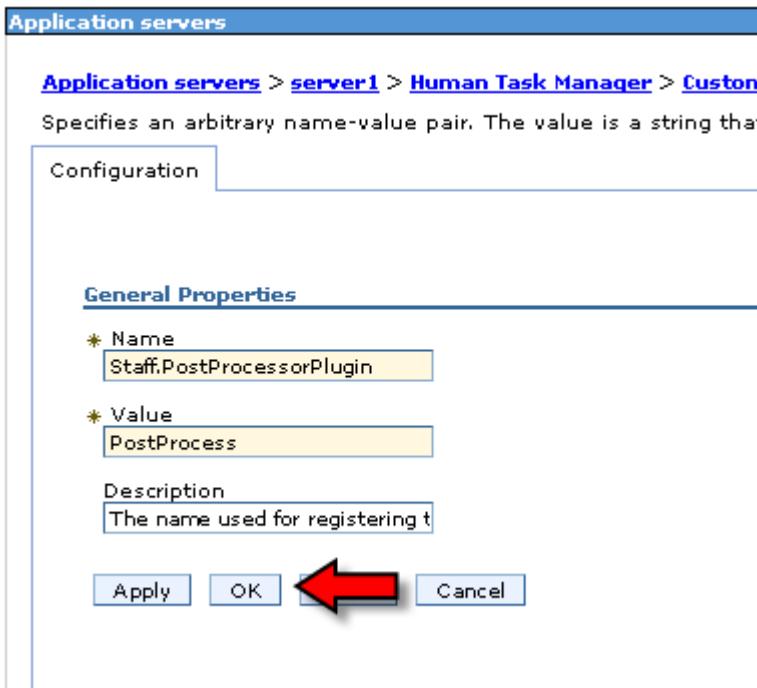
**General Properties**

\* Name  
Staff.PostProcessorPlugin

\* Value  
PostProcess

Description  
The name used for registering t

Apply   OK   **Cancel**



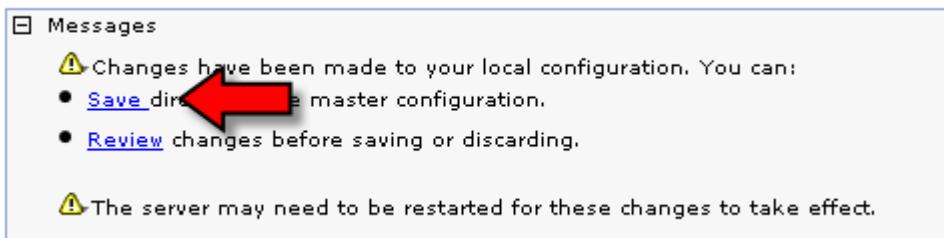
We have now completed all the necessary setup and can save all our changes.

**Messages**

⚠ Changes have been made to your local configuration. You can:

- [Save\\_dir](#) to save the master configuration.
- [Review](#) changes before saving or discarding.

⚠ The server may need to be restarted for these changes to take effect.



The changes will not be visible to WPS until the next server restart so go ahead and restart the server.

# Query Tables

## *Query Table Editor*

The query table editor is provided as part of a Support Pac called

PA71: WebSphere Process Server - Query Table Builder

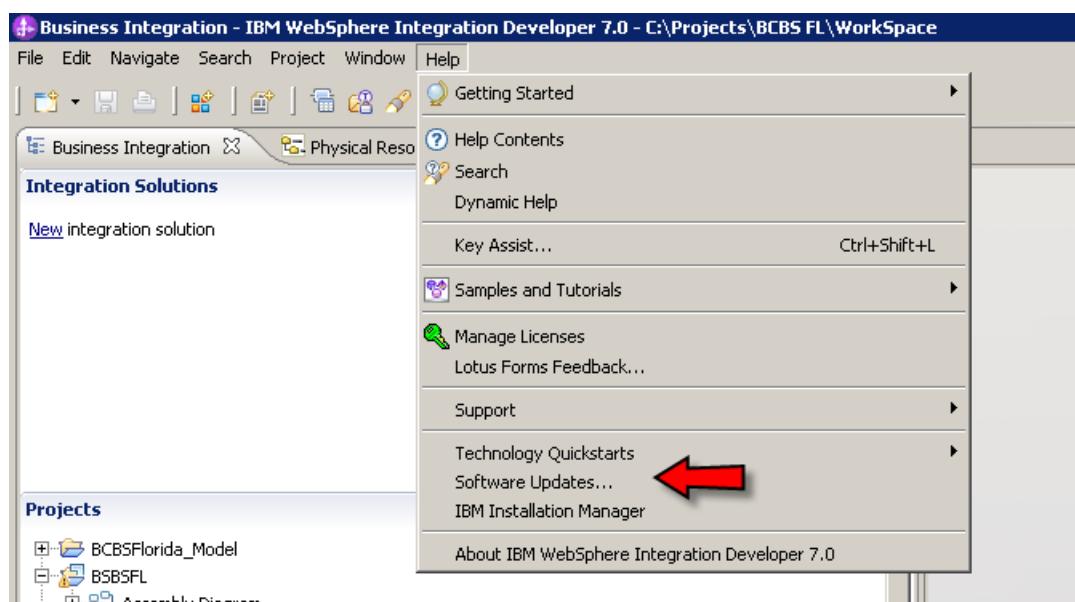
This can be found at the following URL:

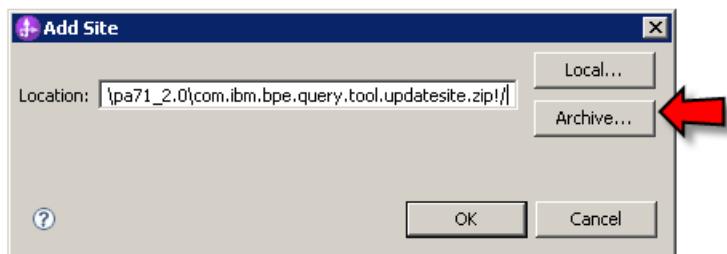
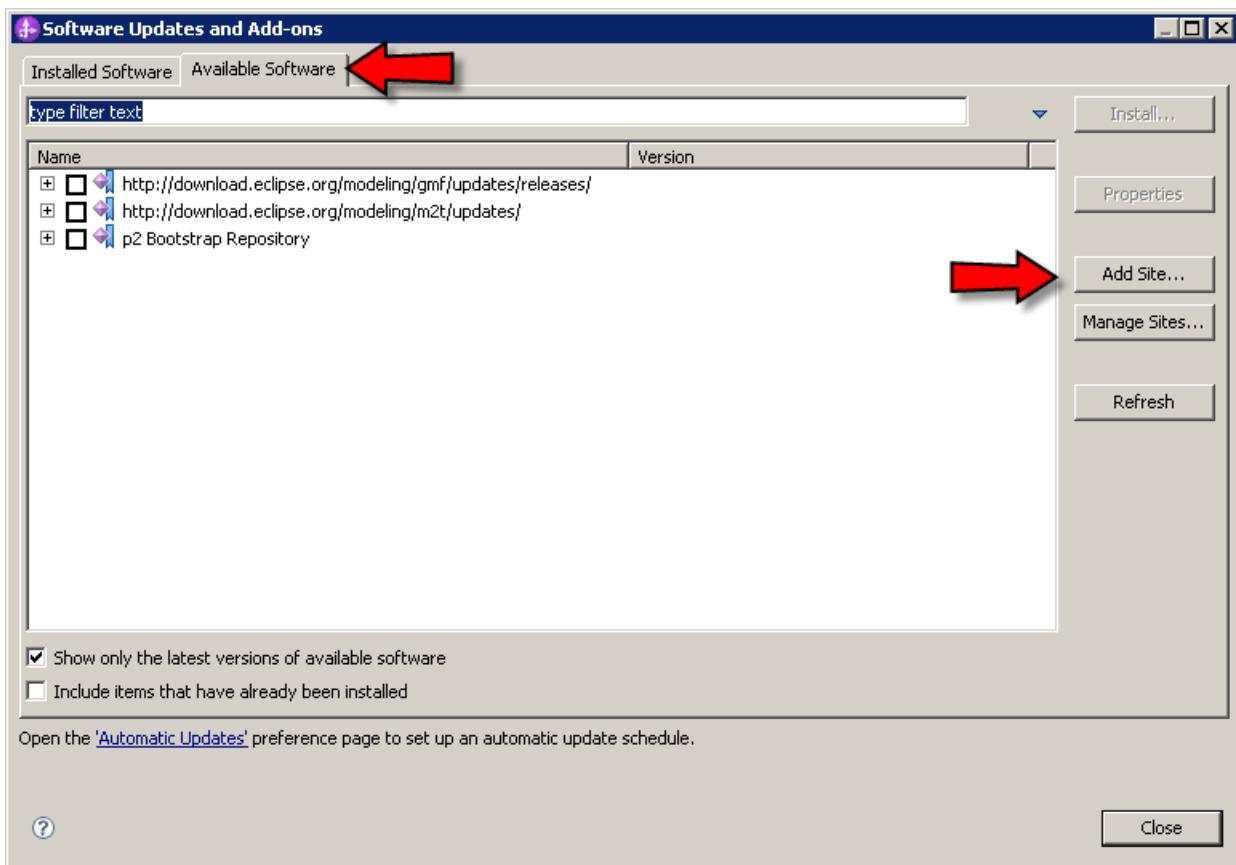
<http://www-01.ibm.com/support/docview.wss?uid=swg24021440>

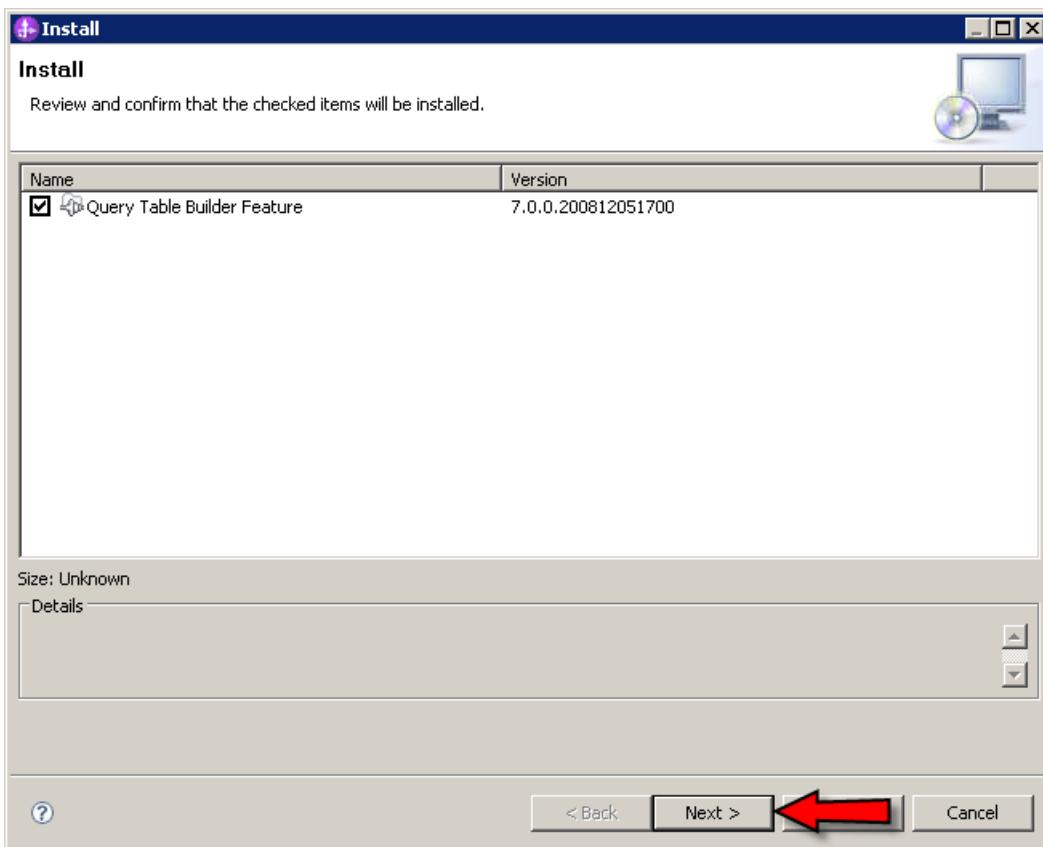
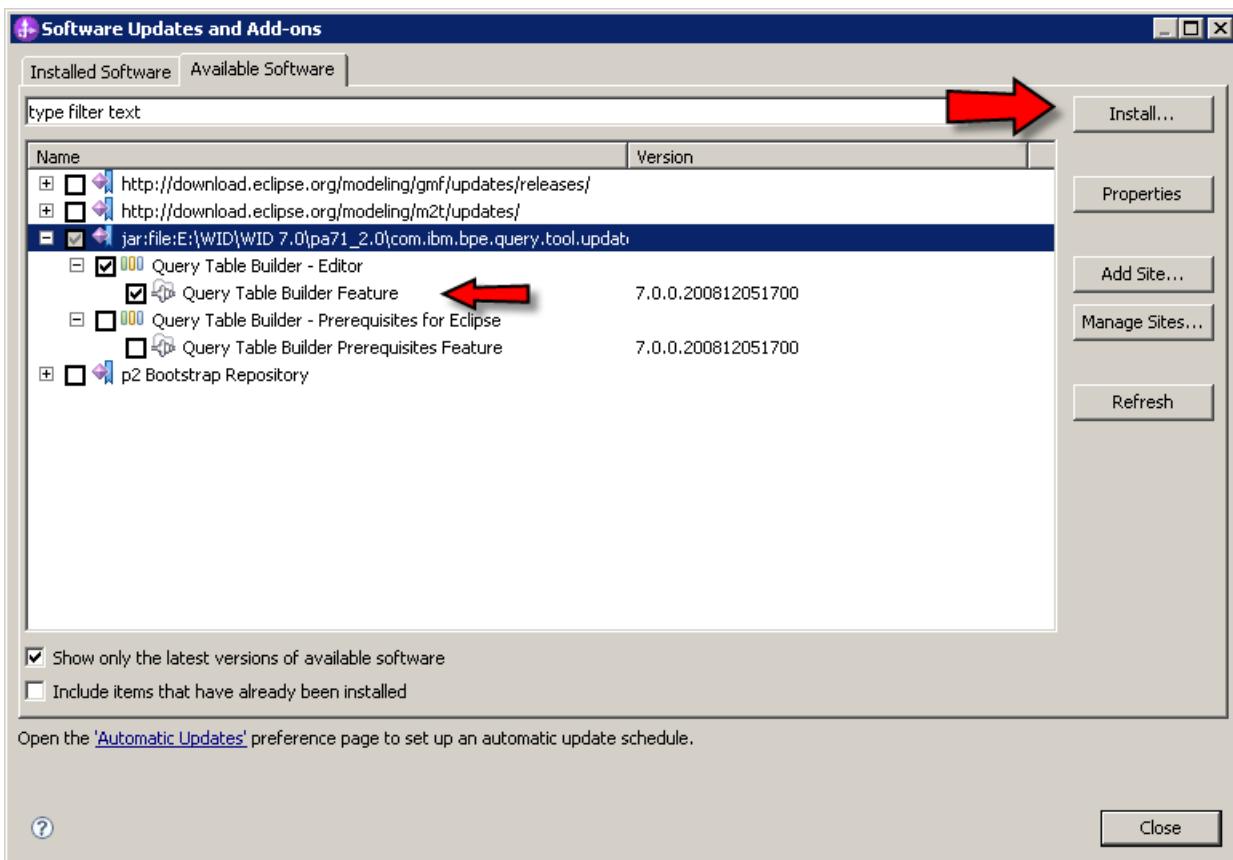
There is detailed documentation associated with using query tables that can be found with the Support Pac.

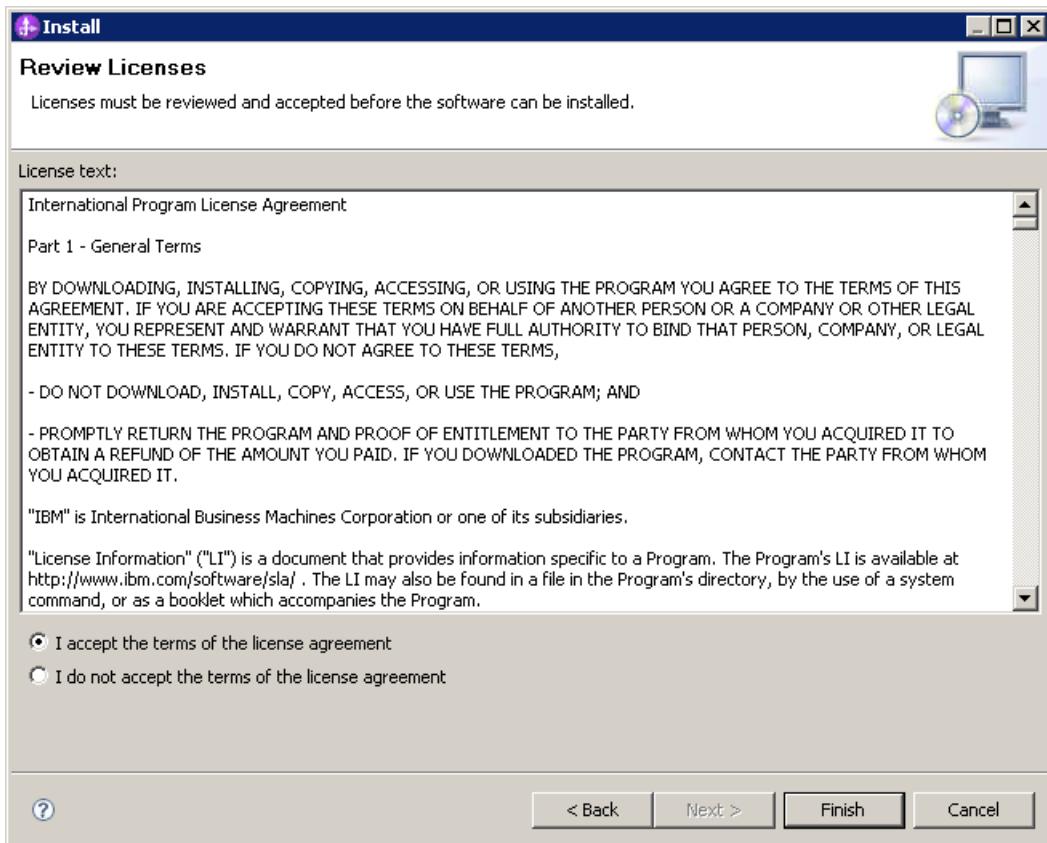
## **Installation of Query Table Editor**

Download the pa71\_<version>.zip file to a directory on the same machine as WID and extract its content. Start WID.









## ***Query Table programming***

The query tables can be accessed via REST interfaces. These include:

### Task Instance Query Tables Resource

This request retrieves a list of the query tables and their attributes. The general syntax for this request is:

```
GET /rest/bpm/htm/v1/tasks/queryTables
```

### Task Instance Entity List Resource

This request retrieves instances of rows in the query tables. The general syntax for this request is:

```
GET /rest/bpm/htm/v1/tasks/query
```

A mandatory parameter called `queryTableName` is needed. This is the name of the table that is to be queried.

Additional parameters include `queryFilter`. This is a HTM standard query that can be used to locate a subset of entries. The default is to return all entries.

An example REST request might be:

```
/rest/bpm/htm/v1/tasks/query?queryTable=MYTABLE.MYDATA&queryFilter=KIND%3DKIND_PARTICIPATING AND STATE%3DSTATE_READY
```

Note the escape character sequence of %3D which is '='.

## Using Query Tables with Business Space

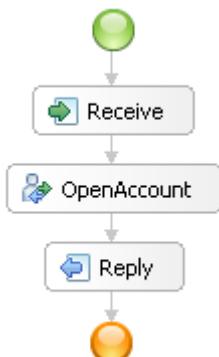
As part of the WPS 6.2 Feature Pack that was made available in July 2009, a new set of Business Space widgets were included. Amongst the new collection is a widget called Tasks List. It presents lists of Human Tasks in various states. From these tasks, a user can select a task and work upon it.

The basic view of a task in the Tasks List widget looks as follows:

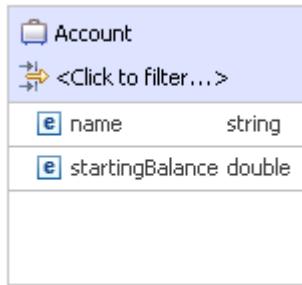
The screenshot shows the WPS 6.2 Business Space interface. At the top, there's a navigation bar with 'Your Business Space' logo, 'Welcome admin | Help | Logout', and 'Manage Business Spaces'. Below the navigation bar is a toolbar with tabs: 'Test1', 'eee', 'sss', 'QueryTables' (which is selected), and 'New Page'. To the right of the toolbar is an 'Add Widgets' button. The main content area is titled 'Tasks List' and displays a table of tasks. The table has columns: Name, Description, Owner, Due time, Start date, Priority, and Pending. One task is listed: 'OpenAccount Open a new account' with 'Jul 21, 2009 2:36:17 PM' as the start date, 'Low' priority, and a pending status. At the bottom of the table is a pagination control showing '1 1'.

Although this looks *fine*. We can do better than this. New for the Tasks List widget is the ability to configure in great detail what is shown on a per task basis. Importantly, the columns shown for a task can include data that can allow the user to see *into* that task without having to open it up.

By way of a trivial example, consider a Business Process that looks like this:



The process includes a human task that is used to present some data to a user. Now assume that the data includes the following Business Object definition:



In the Environment section of the Human Task, a couple of custom properties have been defined:

Build Activities Properties Problems Server Logs Servers

### Human Task - OpenAccount

Description Details Server Exit Condition Expiration Environment Event Monitor Global Event Settings

Custom Properties:

| Name         | Value                    |
|--------------|--------------------------|
| CustomerName | %input\\name%            |
| Balance      | %input\\startingBalance% |

Remove Edit... Add...

Wouldn't it be wonderful if the Business Space Tasks List could also display information relating to the task on the list of tasks? It might look something like this:

Your Business Space Welcome admin | Help | Logout

Feature Pack ▾ Manage Business Spaces

Test1 eee sss QueryTables New Page Add Widgets ▾

Tasks List

Bank Account ▾ Available Tasks ▾ Actions ▾

|                          | Description        | Owner | KIND       | STATE     | Customer Name | Balance |
|--------------------------|--------------------|-------|------------|-----------|---------------|---------|
| <input type="checkbox"/> | Open a new account |       | To-do task | Available | John Smith    | 349.88  |

1 1

Notice that the task list contains application data (Customer Name and Balance) that came

from custom properties values.

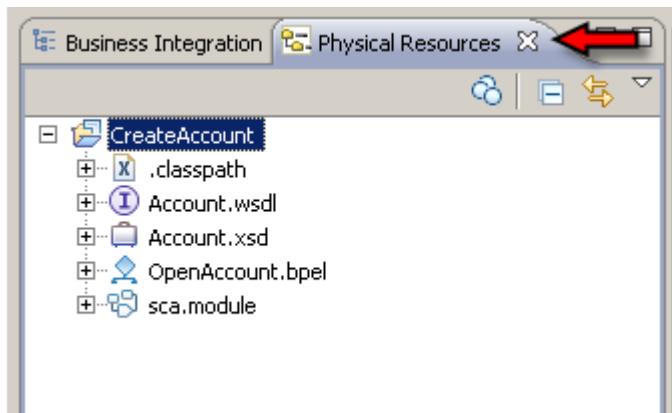
The remainder of this paper will show you exactly how to achieve this effect.

The first thing we need to do is to create a Query Table associated with the data that we want to display for this type of task. A detailed description of Query Tables is beyond the scope of this document. However, the document does assume that the PA71 - Support Pac called WebSphere Process Server – Query Table Builder has been installed. See:

<http://www-01.ibm.com/support/docview.wss?uid=swg24021440>

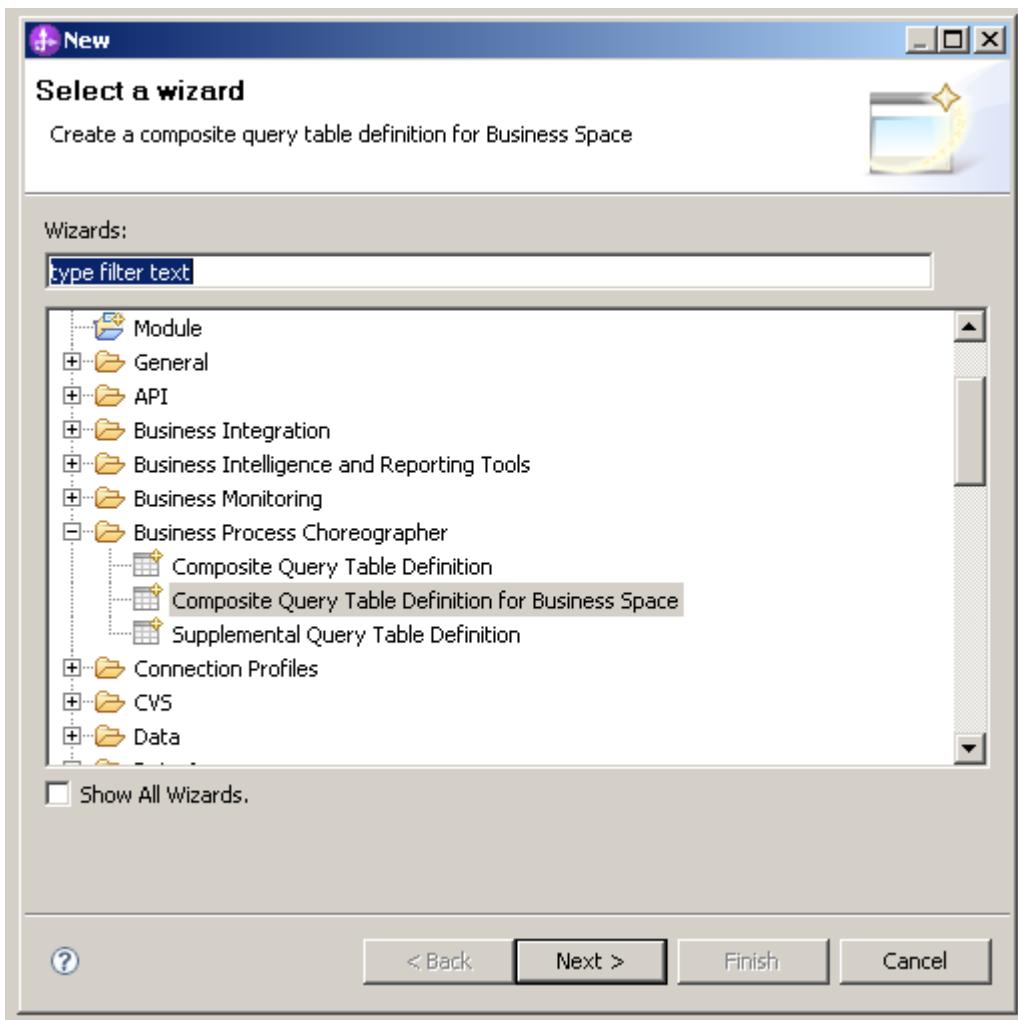
Switch the view to Physical resources

In the Business Integration view, click on Physical Resources.



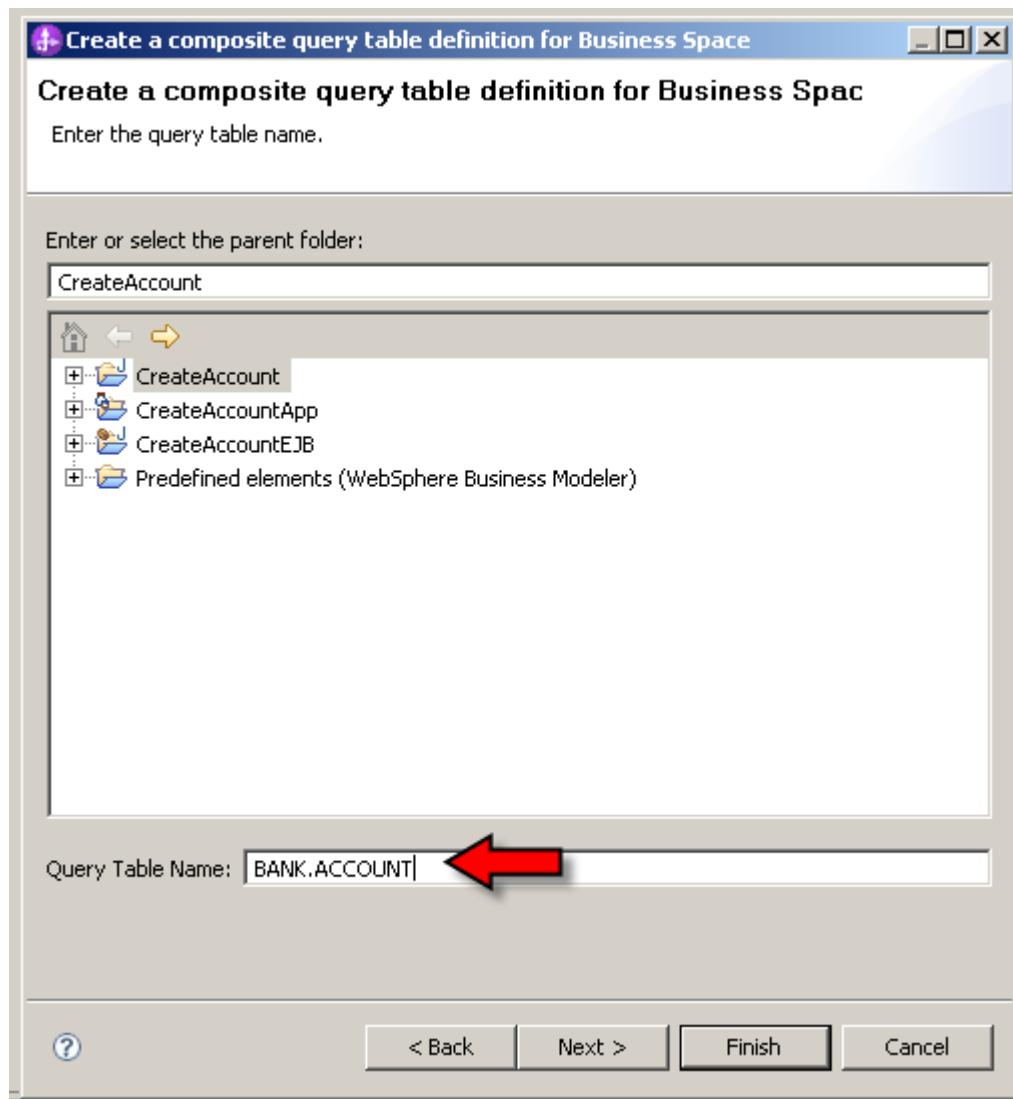
Create a new Query Table for Business Space

Create a new artifact. In the wizard list, expand Business Process Choreographer and select “Composite Query Table Definition for Business Space”.



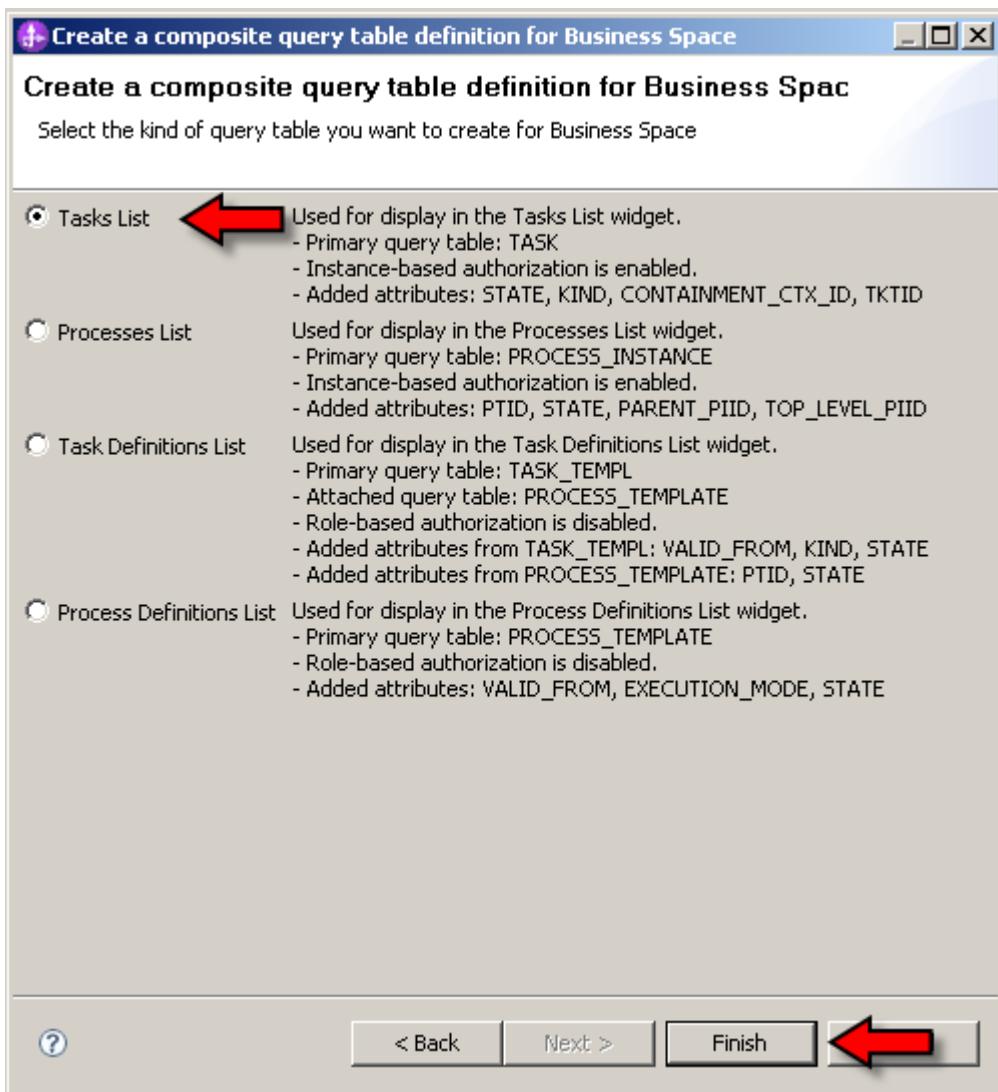
Give the query table a name

Each query table must be given a name. We will call ours "BANK.ACOUNT".



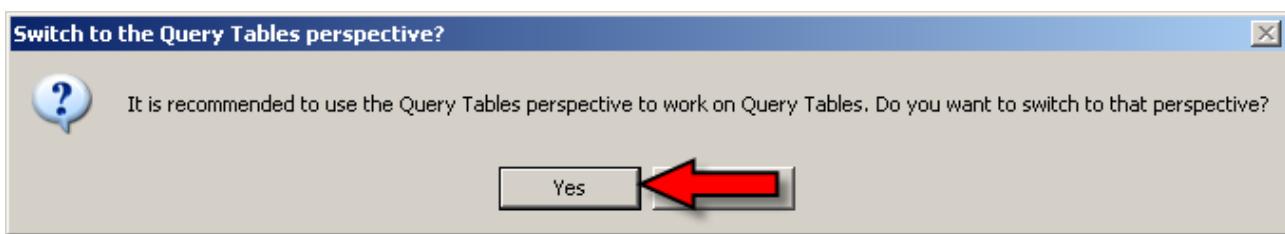
Described the type of Query Table

There are multiple types of query table that can be built. The Query Table we want is to be used to show task lists so we select the Tasks List entry from the set of available query tables.



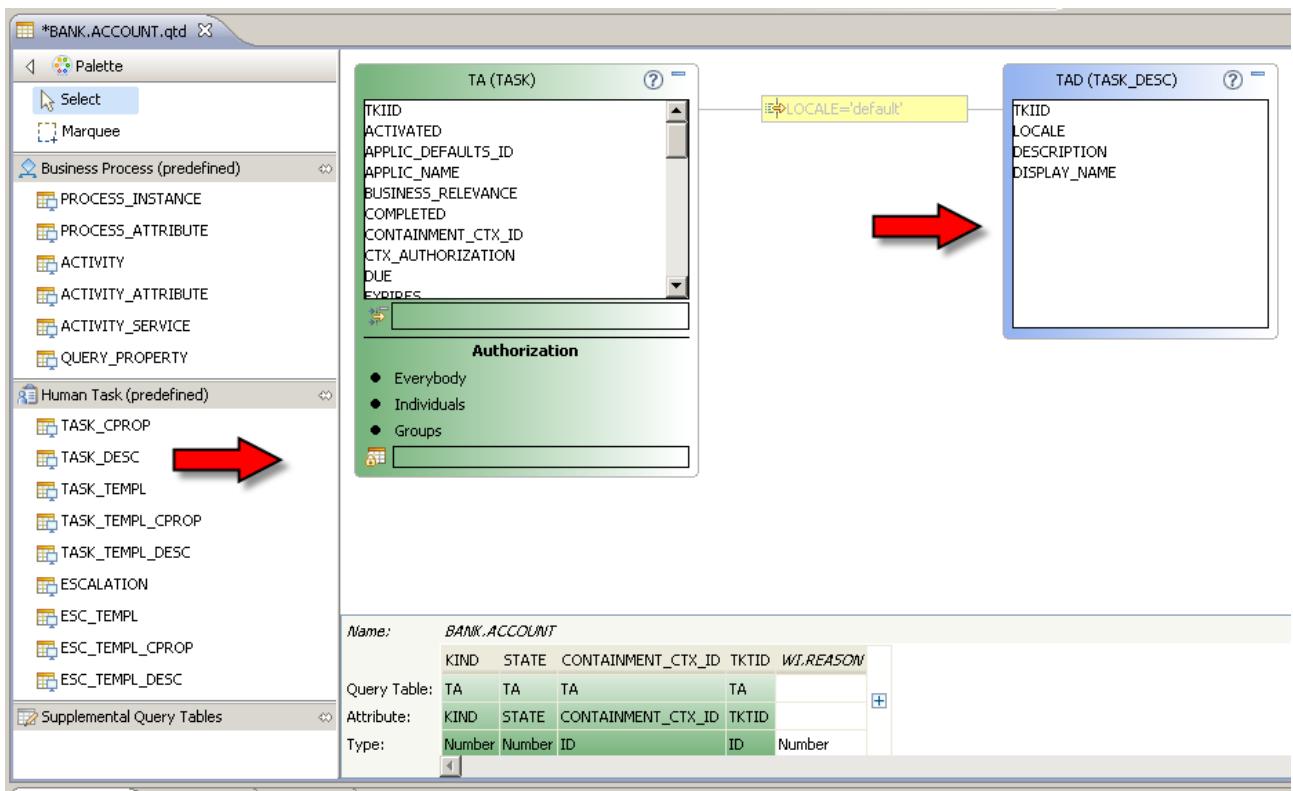
Open the Query Table builder

When prompted to open the Query Tables perspective, select Yes.



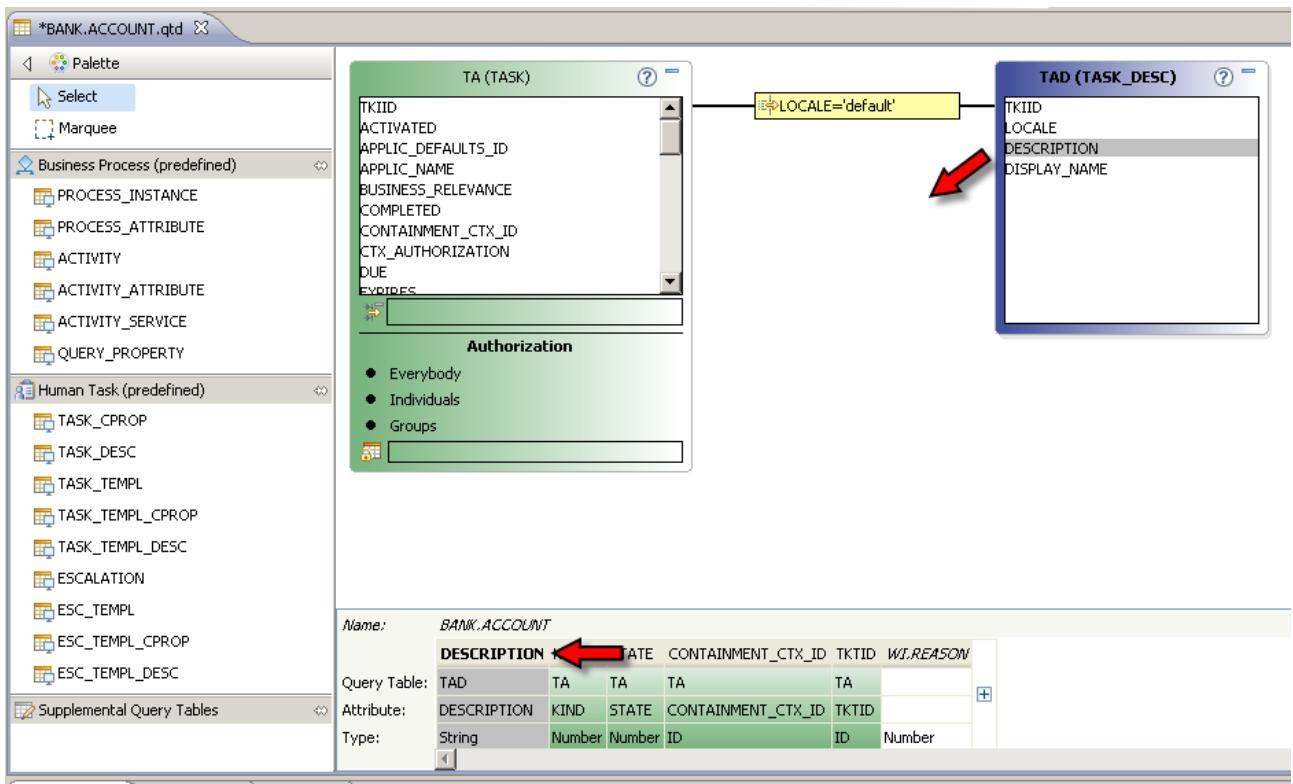
Add the TASK\_DESC table

In the result, we want the description of the task to be included so we add the TASK\_DESC table as a related table. Drag it from the Human Task predefined tables and drop it on the canvas.



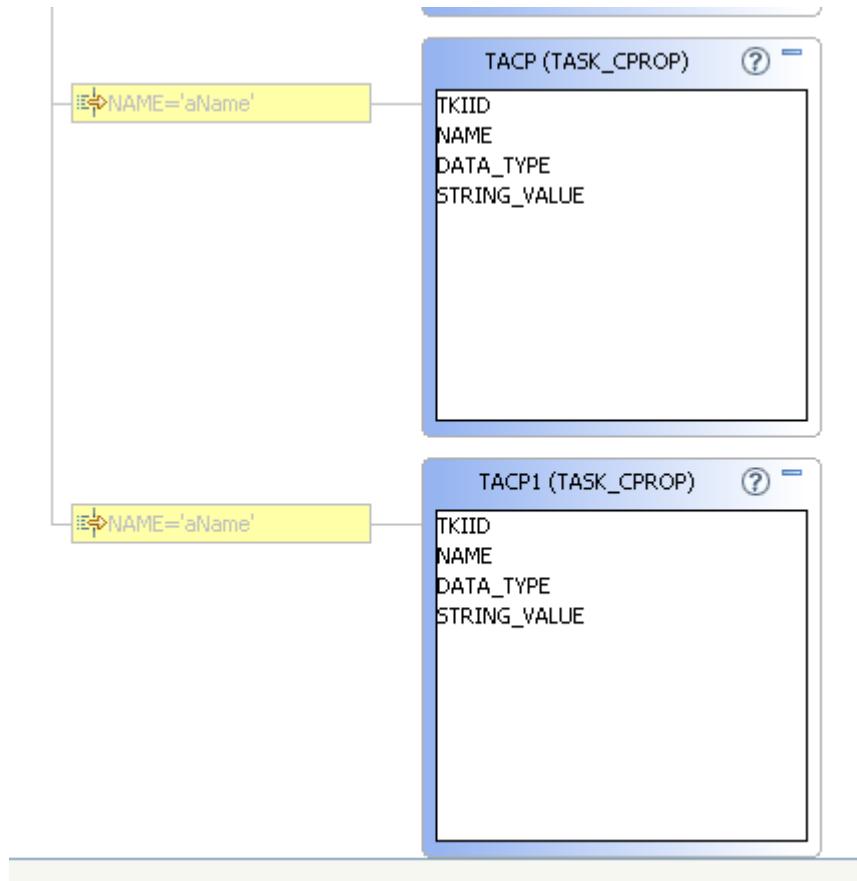
Add the DESCRIPTION into the resulting query table

Drag the DESCRIPTION field from the TASK\_DESC table into the BANK.ACCOUNT set of columns. This will allow DESCRIPTION to be shown in the query result set.



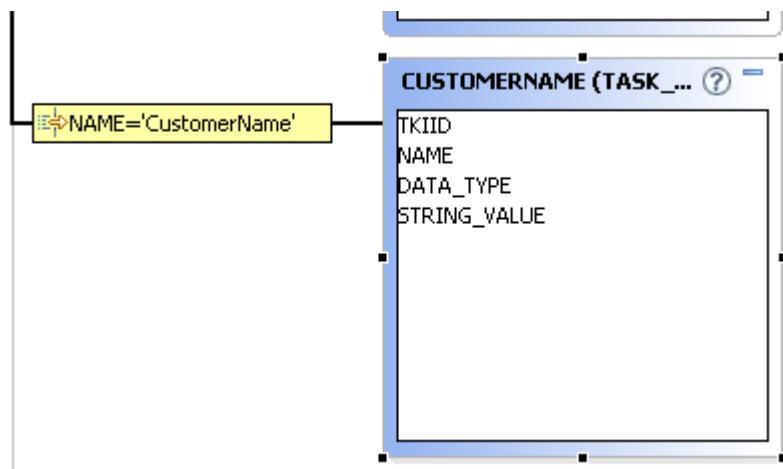
## Add two TASK\_CPROP tables

In our sample, we have two properties associated with a task (Customer Name and Balance). We want these included in the BANK.ACOUNT query table. Drag two instances of the TASK\_CPROP table into the canvas. One will be used for the Customer Name property and the other used for the Balance property.



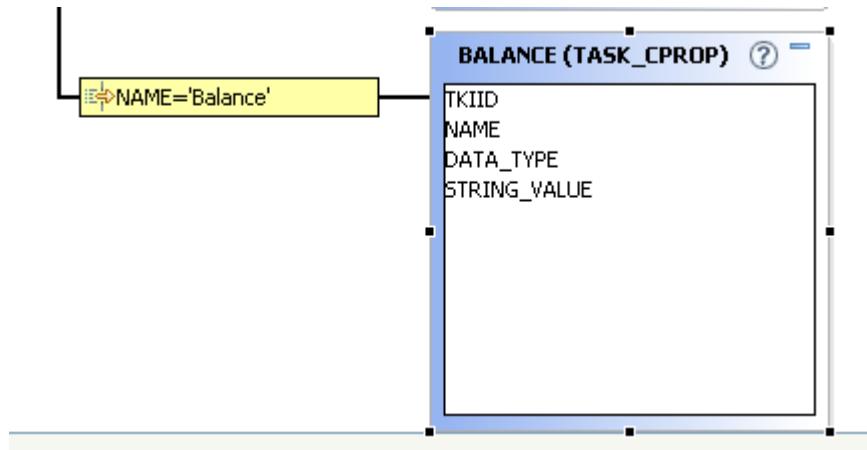
## Configure the Customer Name

Rename the table's short name to be CUSTOMERNAME. Although not required, this aids in readability. In the filter expression used to select the values, change the filter so that NAME matches 'CustomerName' which is the name of the property.



## Configure the Balance

Similar to the previous step, we rename the short name of the table to be BALANCE and set the filter used to select values to be 'Balance' which is the name of the property.



Add CUSTOMERNAME and BALANCE columns to the BANK.ACCOUNT table

Drag and drop the field called STRING\_VALUE from both the TASK\_CPROP tables into the BANK.ACCOUNT description. This will now include this data in the resulting query.

| TE  | CUSTOMERNAME | BALANCE      | CO |
|-----|--------------|--------------|----|
|     | CUSTOMERNAME | BALANCE      | TA |
| TE  | STRING_VALUE | STRING_VALUE | CO |
| ber | String       | String       | ID |

Examine the completed BANK.ACCOUNT query table

The final BANK.ACCOUNT query table looks as follows. It is a combination of data from a number of different sources and includes the additional data that we want to display.

| Name:        | BANK.ACCOUNT |       |       |       |              |              |                    |                    |           |  |
|--------------|--------------|-------|-------|-------|--------------|--------------|--------------------|--------------------|-----------|--|
| Query Table: | DESCRIPTION  | OWNER | KIND  | STATE | CUSTOMERNAME | BALANCE      | CONTAINMENT_CTX_ID | TKTID              | WT.REASON |  |
| Attribute:   | TAD          |       | TA    | TA    | CUSTOMERNAME |              | TA                 |                    | TA        |  |
| Type:        | DESCRIPTION  |       | OWNER | KIND  | STATE        | STRING_VALUE | STRING_VALUE       | CONTAINMENT_CTX_ID | TKTID     |  |

Click on the white space in the diagram. In the Properties view, click on the Description tab. Enter Bank Account in the Display Name.

This will give a human readable name to the Query Table when we select it in Business Space at a later step.

| Language | Language Description | Display Name | Description |
|----------|----------------------|--------------|-------------|
| default  |                      | Bank Account |             |
|          |                      |              |             |

Repeat this for the columns

When the Tasks List widget displays the data, it uses the column names to display the heading for those columns. These aren't very readable. For each of the columns we want to display, select that column and change its Display Name to be a better value. The following columns should be modified.

Description

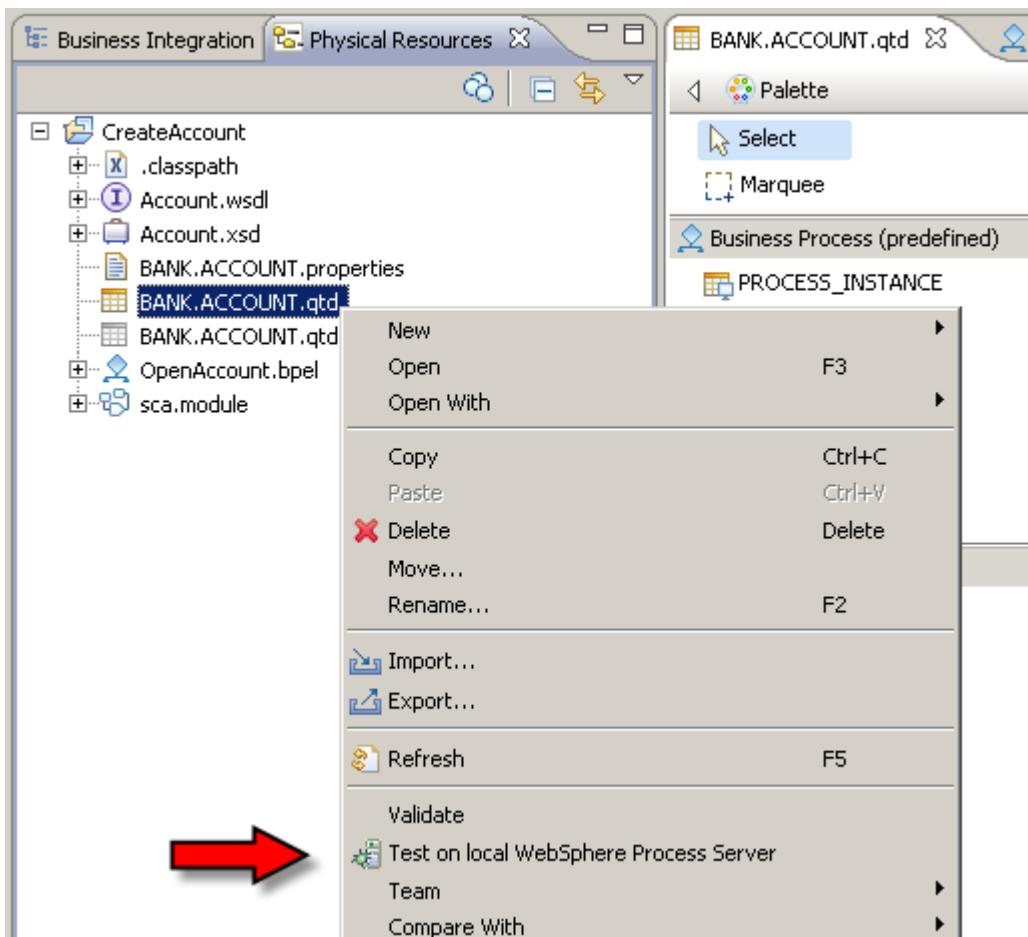
Owner

Customer Name

Balance

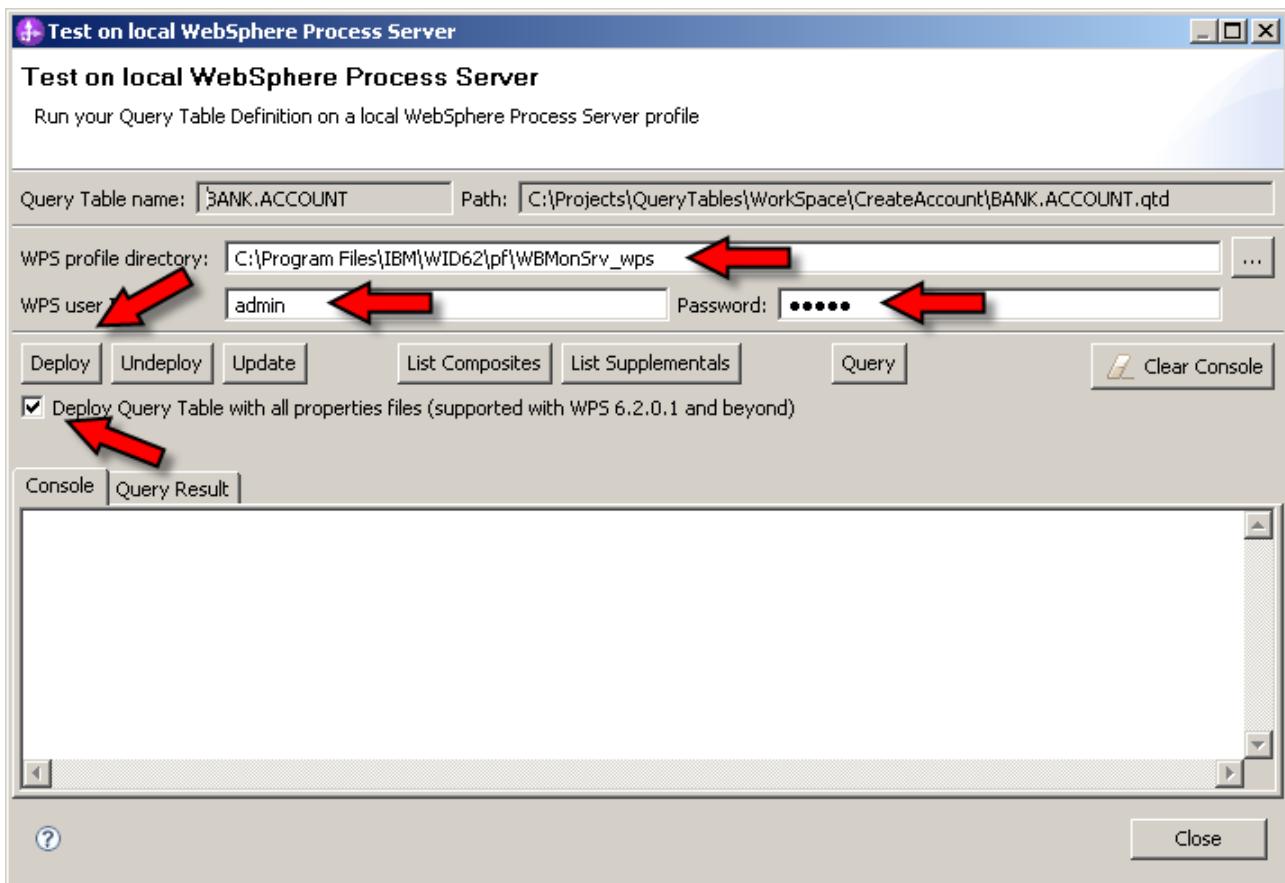
Open up the Query Table testing tool

Save the query table that we have just built and then right click upon it to open up the context menu. From the context menu, select “Test on local WebSphere Process Server”.



Complete the details for deployment

The testing tool allows us to deploy the query table definition to the running WPS server. Complete the form and then Deploy. Make sure that the check box to deploy the additional properties files is also selected.



Open and configure the Tasks List widget

In Business Space, add a Tasks List widget to the page. Open the configuration panel for the widget.

| Name        | Description        | Owner | Due time             | Start date |
|-------------|--------------------|-------|----------------------|------------|
| OpenAccount | Open a new account |       | Jul 21, 2009 2:36:11 |            |

View the available query tables

In the Content tab, notice that there is now a list of query tables. If we pull down the task list to display, we see that a new one is present called "Bank Account". Here we can select which columns and their order that should be displayed in the Tasks List.

**Tasks List**

Content Filter Display Refresh Priority

Select task list to display

All tasks

Bank Account

Add task lists... ▾

Name Description Owner Due time Start date Priority

Check: All | None

### Change the view of the Tasks List

In the Tasks List widget, pull down the view (currently set to All tasks) and select “Bank Account”. The widget will then change to show the Bank Account view.

**Tasks List**

All tasks ▾ Available Tasks ▾

All tasks

Name Description Owner Due time Start date

OpenAccount Open a new account Jul 21, 2009 2:36:1

### Review the result.

Now we see the final result. It is a list of tasks with the data for those tasks taken from the query table that we just built.

**Tasks List**

Bank Account ▾ Available Tasks ▾ Actions ▾

| Description        | Owner      | KIND      | STATE      | Customer Name | Balance |
|--------------------|------------|-----------|------------|---------------|---------|
| Open a new account | To-do task | Available | John Smith | 349.88        |         |

## Business Objects

Business Objects are the data abstractions used in WPS. When data is received or sent between components or services, if the data is anything other than simple data, it is treated as a Business Object. Think of the Business Object as a logical container of fields where each field has a name

and a data type. The data type can be a simple data type such as String or Integer or can itself be another Business Object.

## ***SDO Programming Model v1.0***

Business Objects are SDO. The in memory representation of a business object is a 1.0 Service Data Object. The programming model is the SDO 1.0 programming model as a base with the Business Object Framework Services providing incremental services in addition to what is available in SDO 1.0. It is recommended that you leverage the 1.0 Java Doc for reference. It is available for download from developerWorks here:

- [Commonj-SDO-Specification](#) (pdf)
- [SDO API documentation](#) (JavaDoc)

## ***SDO Programming Model v2.0***

The WPS Business Object Framework is not based on this version but you can access the published 2.0 doc at this address.

- [SDO API documentation](#) (JavaDoc)

## ***Recording changes***

One of the primary functions of SDO has been for the 'disconnected data source' pattern. This model is the idea that a data owning system can provide a copy of an instance of data that it owns and that data is contained within the SDO. The application that requested the data can now work against the SDO copy of the data. At some time in the future, the SDO DataObject can be supplied back to the data owning system and, if possible, only the changed data items need to be updated by that system. This disconnected data model allows a consumer of data to get a copy, be disconnected from the originating data system, do work against that data and then have the resulting changes pushed back into the data owning system. SDO attempted to provide this function through a mechanism it called a DataGraph. The reality of the situation after years of SDO availability and practice is that this capability is never used.

## ***Splitting namespaces across projects***

Be very very careful when splitting namespaces across projects. The following won't work (by design)

MyLibrary contains BO1 in namespace <http://com.test.types>

MyModule contains Iface1 and BO2 in namespace <http://com.test.types>

BO1 will not be visible to Iface1 or BO2 (or anything else in MyModule with namespace <http://com.test.types>). From the editor - you will not even be able to create an attribute in BO2 of type BO1. Nor an input/output parameter of type BO1 in Iface1.

Best practice: do not split namespaces across projects.

What you should be aware of: This applies to null namespaces.

## ***Business Object Programming***

This section describes some of the issues and concepts relating to Business Object programming in Java.

### SDO Representation of XML Schema Types

#### SDO Representation of XML Schema Types

##### Arrays

When a Business Object field is flagged as an Array, the DataObject getter and setter is getList() and setList() which interact with the java.util.List class.

Note that getList() will always return a List object, never null. The returned List can track changes to the Business Object (it is not just an ArrayList or LinkedList), which is important for Business Objects that use an internal Sequence (see below).

Similarly setList() does not replace the internal List, it replaces the contents (according to the SDO 1.0 spec, it is equivalent to clear() followed by addAll()).

The List interface has a size() method that can be used to determine the number of elements in the list (and hence the number of elements in the array).

### **Sequenced Objects**

When a Business Object represents complex content (for example, nested, repeating sequences in XML Schema), the object maintains an internal sequence. As properties are set, or values added to Array properties (Java List interface), the internal sequence is updated automatically. For these complex cases, it is the responsibility of the programmer to set / add values in an order that matches the schema.

### **Nested BOs**

BOs typically can contain other BOs as fields resulting in a tree or hierarchical structure. To create a child BO, one could use the BOFactory and create an appropriately typed BO and then add it but there is an easier way. The DataObject contains a method called createDataObject that takes a parameter a property name and returns a DataObject of the appropriate type for the named property contained in the parent DataObject.

For example if a BO called "A" contains:

- Name Type
- LastName String
- Age int
- Billing Address
- OrderValue double

then if we have a reference to an "A" DataObject calling the A.createDataObject("Billing") method will return a DataObject of type Address.

BOFactory: createByElement or create?

In WPS 6.0.1+ there are two methods on the BOFactory you will get to know well. They are createByElement and create. Use create when the Business Object xsd definition is contained in a <xsd:complexType> tag. Use createByElement when the Business Object xsd definition is contained in a <xsd:element> tag. If you created your Business Object in WID, then it generated an xsd that via the complexType tag and you will use the create(..) method.

BOs of 'any' type

Firstly, there are (at least) 3 cases to be aware of:

- BOs using inheritance (or extension, in schema terminology)
- BOs using xsd:anyType (a named element, but unspecified type)
- BOs using xsd:any (any sequence of mixed content)

In the first case, a BO can specify the base type for a property, and at runtime a BO inheriting from the base can be used. This is supported in the runtime and the BO editor.

In the second case, the corresponding property will have a type of Object (not DataObject) at runtime. The BO editor will let you specify this, however it could cause problems at runtime (where components assume that they will get a DataObject, not an Object).

In the third case, this is a schema feature and shouldn't be used in BOs, but may occur where an external schema is imported (e.g. Web Services). SDO supports this for parsing XML / reading from the DataObject, but not when creating a DataObject. See technote 1250444 'Problems associated with using xsd:any' for a workaround.

WSDL files and externally referenced XSDs

WID

When dealing with remote business objects, the current situation (PMR 90542-999-000) requires you to import the XSDs into WID. You can do this by

\* File > Import... > HTTP & then specify the source & the target locations.

By importing the XSDs, the other editors are able to use them.

WPS

If you currently want to reference a "remote BO," defined as one which has the schemaLocation attribute pointing to a URI that contains the XSD, the runtime will load it. If you cannot load your "remote" BO, check that the schemaLocation attribute is set properly (for example, you are able to load get the XSD from the URI you specified) and then open a PMR. In some future release of WPS, there may be truly "remote" BOs in which you do not need to hardcode the schema location.

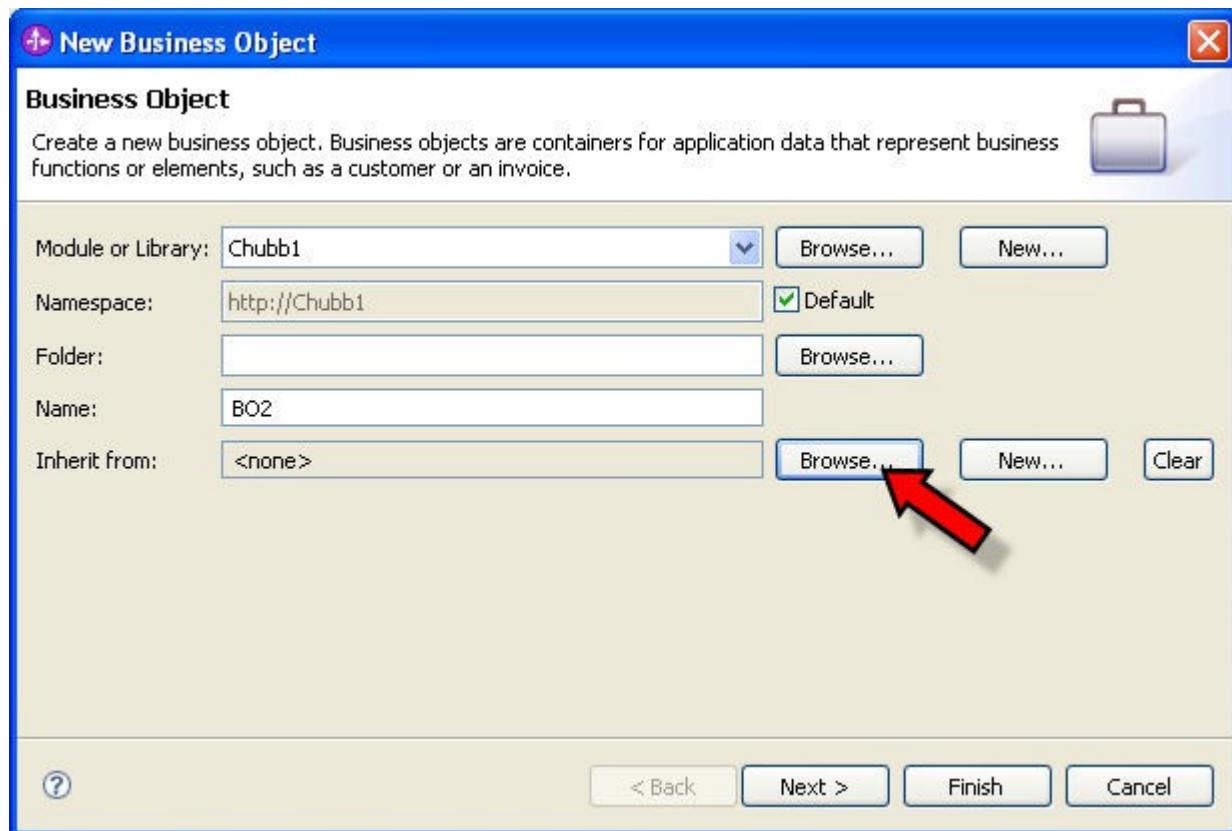
## OpenSource SDO Utilities and Classes

As we build SDO based projects, we will find that we generate code artifacts that do interesting things (such as dump the contents of a DataObject). To capture such artifacts, an Open Source (but internal to IBM) repository has been created that is built on a CVS server. This allows you to immediately import useful function directly into WID. You can also update the code or add new assets and post them back to the repository. There are more details available on the OpenSource repository.

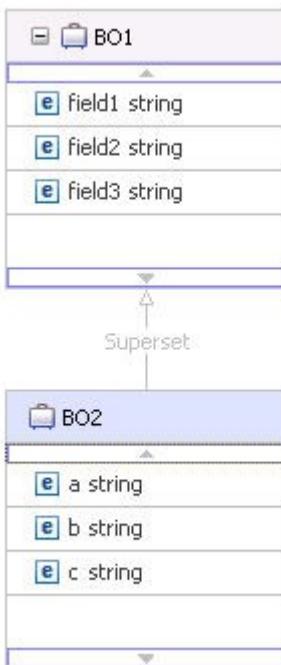
## ***Business Object Inheritance***

A Business Object can inherit from another business object when it is defined. This allows one BO to be a superset of another. The superset BO contains all of the fields of the subset BO plus additional fields. In an interface, a subset BO specified as a parameter can be passed a superset BO as an actual parameter.

The following shows the creation of a new Business Object called BO2. We want this to inherit from BO1 and select BO1 in the "Inherit from" area.



The result is a new BO called BO2 that is a superset of BO1.



## ***Application Specific Information (ASI)***

Business Objects can have meta data associated with them and the fields contained within them. This meta data is called "Application Specific Information" or "ASI". Unfortunately, there does not appear to be any SDO API to retrieve (or set) this information. Fortunately IBM has provided a class called `AdapterBOUTil` that is supplied with the Adapter Foundation Class (`CWYBS_AdapterFoundation.jar`). There are a number of methods for working with the ASI information including:

- `getMetaDataForObject()`
- `getMetaDataForProperty()`
- `getMetaDataForOperation()`

See Also:

DeveloperWorks – [Validating business objects in Websphere Process Server](#) – 2009-07-29

## ***SDO Programming***

The Service Data Object (SDO) has a rich set of programming APIs. JavaDoc for the APIs (v2.4.2) is available as well as more general documentation.

This section shows some illustrative techniques when working with the SDO API.

### **Walking a DataObject**

Walking or iterating through a DataObject can be achieved through the following code fragment.

```

DataObject myDO;
...
Type myType = myDO.getType();

```

```

List propertiesList = myType.getProperties();
Iterator i = propertiesList.iterator();
while(i.hasNext())
{
 Property currentProperty = (Property)i.next();
 ...
 Object value = myDo.get(currentProperty);
}

```

The code works by retrieving the Type of the DataObject. The type contains details of all the different properties contained within. A Java list of those properties is obtained. Next, the list is iterated and something done with each of the properties within.

## Create and Populate a DataObject

This method creates a DataObject of the named type. An optional (specify null if not needed) map is supplied containing name/value pairs. For each name, a field in the DataObject is expected to exist that can be set to the supplied value.

```

public DataObject createDataObjectWithContent(
 String namespace,
 String name,
 HashMap<String, Object> map) {

 BOFactory boFactory = (BOFactory) ServiceManager.INSTANCE
 .locateService("com/ibm/websphere/bo/BOFactory");
 DataObject dataObj = boFactory.create(namespace, name);
 if (dataObj == null) {
 return null;
 }
 if (map == null)
 {
 return dataObj;
 }
 Set<String> keySet = map.keySet();
 Iterator<String> i = keySet.iterator();
 while (i.hasNext()) {
 String key = i.next();
 Object value = map.get(key);
 dataObj.set(key, value);
 }
 return dataObj;
}

```

## IBM DataObjectUtils

IBM provides a class called `com.ibm.bpc.clientcore.DataObjectUtils` which parses a DataObject and populates a HashMap with the content of that DataObject. The keys to the hashmap are the XPath strings that would be used to access a field. The values of map are the contents of the object. In addition to parsing an object, the map can also be re-written into the object.

This class was designed for JSF usage and can be located in the JAR:

`<WPS>/ProcessChoreographer/client/bpcjsfcomponents.jar`

Testing seems to show that the class only works in a JSF environment which is a pity because it seems generically useful.

## **SCA Business Objects**

### **Creating an SCA Business Object**

A Business Object represents a structured piece of data. The Business Object can be created in an SCA application through a Java class called the BOFactory. This factory must be retrieved from a named SCA location:

```
BOFactory boFactory =
(BOFactory) ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory");
```

The BOFactory is part of the `com.ibm.websphere.bo` package and is fully documented along with the other related classes. At the simplest level, to create a new Business Object, we can use:

```
DataObject dataObj = boFactory.create("targetNameSpace", "complexTypeName");
```

As you will notice, the DataObject is created from the factory through the combination of target namespace and type name. The BOFactory appears to scan its current class path looking for XML or XSD documents that match these requirements and, from these values, then creates the associated DataObject. What this means is that if you need to create a DataObject in an arbitrary class, then the BO (.xsd file describing it) needs to be included as a dependency in the project in which the calling class is contained.

Here is an example of creating a DataObject. If you have a WPS Library called "Lib" that contains a BO definition called "Customer" in namespace "`http://mybiz`", then if you want to create a DataObject instance of that type you would need to call:

```
DataObject myObject = boFactory.create("http://mybiz", "Customer");
```

If the code that creates the BO is in a WPS Module, then "Lib" must be flagged on that Module as a dependency.

If the code that creates the BO is a J2EE App, then the JAR file representing "Lib" must be added as a dependency to the J2EE Deployment Descriptor.

### **Converting to/from XML**

WPS provides a utility class to convert a DataObject to/from a specific formatted XML document. The class implementing this function is called BOXMLSerializer. An instance of this class can be retrieved through:

```
BOXMLSerializer mySerializer =
(BOXMLSerializer)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
```

Some of the methods of this class expect a `rootElementName`. To determine the root of a current DataObject, use the following:

```
dataObject.getType().getName()
```

Some of the methods of this call expect a namespace. To determine the namespace of the current DataObject, use the following:

```
dataObject.getType().getURI();
```

The following example illustrates turning a BO into an XML document:

```
BOXMLSerializer mySerializer =
(BOXMLSerializer)
```

```

ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
String rootElementName = dataObject.getType().getName();
String targetNamespace = dataObject.getType().getURI();
ByteArrayOutputStream baos = new ByteArrayOutputStream();
mySerializer.writeDataObject(dataObject,
 targetNamespace, rootElementName, baos);
String xmlText = new String(baos.toString());

```

To convert an XML string to a DataObject, the following code may be utilized:

```

String xmlText = "<XML to become a BO>";
...
BOXMLSerializer mySerializer =
(BOXMLSerializer)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
ByteArrayInputStream bais = new ByteArrayInputStream(xmlText.getBytes());
BOXMLDocument document = mySerializer.readXMLDocument(bais);
DataObject dataObject = document.getDataObject();

```

## **XML documents with no namespace**

It is common to find XML documents that have no namespace information associated with them. In order to be able to support this kind of document, the corresponding Business Object should also be set to have no namespace associated with it.

When WPS is supplied an XML document to be turned into a Business Object, it examines that document and provides a fully qualified name of the format {Namespace}Name. This is used to then find the corresponding Business Object definition.

When an XML document with no namespace is presented, then the result is {}Name. In order to find a matching Business Object definition, the BO must also appear as {}Name and this is achieved through nulling out the namespace for the BO. This will result in a WID based warning but this may be safely ignored.

Here is an example. Consider a Business Object called BO1 (no namespace) that has three fields. Converting this to its XML representation results in:

```

<?xml version="1.0" encoding="UTF-8"?>
<BO1 xsi:type="BO1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <f1>1</f1>
 <f2>2</f2>
 <f3>3</f3>
</BO1>

```

## **SCA**

### ***Asynchronous processing***

#### **See Also**

- DeveloperWorks – [Asynchronous processing in WebSphere Process Server](#) – 2009-04-29
- DeveloperWorks – [WebSphere Process Server invocation styles](#) – 2008-10-29

### ***SCA Import and Export protocol bindings***

When we spoke about SCA Import and SCA Export components, we found that there are protocol bindings associated with them. These protocol bindings describe how communication inbound (for exports) and outbound (for imports) are achieved. There are a variety of options for protocol bindings. The following sections drill down into each one. The choice include:

- Web Services
- JMS
- MQ
- EJB
- <others>

## **Web Services**

### **SOAP/HTTP**

The SCA bindings for SOAP/HTTP leverage the underlying WAS Web services capabilities.

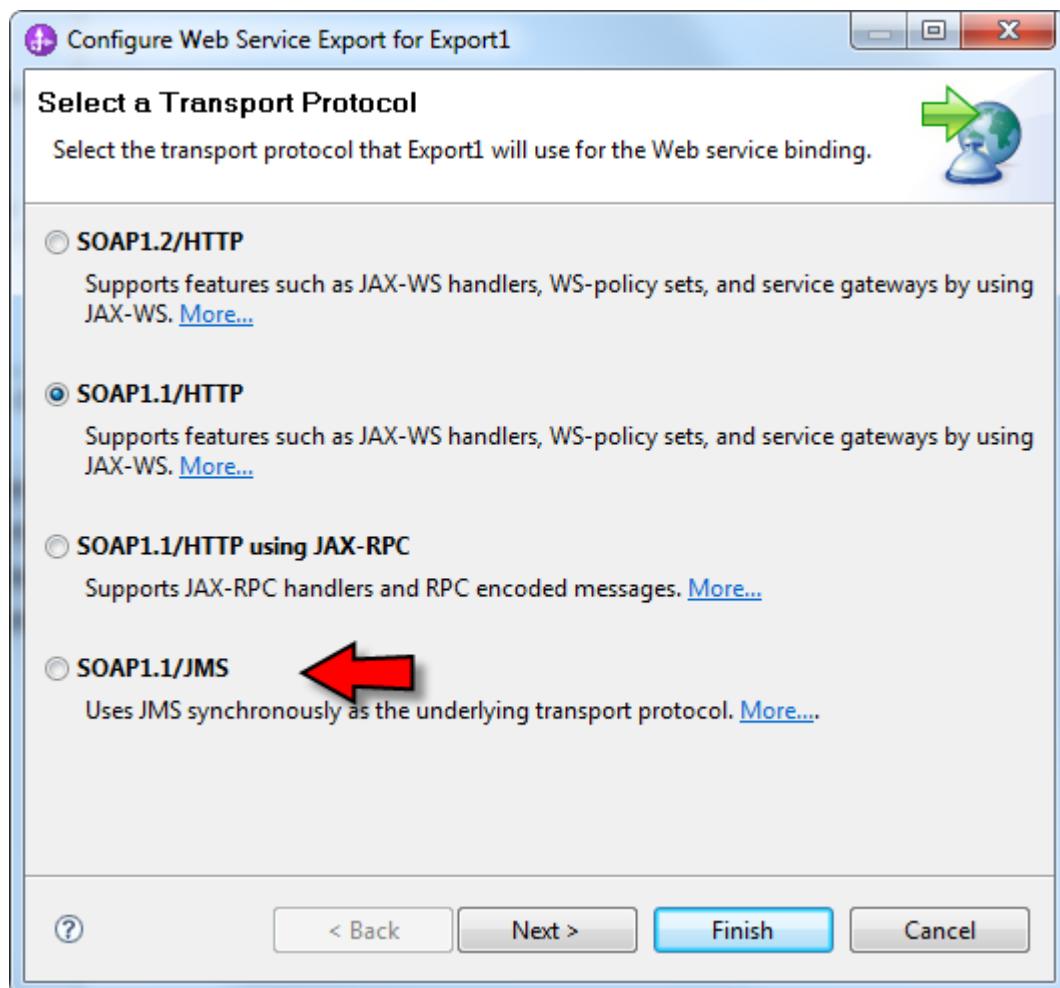
### **SOAP/JMS**

SOAP over JMS is the idea that a service caller and service provider will interact with each other by the service caller building a SOAP request message and depositing that message on a JMS queue. The service provider will be monitoring that queue and when a message arrives on it, the provider will retrieve the message and process it sending back a response as needed. The usage of JMS as opposed to the much more common HTTP protocol is a consumer's choice.

To illustrate how SOAP/JMS works, we will start with a simple story. Consider the following simple SCA module that shows an unbound SCA Export connected to a component. The component will simply log the fact that it has been called. The interface is a simple one-way operation was a simple Business Object as input.

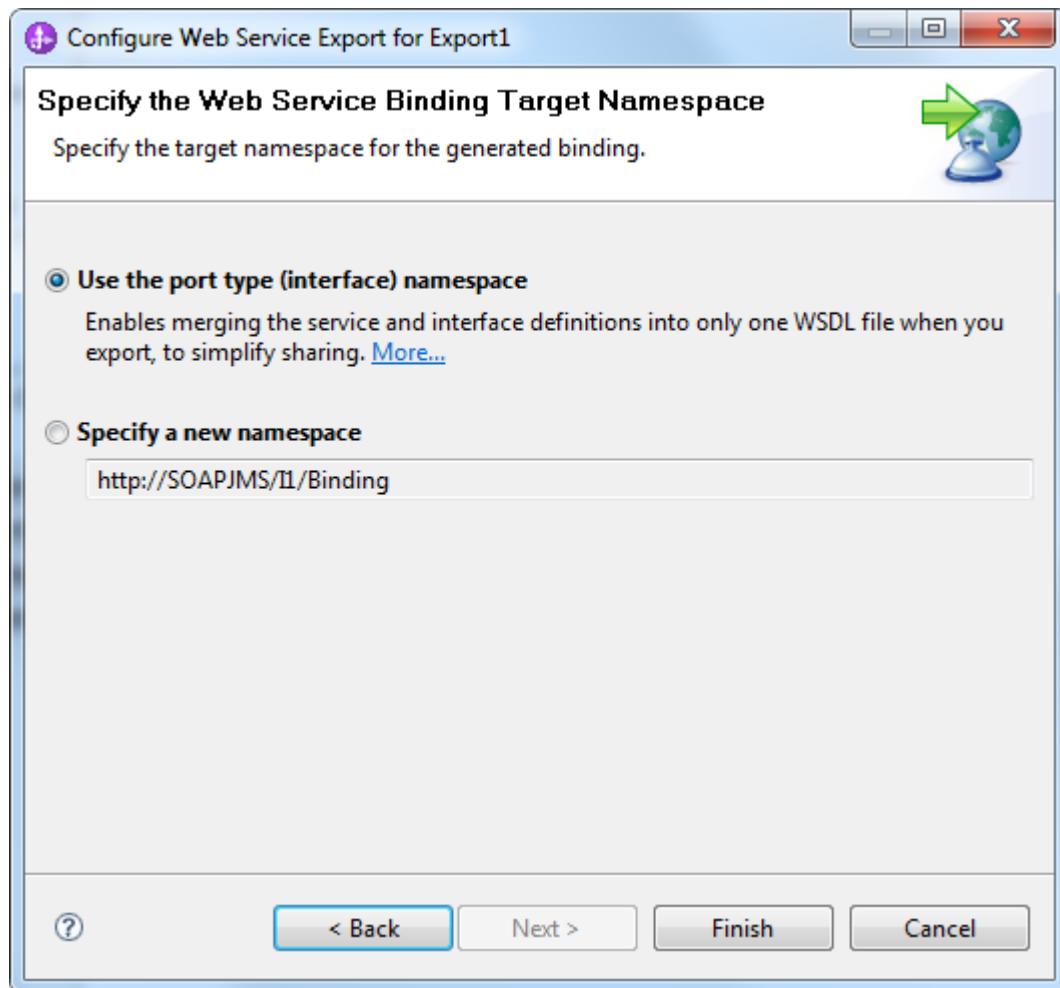


When we select the generate bindings for Export1 and then Select Web Services Binding as shown next:



We have the opportunity to build a SOAP/JMS inbound listener.

The next page of the wizard looks as follows:



As we can see, this is a pretty sparse wizard. When the binding has been built, the result looks like a standard Web Services binding:



However, if we look at the binding details we find that the transport type is defined as SOAP/JMS.

The screenshot shows the 'Properties' dialog for 'Export: Export1 (Web Service Binding)'. The 'Binding' tab is selected. A red arrow points to the 'Transport' field, which contains 'SOAP1.1/JMS'. Other fields include 'Address' (jms:/queue?destination=jms/Export1&connectionFactory=jms/Export1QCF&targetService=Export1\_I1JmsPort), 'Port' (Export1\_I1JmsPort), 'Service' (Export1\_I1JmsService), and 'Namespace' (http://SOAPJMS/I1).

With SOAP/HTTP transport of Web Services, the Address corresponds to a standard URL, however for SOAP/JMS, the format is different. Examining the format, we find it to be:

```
jms:/queue?destination=<jndi queue>&connectionFactory=<jndi connectionFactory>&targetService=<service port>
```

We see that there are two properties that pertain directly to JMS. One is the JNDI name of a connection factory for forming connections to the JMS provider and the other is a JNDI name for a JMS queue resource that will be monitored for incoming SOAP messages.

Experimentation has thrown up some useful nuggets of information:

- The connectionFactory entry must be a queue connection factory and **not** a generic JMS connection factory.

By using an Import component and writing a test message to a JMS queue, we can see what an example JMS message contains. We find that the JMS message type is a BytesMessage and a typical content is as shown next:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header/>
 <soapenv:Body>
 <i1:operation1 xmlns:i1="http://SOAPJMS/I1">
 <input1>
 <f1>1</f1>
 <f2>2</f2>
 <f3>3</f3>
 </input1>
 </i1:operation1>
 </soapenv:Body>
</soapenv:Envelope>
```

Looking at the JMS headers on the message, we find a header called targetService. It appears that this is used to correlate the message with the service associated with that message.

For an Export component, it appears that the JNDI connectionFactory and JNDI queue name resources are created automatically as part of the application deployment. This is rather unusual as other bindings for SCA components seem to offer us the choice of binding to administratively defined resources as opposed to the deployment creating them for us.

For example, if we deploy a module with an SCA Export with SOAP/JMS bindings, the following resources were found to have been automatically created:

- Activation spec: jms/Export1AS against bus SCA.System.\*.Bus and queue jms/Export1
- Queue: jms/Export1 against bus SCA.System.\*.Bus
- Queue Connection Factory: jms/Export1QCF against bus SCA.System.\*.Bus

Notice that we are going against the SCA System Bus. It is not clear if we can change these resource definitions and have them made permanent.

See also:

- DeveloperWorks - [Develop a SOAP/JMS JAX-WS Web services application with WebSphere Application Server V7 and Rational Application Developer V7.5](#) – 2009-03-18
- DeveloperWorks - [Building a JMS Web service using SOAP over JMS and WebSphere Studio](#) – 2004-02-01

- W3 – [SOAP over JMS Working Group](#)
- W3 - [SOAP over Java Message Service 1.0](#) (Spec)

## JMS

SCA modules support JMS for both inbound and outbound requests. A JMS Import looks as follows on the SCA assembly diagram:



If the interface is defined as one-way, then the message will be put to the queue and that will be the end of the story. However, if the interface is defined as two-way, then the message will be put to the queue and then wait for a response on the reply queue.

When configuring a JMS Import, we provide settings for the JMS Connection factory, the JMS Send Destination and for two-way requests, the Receive Destination.

When using a JMS Import being called by a mediation flow and the mediation flow expects a reply, make sure that the Asynchronous invocation quality is set to "Call" as opposed to the default of "Commit". When set to "Call", the mediation flow work is committed before the send of the message.

### See Also

- DeveloperWorks - [How to use additional JMS providers with WebSphere Process Server and WebSphere Enterprise Service Bus Version 6.02](#) - 2007-02-28

## HTTP

### See Also

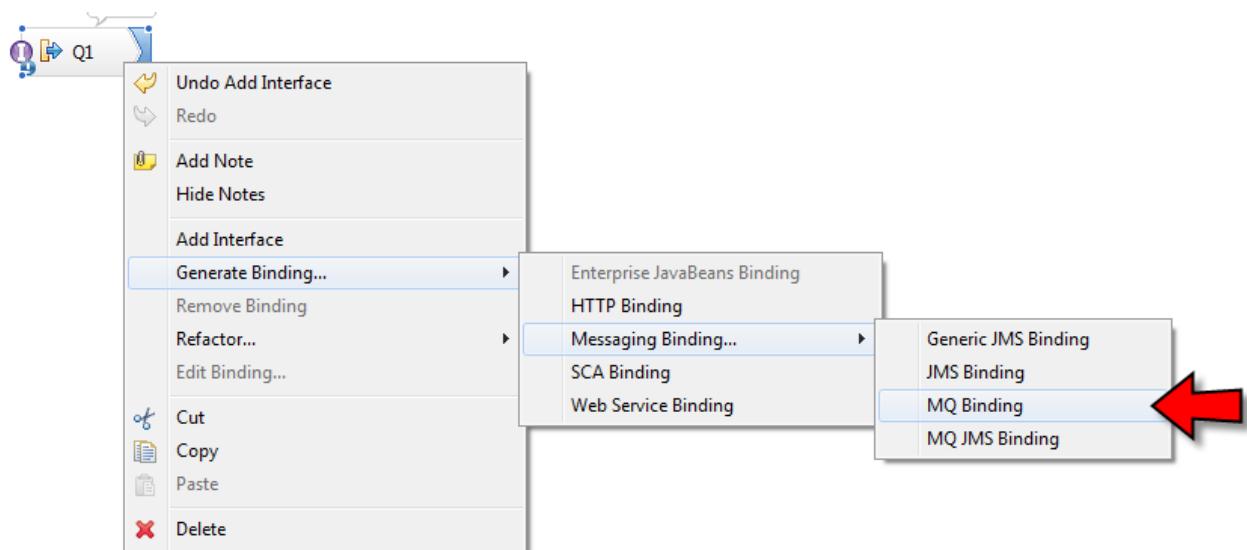
- DeveloperWorks - [Integrating RESTful web services into a business application in IBM Business Process Manager](#) - 2013-12-11
- DeveloperWorks - [Implementing your WebSphere Process Server 6.1 business module the RESTful way](#) - 2009-07-15
- DeveloperWorks - [Developing a PHP client using the REST API in WebSphere Process Server 6.2](#) - 2009-04-01
- DeveloperWorks - [Configuring SCA HTTP binding to enable real-life scenarios](#) - 2008-03-26

## MQ

SCA supports WebSphere MQ as a source and target of messages. This means that we can have an SCA Export bound to an MQ queue such that when a message arrives on that queue, an SCA module will be triggered. Conversely, an SCA Import can be bound to an MQ queue such that when the Import is reached, a new message is delivered to a queue.

The message sent via an Import has to be serialized into a physical data stream in order to be placed on the queue. For an Export, the incoming message has to be parsed from its physical format and a Business Object built.

When working with an Import component, the MQ bindings can be selected from the list of available bindings:



The selection of a binding for an Import brings up a wizard page show next:

**Configure WebSphere MQ Import Service**

### Specify the Configuration Properties

The following configuration properties are required to deploy and run a WebSphere MQ import service.

**End-Point Configuration**

Specify a configuration view option:

Specify the properties to use to configure WebSphere MQ resources  
 Specify the JNDI name for the pre-configured WebSphere MQ resources

JNDI name for MQ connection factory:

JNDI name for send queue:

Request queue manager: \*

Send queue: \*

Transport:

CCDT file:

Host name: \*

Server channel: \*

Port: \*

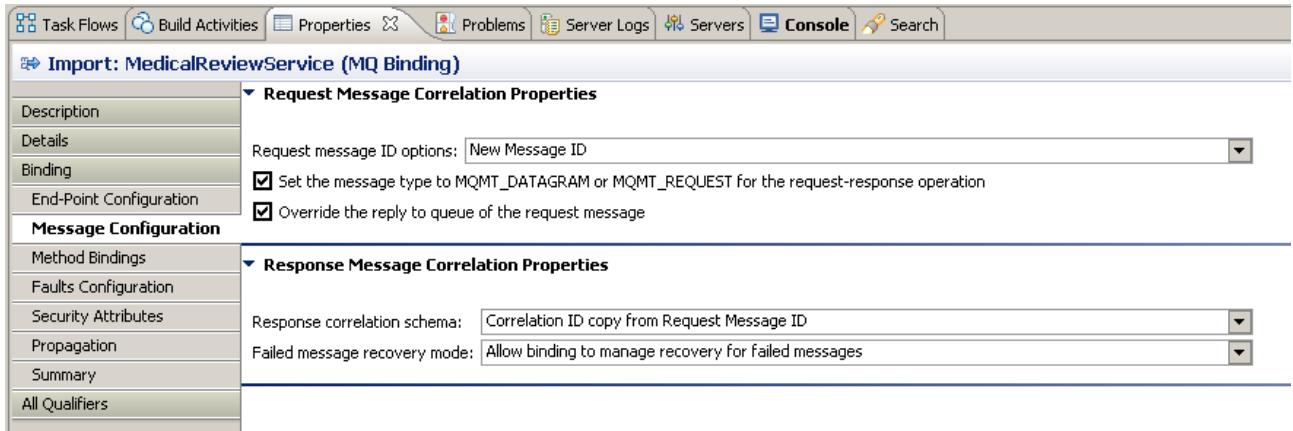
**Data Format**

Default request data format:

In order to connect to an MQ queue manager we must supply some connection details. SCA can connect to a queue manager using either *client* or *bindings* mode (these are MQ terms). In client mode, a network connection is made to a remote queue manager. In bindings mode, the queue manager must be co-located on the same machine as IBPM.

## **MQ Correlation messages**

When sending a request MQ message through an SCA Import component and a response is expected it is common to use an MQ correlation ID. This allows the request and response messages to be correlated together. In the Properties > Binding > Message Configuration section of an MQ bound SCA Import, there are settings that describe how the request message and response message should be related.



There are three choices:

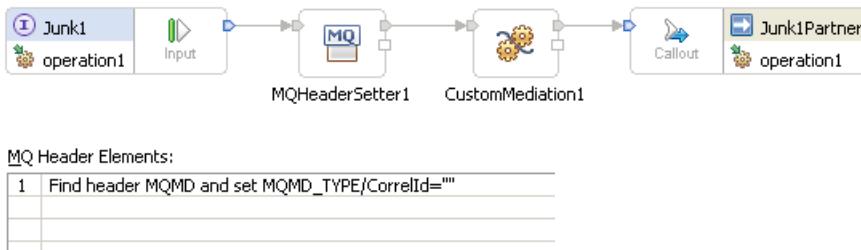
Correlation ID copy from Request Message ID – The matching response message will have a Correlation ID value equal to the Message ID in the request message.

Response Message ID copy from Request Message ID – The matching response message will have a Message ID value equal to the Message ID in the request message.

Correlation ID copy from Request Correlation ID – The matching response message will have a Correlation ID value equal to the Correlation ID in the request message.

When a message is sent by SCA, a new Message ID is generated. For some MQ applications, the Correlation ID in the request message is used for the matching response message. This can pose a problem as the default Correlation ID sent by an SCA Import is a value of all zeros ... which is useless for a correlation value. Although MQ has an option at the MQ layer to generate a unique Correlation ID, that ability is not surfaced in the WPS product. To achieve what we want, we need a different technique.

First, we need to ensure that the SMO has an MQHeader in its structure. We do this by using the MQHeaderSetter primitive. Setting a dummy value in it.



Next comes a CustomMediation which allows us to code some logic in Java. Here is an example

for setting the Correlation ID to a unique value:

```
MQMD myMQMD;
UUID uuid = UUID.randomUUID();
myMQMD = smo.getHeader().getMQHeader().getMd();
ByteArrayOutputStream bos = new ByteArrayOutputStream();
DataOutputStream dos = new DataOutputStream(bos);
try {
 dos.writeLong(uuid.getLeastSignificantBits());
 dos.writeLong(uuid.getMostSignificantBits());
 dos.writeLong((long)0);
 dos.flush();
} catch (Exception e) {
 e.printStackTrace();
}
byte[] data = bos.toByteArray();
myMQMD.setCorrelId(data);
out.fire(smo);
```

Note that MQMD class is `com.ibm.websphere.sca.mq.structures.MQMD`.

What this code does is get the MQMD from the SMO and then turn an instance of a Java UUID into an array of bytes. These bytes are then set to the Correlation ID in the MQMD in the SMO MQHeader. **Note:** It is very important that the data array for the Correlation ID be 24 bytes and not less.

### **Testing MQ bindings**

The IBM Support Pac known as IH03 provides a wealth of MQ testing tools that can be used with great effect for WPS testing. Included in that package is the great RFHUTIL GUI tool for examining messages.

For testing request/response handling, the utility known as mqreply may be used. Its general syntax is not clear from the documentation but the following has been shown to work well:

```
mqreply -f parms.txt -r data.txt -m <queue manager name> -q <request queue name>
```

The `parms.txt` file may be empty.

The `data.txt` is the data to be sent back in n the reply. The request message `replyToQ` will be used.

### **MQ messages and XML**

When a message is put to a queue, the message is placed there as a sequence of bytes. It is not uncommon to place a message on the queue which is a string representation of an XML document. Conversely, we also want the ability to retrieve messages from a queue which are also XML based. It is here that the XML Data Handler comes into play.

See Also:

- Converting to/from XML

See Also

- DeveloperWorks - [Message binding enhancements in WebSphere Process Server V7.0, Part 1: Developing integration components that interact with WebSphere MQ and other messaging providers](#) - 2011-04-13
- DeveloperWorks - [Using WebSphere MQ bindings in WebSphere ESB, Part 1: Manipulating MQ headers in WebSphere ESB using the XSL transformation primitive and WebSphere Integration Developer](#) - 2010-03-03
- DeveloperWorks – [Using the MQ binding plug-in for WebSphere Integration Developer 6.0.2](#) – 2009-10-07

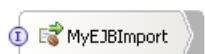
- DeveloperWorks - [Using WebSphere MQ bindings in WebSphere ESB, Part 2: Creating MQ headers using the WebSphere ESB MQ Structures API](#) – 2009-01-21
- DeveloperWorks - [Using WebSphere MQ bindings in WebSphere ESB, Part 3: Using custom WebSphere MQ headers with WebSphere ESB](#) – 2009-01-28
- DeveloperWorks - [Enabling SCA-MQ integration via MQ bindings](#) – 2008-02-06

## EJB SCA Bindings

Within an SCA module we can couple together the relationship between a wide variety of components. This can include stateless session EJBs.

Within the ID tooling, we can open an Assembly Diagram and then drag/drop an EJB onto the canvas which will create an import that is bound to the EJB.

The icon for the import also shows that it is an EJB binding:



The source of the drag for the drag/drop is the EJB icon in the Project Explorer view in the J2EE perspective. This is usually found under:

EJB Projects > EJB Project Name > Deployment Descriptor > Session Beans > Bean Name

Once imported, an interface that can be used within SCA can be found. Unfortunately, the interface generated is a pure Java interface and can not be directly used in the majority of cases because WSDL is the norm.

### ***Auto-generated WSDL to Java Bridge Component***

When dropping the EJB onto the canvas you'll be asked if you want to have a facade map component (Bridge) generated. This will generate a WSDL version of the interface along with a Java component that maps the wsdl operations to java methods.

Currently if you have any overloaded methods on your EJB the generator will not work and you will have to perform the steps as if you were in version 6.0.1.

In 6.0.1 manually construct a WSDL version of the interface and create a Java Component that has the WSDL interface and the Java reference.

One way to achieve this is to manually create an Interface with the same operations as the EJB. The resulting interface will be WSDL typed. Next create a Java Component on the assembly diagram and give the component the newly created WSDL typed interface and also give it a reference of the interface for the EJB. Generate the code for the Java Component and, for each operation contained within the Java code, invoke the EJB operation.

The generated code assists with this technique. If the reference on the Java Component is called XYZReference then a method is created in the Java Code called:

```
public XYZInterface locateService_XYZReference();
```

where the return type is a Java Interface that corresponds to the operations available on the EJB itself.

### See Also

- RedBook – [Developing Enterprise JavaBeans Applications](#) – REDP-4885-00 - 2012-07-31
- DeveloperWorks - [Consuming Java artifacts made easy with WebSphere Process Server and WebSphere Integration Developer](#) - 2007-06-13

- DeveloperWorks - [Using BPEL and EJBs with WebSphere Process Server and WebSphere Integration Developer](#) - 2006-08-09
- DeveloperWorks - [Integrate EJB services with WebSphere Process Server, Part 2: Advanced scenarios for integrating EJBs running on WebSphere Process Server and WebSphere Application Server](#) - 2006-05-03
- DeveloperWorks - [Integrate EJB Services with WebSphere Process Server](#) - 2006-02-02

## **DataHandlers**

For many of the SCA Imports and Exports, incoming data has to be parsed to construct Business Object while outbound data has to be serialized to a stream of bytes. The Import and Export SCA components, during their operation, invoke the services of a logical function called a DataHandler. A DataHandler is responsible for either parsing an input stream to construct a Business Object or serializing a Business Object into a data stream. Because there can be many different physical data structures (XML, comma delimited, fixed width etc), one DataHandler is not sufficient for all uses. To resolve this problem, IBM has created a Java Interface that describes the purpose of a DataHandler. During development, a developer defines which DataHandler to use for the task at hand. A DataHandler is no more and no less than a Java class implemented to conform to an IBM described Java interface that provides the DataHandler contract.

IBM supplies a set of pre-built DataHandlers that apply to many common formats.

| Name                          | Class                                                          |
|-------------------------------|----------------------------------------------------------------|
| Atom feed Data Handler        | unknown                                                        |
| Delimited Data Handler        | com.ibm.wbiserver.datahandler.delimited.DelimitedDataHandler   |
| FixedWidth Data Handler       | com.ibm.wbiserver.datahandler.fixedwidth.FixedWidthDataHandler |
| JSON Data Handler             | com.ibm.wbiserver.datahandler.json.JSONDataHandler             |
| WTX Invoker Data Handler      | com.ibm.wbiserver.datahandler.wtx.WTXInvokerDataHandler        |
| WTX MapSelection Data Handler | com.ibm.wbiserver.datahandler.wtx.WTXMapSelectionDataHandler   |
| XML Data Handler              | com.ibm.wbiserver.datahandler.xml.XMLDataHandler               |

## **XML Data Handler**

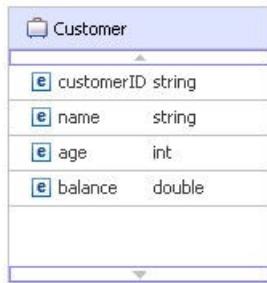
Unlike other DataHandler's, the XML DataHandler holds a special place in SCA and needs a little more explanation than some of the others. Let us start with the basics of an XML DataHandler.

Like other DataHandlers, the XML DataHandler is responsible to taking raw data and creating a Business Object and for taking a Business Object and creating raw data. For the instance of a XML DataHandler, the raw data's encoding is the industry standard known as XML. XML continually seems to hold a special place in the hearts and minds of customers and programmers. Many people make a big thing of data being in XML format and think that is the end of the task. In reality, an interesting thing is found. When an XML document is received by an application, almost invariably the first thing that happens is the parsing of that XML document into a machine internal representation ... whether that is a DOM tree, Java Objects or something else ... the XML physical encoding is removed. Similarly when generating an XML document, the generator usually holds

the data that will eventually become an XML document in some internal format. With this in mind, what we find is that the XML is actually used as an encoding between transmission at one end and reception at another. If the sender and receiver are the same, then XML becomes a common storage encoding for data.

As mentioned, the SCA XML DataHandler is responsible for taking in an XML document and building a Business Object and for taking a Business Object and building an XML document. But here comes an interesting concept. Business Objects, as we know, are created from templates. An instance of a Business Object can't simply contain any fields that it chooses, instead those fields have to conform to the rules laid down by the designer of that Business Object. The Business Object designer uses an ID supplied tool called the Business Object editor to create and edit the Business Object template definitions. We also find that Business Object templates are created for us when we perform tasks such as importing WSDL definitions or running adapter creation wizards. We are comfortable with this model.

If we ask ourselves, what is the nature of a Business Object template, we will quickly find that it is a named container that is able to describe a set of fields.



We also find that the container has a namespace associated with it that allows it to be uniquely identified even if there are other Business Object containers having the same name. If we now ask "Is there an existing encoding language that can be used to describe a container with fields and namespaces?" ... a language that can be used to describe Business Objects ... we find that such a language already exists and there is no need for IBM to create a new one. The name of that language is XML Schema Definition (XSD).

Although originally designed for describing the validity of well formed XML documents, XSD can serve the double duty of being used as a description language for what is to be logically considered a correct instance of a Business Object. With reference to the image of the Customer Business Object above, the following is an example of XSD that is an equivalent encoding of the Business Object:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://Kolban_Module">
 <xsd:complexType name="Customer">
 <xsd:sequence>
 <xsd:element minOccurs="0" name="customerID" type="xsd:string" />
 <xsd:element minOccurs="0" name="name" type="xsd:string" />
 <xsd:element minOccurs="0" name="age" type="xsd:int" />
 <xsd:element minOccurs="0" name="balance" type="xsd:double" />
 </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

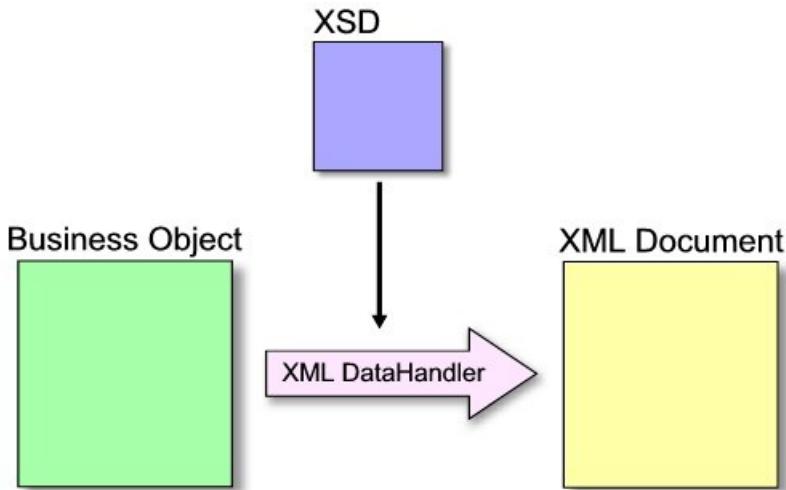
Very quickly we can map the constructs in XSD to the equivalent ideas in the Business Object. Thankfully, the Business Object editor and the rest of WID and WPS usage allows us to hide the "gory" details of the XSD encoding from the nice, high-level, representation of the Business

Object allowing us to consider the Business Object at a much higher and abstract level.

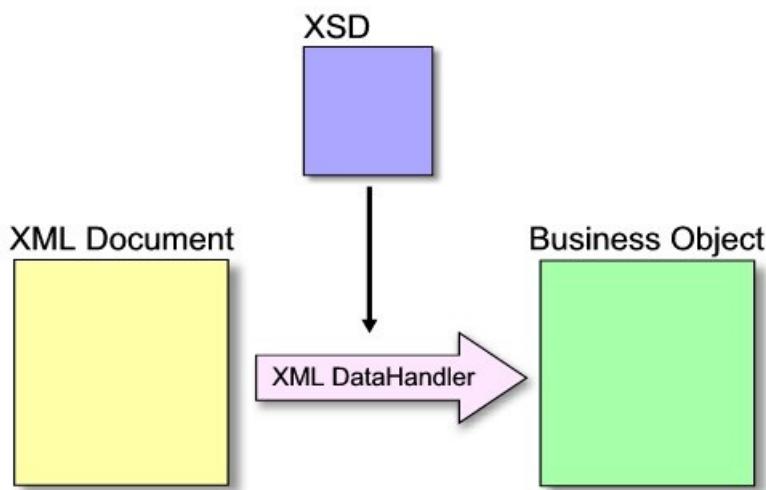
Under the covers, ID and SCA actually use XSD as the encoding/description of Business Objects. In theory, some other encoding could have been used but IBM development leveraged the existing XSD description language to good effect. The fact that XSD is mechanically used to describe the encoding of Business Objects is mostly hidden from the end users through the Business Object editor, BPEL editor, BO mapper and other techniques ... but the fact that XSD still remains under the covers is a truism.

Now let us turn our attention back to the original discussion which is that of an XML DataHandler. Our goal is to turn an XML document into a Business Object and turn a Business Object into an XML document. Since a Business Object's description implementation is encoded via XSD, can this then be used to describe an XML document? The answer to this is yes and this is in fact exactly how the XML DataHandler works. The core principle here is that EVERY valid XML document can be described via an XSD. So if we want to produce or consume a valid XML document, then there has to be an XSD that can be used to describe it. Take care here to realize that this is being qualified by a valid XML. If the data desired to be produced or consumed is not a real XML document but is instead something that may on first glance "look" like XML, it may not be able to be XSD described.

The XML DataHandler can take an instance of a Business Object and using its XSD template of that Business Object can produce an XML encoding of the content of an instance of that Business Object.



Conversely, the XML DataHandler can take an XML document, and using the XSD template for the desired Business Object, create an instance of that Business Object from the XML document.



### **Default XML Data Handler serialization**

Let us now look at default XML Data Handler serialization. We will do this with an example. Consider the following interface. This interface shows a one-way operation that takes a Business Object as a parameter.

**Interface**

| Name          | OneWay                   | Refactor name                                                                |
|---------------|--------------------------|------------------------------------------------------------------------------|
| Namespace     | http://MQTests/OneWay    | Refactor namespace                                                           |
| Binding Style | document literal wrapped | <a href="#">Change binding style to document literal non-wrapped</a> More... |

---

**Operations**

| Name       | Type   |
|------------|--------|
| operation1 |        |
| Inputs     | input1 |
|            | BO1    |

With the following Business Object definition.

**Business object**

| Name      | BO1        | Refactor name      |
|-----------|------------|--------------------|
| Namespace | http://BO1 | Refactor namespace |

---

**Definition**

|                      |
|----------------------|
| BO1                  |
| <Click to filter...> |
| a string             |
| b string             |
| c string             |

If we use the out of the box XML Data Handler, we see the following generated:

```

<?xml version="1.0" encoding="UTF-8"?>
<way:operation1 xmlns:way="http://MQTests/OneWay">
 <input1>
 <a>aVal
 bVal

```

```

<c>cVal</c>
</input1>
</way:operation1>

```

It is obviously XML but where did the encoding come from? The answer is that WPS encoded the request in a form that WPS will be able to decode using the same DataHandler. Notice that the encoding is of the form:

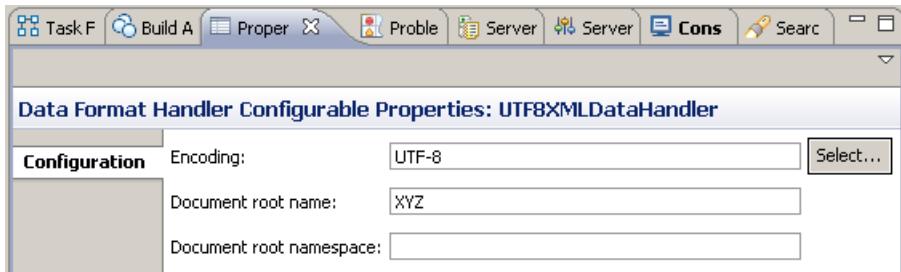
```

<Operation Name>
 <Parameter Name>
 <Parameter values ...>
 <Parameter values ...>
 </Parameter Name>
 <Parameter Name>
 <Parameter values ...>
 <Parameter values ...>
 </Parameter Name>
</Operation Name>

```

Again, this encoding is great assuming that receiver also wants to apply this encoding. But what is the receiver already exists and expects different data?

The properties for the XML Data Handler allow us to provide an alternate Document root name. In this example, we set the value to XYZ. Re-running the previous test, we now find that the output is as follows:



```

<?xml version="1.0" encoding="UTF-8"?>
<XYZ xsi:type="bo1:BO1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:bo1="http://BO1">
 <a>aVal
 bVal
 <c>cVal</c>
</XYZ>

```

As we can see, this is quite radically different. Now we have a general format of:

```

<Document Root>
 <Parameter Values ...>
 <Parameter Values ...>
</Document Root>

```

One special characteristic of note here is the introduction of the `xsi:type` attribute on the document root element. This attribute is key to SCA operation. It allows SCA to know which type of Business Object to create when presented with such an instance of an XML document.

Unfortunately, things change again if there is more than one parameter to the request. Examining the following interface, we see that it now has two formal parameters.

The screenshot shows the BizTalk Configuration interface. The 'Interface' tab is selected, displaying configuration details for a OneWay2 interface. The 'Operations' tab is also visible below it.

|               |                          |                                                                                              |
|---------------|--------------------------|----------------------------------------------------------------------------------------------|
| Name          | OneWay2                  | <a href="#">Refactor name</a>                                                                |
| Namespace     | http://MQTests/OneWay2   | <a href="#">Refactor namespace</a>                                                           |
| Binding Style | document literal wrapped | <a href="#">Change binding style to document literal non-wrapped</a> <a href="#">More...</a> |

Operations and their parameters:

| Name              |        | Type |
|-------------------|--------|------|
| <b>operation1</b> |        |      |
| Inputs            | input1 | BO1  |
|                   | input2 | BO2  |

The generated XML looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<MyRoot xsi:type="way2:operation1_._type"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:way2="http://MQTests/OneWay2">
 <input1>
 <a>a
 b
 <c>c</c>
 </input1>
 <input2>
 <x>x</x>
 <y>y</y>
 <z>z</z>
 </input2>
</MyRoot>
```

As we can see, the parameter names have now been introduced into the output as children of the document root.

### ***XML Documents that are different from Business Objects***

One potential problem with the scheme outlined above is the notion that the XML document provided may not match the Business Object we want to use. Looking again at the Business Object template that we may want to use in our solution:

The screenshot shows a Business Object template for 'Customer'. It includes fields for customerID (string), name (string), age (int), and balance (double).

|                   |
|-------------------|
| Customer          |
| customerID string |
| name string       |
| age int           |
| balance double    |

If we have an XML document that is coming into us looking like:

```
<AccountHolder id='12345'>
 <DateOfBirth>1964-07-11</DateOfBirth>
 <FirstName>Neil</FirstName>
 <LastName>Kolban</LastName>
 <Balance>1234.56</Balance>
 <LastTransaction>2008-07-21</LastTransaction>
</AccountHolder>
```

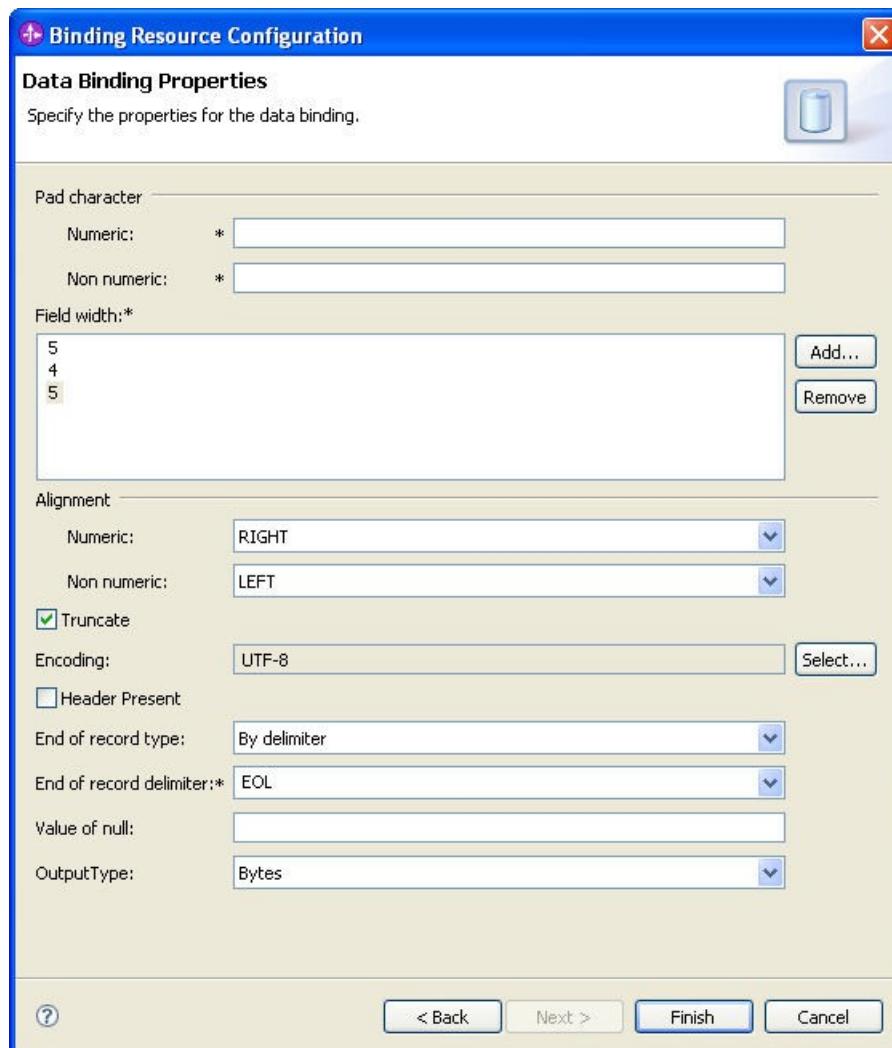
We can quickly see that this doesn't match the Business Object. What we have to do in this case is to find or create a XSD that describes the XML document. When we add this XSD to WID, it "appears" as a new Business Object (because to WID, a BO is described under the covers by an XSD). Now we have two Business Object definitions. One which matches the XML document and one which is the template for the Business Object we want to use in our solution. What we can now do is to employ a Mediation Flow to map from our Business Object to our Data Business Object for generation of XML and map from our Data Business Object to our Business Object for incoming processing of XML.

## Fixed Width DataHandler

IBPM provides a processor for "Fixed Width" data. It is supplied in two forms. One is a DataBinding for MQ or HTTP and the other is a DataHandler implementation.

The first step required is the creation of a DataBinding Resource Configuration.

When the Fixed Width handler is specified, a configuration panel is displayed to input the properties for the parsing and construction.

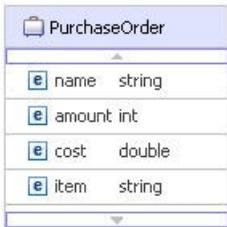


This panel is also later editable when looking at the Binding Resources for the defined Data Binding.

Like other DataBinding/DataHandler functions, the purpose of the FixedWidth processor is to

fold. One part is to take raw physical data as input and populate a Business Object while the second part is to take a Business Object and serialize it to a physical representation. For the Fixed Width handler, the format of the data is expected to conform to the notion of it being a series of fields where each field is of some pre-defined (or fixed) size (or width).

Consider a logical purchase order whose Business Object might look as follows:



There are many ways that this data could be physically represented outside of the context of WPS or WESB but in this article, we care about Fixed Width representation. Fixed Width declares that each field in the data is contained within a fixed width of data.

For example, a Purchase Order that has the following:

Field	Value	Size
name	John Doe	15
amount	3	3
cost	1234.99	8
item	Flat Screen TV	15

May have a physical representation of:

|                   |   |           |  
12345678901234512312345678123456789012345  
John Doe~~~~~3~1234.99Flat Screen TV~

When we look at this format, we can see that there are "rules" that govern its format. These rules have to be configured against the Fixed Width data handler.

- **Pad Character Numeric** - This is the padding character for numeric data. If a field in the Business Object is numeric then this padding can apply. If the data is less than field size then the physical data must be padded with some value. The character value for padding is supplied here. Typical values here would be a space or '0'.
- **Pad Character Non numeric** - This is the padding character for non-numeric data. If a field in the Business Object is not numeric then this padding can apply. If the data is less than field size then the physical data must be padded with some value. The character value for padding is supplied here. A typical here would be a space.
- **Field width** - a list of field widths, one for each field in the data. The number of items here must match exactly the number of items in the data. If these don't match, an exception will be thrown at runtime. For example:

`commonj.connector.runtime.DataHandlerException: The input data has more elements than <x> which is the number of entries in the field width property`

- **Alignment Numeric** - When a BO field is less than size of the fixed width field and is to be padded, this field defines whether the padding will occur on the left or the right. This setting

applies to numeric fields only.

- Alignment Non numeric - When a BO field is less than size of the fixed width field and is to be padded, this field defines whether the padding will occur on the left or the right. This setting applies to non-numeric fields only.
- Truncate - During outbound processing when a Business Object is serialized, the length of the data in a field may be larger than the expected field in the output. If this flag is checked, then the data will be truncated to fit exactly within the field. If the flag is not checked the an exception will be thrown. For example:

commonj.connector.runtime.DataHandlerException: CWLAP0300E: The 16 token is larger than the 15 field width; the token length cannot be truncated.

- Encoding - The incoming textual data may be physically encoded in a variety of character encodings. This field defines the encoding format. For example, IBM-037 is EBCDIC.
- Header Present - The first line of the incoming data may be used to represent a header. The header contains the name of the fields expected. If this field is enabled, a header is generated on output and expected in input. If the field name of the header does not fit within the field width, it is truncated. For our example, the header may be:

```
| | | |
12345678901234512312345678123456789012345
name~~~~~amo~~~costitem~~~~~
```

- End of record type - A single record can be identified either by size or by a delimiter. By size means that the record is exactly the sum of each of the fixed width fields. In our example, the size would be  $15+3+8+15 = 41$  bytes. If the end of record is by delimiter, the delimiter is appended at the end of the record. The choices for this parameter are by delimiter or by size.
- End of record delimiter - If the end of record type, this property defines the end of record delimiter. This can be entered as a string or the special EOL may be used to specify an End Of Line character(s) should be used.
- Value of null - See [TechNote 21306337](#)
- OutputType - This value can be either bytes or characters. This property is on the data binding and not on the data handler. This property is available for MQ and JMS only. Since JMS also has the concept of JMS BytesMessage and JMS TextMessage. If the type is bytes, then a JMS BytesMessage is produced. If the type is characters, a JMS TextMessage is produced. For MQ, if the output type is bytes, the message is written exactly as it is but if characters, the message is encoded into the character set for the MQ message.

## Custom DataHandlers

The DataHandler is a protocol neutral transformation scheme. It can be called by a DataBinding!!

Let us consider the idea of character delimited data as an example. Imagine the physical data "A,B,C". This logically represents three data fields delimited by commas. Now let us think about how this data may arrive at IBPM. It could come in through a JMS queue, an MQ queue, an HTTP connection or read from a flat file. These are only some of the physical deliveries, there could easily be others and even more to come. Each one of these physical deliveries has its own associated DataBinding concept. Each DataBinding implementation concerns itself with converting the physical data to the logical data. But ... and here is the key point ... each DataBinding type is

specific to the physical transport. There is a DataBinding type for MQ, a DataBinding type for JMS, a DataBinding type for HTTP and so on. The DataBinding is responsible for both the physical format of the data as well as responsible for creating/parsing Business Objects. A DataBinding for MQ has to concern itself with MQMD headers, a DataBinding for HTTP has to concern itself with HTTP headers. But if we pause for a second, we realize that there is commonality between all these DataBindings. Each one is still responsible for parsing physical data to construct a BO and constructing physical data from a BO and this is beyond the protocol specific nature.

This is where the concept of the DataHandler comes into play. A DataHandler takes a stream of physical data and converts that to a Business Object and conversely can take a Business Object and convert that into a stream of data. Although this superficially sounds just like a DataBinding, the DataHandler does NOT see any of the protocol specific header or transport information. An instance of a DataHandler is agnostic. What this means is that if we have a DataBinding for a specific protocol, that DataBinding can invoke a protocol neutral DataHandler to perform the core of the transformation work. And this is where the benefit lies ... the same DataHandler can be used by different DataBindings. So a DataHandler that knows how to work with delimited data could be used by an MQ DataBinding, a JMS DataBinding, an HTTP Databining and so on.

The DataHandler technology is part of the `commonj.connector.runtime` story and has a defined Interface called `DataHandler`.

The signature of the Interface looks as follows:

```
package commonj.connector.runtime;
public interface DataHandler extends commonj.connector.runtime.BindingContext
{
 public Object transform(Object source,
 Class targetClass,
 Object options) throws DataHandlerException;

 public void transformInto(Object source,
 Object target,
 Object options) throws DataHandlerException;
}
```

Both of these methods are called to transform data from one format to another. They differ in that the first one is expected to create and return an object while the second is supplied an object that is to be populated. When a DataHandler is called to externalize data from IBPM, the source will usually be a `DataObject`. When IBPM is receiving data from an external source, then the target will usually be a `DataObject`.

The `source` parameter is the source of the data to be transformed. It is commonly one of the following:

```
java.io.StringReader
```

The `targetClass` parameter describes the nature of the target data. It is an instance of this class that should be returned from the function call.

The DataHandlers supplied by IBM can be found in the JAR called `com.ibm.wbicmn.system_version.jar`. These are useful to look at with a decompiler to see how IBM implemented certain functions.

During development, it is a good idea to provide some debugging of the parameters passed into the `transform` and `transformInto` methods. For example:

```
System.out.println("transformInto called: source is " +
source.getClass().toString() + ", target is " + target.getClass().toString() + "
```

```
and options value is " + options);
```

Even if a custom DataHandler has no custom properties of its own, it appears that a properties class as described in the following section is required for it to be seen/used as part of the IBM Integration Designer tooling.

### **DataHandler/DataBinding properties**

But what of the configuration information for the DataHandler? This comes in two parts. One part is the build time and the other is the runtime. At build time, the properties need a way to be entered into the tooling. The description of the available properties is achieved through the creation of a JavaBean that is a companion class to the Data Handler implementation.

A JavaBean is created by the DataHandler developer that is highly name sensitive. If the Java class that implements the DataHandler is called `com.sample.MyDataHandler` then the Java Bean must be called `com.sample.MyDataHandlerProperties`. Any Java Bean properties exposed become parameters to the configuration of the Data Handler.

Example:

```
public class MyDataHandlerProperties implements Serializable {
 private String xyz;

 public String getXyz() {
 return xyz;
 }

 public void setXyz(String xyz) {
 this.xyz = xyz;
 }
}
```

The supported property types are:

- int, Integer
- long, Long
- byte, Byte
- short, Short
- float, Float
- double, Double
- boolean, Boolean
- char, Character
- String
- JavaBean
- BindingTypeBeanProperty

Of these types, all but `BindingTypeBeanProperty` are self explanatory. The `BindingTypeBeanProperty` needs some clarification. The purpose of this property is to be able to select a DataBinding, DataHandler or FunctionSelector. When this property is defined in the JavaBean properties for the DataHandler, DataBinding or FunctionSelector, WID provides a mechanism to allow user selection. Consider the following code fragment:

```

public class MyMQDataBindingProperties {
 private BindingTypeBeanProperty dataHandler;

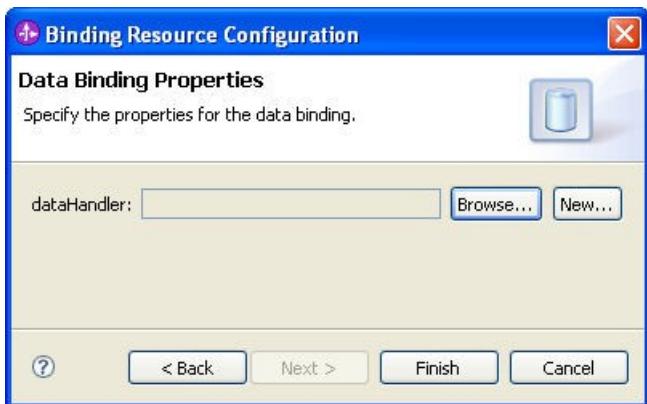
 public MyMQDataBindingProperties() {
 dataHandler = new BindingTypeBeanProperty();
 dataHandler.setTags(new String[] {BindingTags.BINDING_KIND_DATAHANDLER});
 } MyD

 public BindingTypeBeanProperty getDataHandler() {
 return dataHandler;
 }

 public void setDataHandler(BindingTypeBeanProperty dataHandler) {
 this.dataHandler = dataHandler;
 }
}

```

When a DataBinding of this type is created, a Property page will be displayed as follows:



From here, a DataHandler can be selected from a selection window. At runtime, the DataBinding implementation can query its `dataHandler` property to determine the DataHandler configuration selected by the user. The `BindingTypeBeanProperty` has a method on it called `getValue` that returns a QName. It is this QName that names the DataHandler to be used.

The next question becomes that of how does the DataBinding, DataHandler or Function selector actually obtain the Java Bean that contains the properties during runtime execution? The answer to this will be fully explained in the BindingContext section but for now assume that there is a getter that can be used to retrieve the configured Java Bean at run time.

The presentation of the properties in the WID tooling is generated by introspection of the Java Bean for the properties. If the setting of the properties in the WID UI needs to be more advanced, this can be achieved by providing another helper class. This class is given the name `<<BaseClass>>Configuration`. For example, if the Java class that implements the DataHandler is called `com.sample.MyDataHandler` then the Java Bean must be called `com.sample.MyDataHandlerConfiguration`. This class must implement the `BindingConfigurationEdit` class.

## **BindingContext**

The DataHandler interface itself extends `BindingContext` which is an interface with the following signature:

```
public void setBindingContext(Map context)
```

A common implementation of this method saves the map supplied for later use. The DataHandler interface extends `BindingContext`. It is interesting to note that neither DataBinding nor

FunctionSelector extend this interface.

The binding context map provides some runtime properties to the DataBinding, DataHandler or FunctionSelector. These properties can be retrieved from the map using architected keys into the map. The keys of interest to us are:

- `BindingContext.BINDING_CONFIGURATION`
- `BindingContext.EXPECTED_TYPE`

The `BindingContext.BINDING_CONFIGURATION` key returns the corresponding Java Bean that contains the properties.

For example:

```
MyMQDataBindingProperties myProperties;
myProperties =
(MyMQDataBindingProperties)context.get(BindingContext.BINDING_CONFIGURATION);
could be used to retrieve the properties Java Bean.
```

The `BindingContext.EXPECTED_TYPE` key will be discussed in the context of creating Business Objects.

### ***Creating a Business Object***

When a DataHandler needs to create a Business Object, we seem to have a problem. A Business Object is characterized by the pair of name and namespace, together called a QName. In order to create a new Business Object, we need to be able to obtain the expected QName. This is where one of the context mapping values comes into play. The map value for the key

`BindingContext.EXPECTED_TYPE` returns a QName instance of the expected Business Object type.  
`QName expectedType = (QName)context.get(BindingContext.EXPECTED_TYPE);`

### ***The Binding Registry***

When a DataHandler needs to be used, it has to be created. The architected mechanism is a lookup in something called the "Binding Registry". The lookup key for a DataHandler is a QName which matches the DataHandler defined in WID.

```
QName qName= new QName("http://lib",
 "MyDataHandler");
BindingRegistry bindingRegistry = BindingRegistryFactory.getInstance();
DataHandler dataHandler =
 (DataHandler) bindingRegistry.locateBinding(qName, bindingContext);
```

It is not clear what it means to pass in a bindingContext to the locateBinding method. Experience has shown that passing in null is sufficient. This causes the `setBindingContext()` method on the DataHandler instance to be called with a BindingContext that contains the necessary information including the Data Handler Properties object. The Binding Registry can be used within a Java Component or Java Snippet to obtain a DataHandler outside the context of a DataBinding. This can be extremely useful as DataHandlers can thus be invoked during other forms of processing.

### ***Calling a DataHandler from a DataBinding***

A DataBinding is a concrete mapping that is protocol/technology specific. When it is invoked, it has to transform data. This is where a DataHandler can come into play. As illustrated in the Binding Registry section, a DataHandler object can be retrieved from the registry by name. Once the

DataHandler has been obtained, either its `transform` or `transformInto` methods can be called to achieve the actual transformation. Although the name of the DataHandler can be hard coded into the DataBinding implementation, this is probably not a good idea. Instead, the selection of the DataHandler to be used can be supplied on the properties bean of the DataBinding. If a DataBinding implementation class is called MyDataBinding then the properties bean will be called MyDataBindingProperties. Refer back to the DataBinding properties section. If a property of type BindingTypeBeanProperty is created on the properties bean then selection of a DataHandler can be made much easier through the use of a search dialog. The result from the BindingTypeBeanProperty will be a QName that can be supplied to the Binding Registry to retrieve the desired DataHandler.

More details on DataHandlers, DataBinding and FunctionSelectors can be found in the EMD 1.1 specification.

### ***Creating a UI Configuration class***

When a DataHandler is configured in ID, its properties are normally displayed on a best effort basis. This can be dramatically improved by the creation of a companion class for the DataHandler that provides instructions to the ID tooling on how to display and enter the configuration properties.

Let us start with a review of the goal. The goal is to display configuration information for the user of the DataHandler. The configuration for the data handler is held in a Java Bean with the type `<DataHandler>Properties`. Each property exposed by the bean is supposed to be able to be modified by the end user.

A Java class called `PropertyDescriptor` is used to describe each property. So if the DataHandler is to show three properties, there will be three instances of a `PropertyDescriptor`. `PropertyDescriptor`s are *owned* by another Java class called a `PropertyGroup`. It is an instance of a `PropertyGroup` that is returned by the `<DataHandlerName>Configuration` class when the `createProperties()` method is invoked.

The `PropertyDescriptor` looks as follows:



- `getDisplayName()` returns the string used to provider a label or other indicator associated with the property.
- `getDescription()` returns a string that is used to provider a description of this property. This might be used in a tooltip or other help information.
- `getName()` returns the name of the property. This must be unique within all the properties within the property group. This name is used internally within the framework and should never be exposed to the end users.
- `getID()` returns another unique name for the property. It is not known how this is used by the framework. An implementation might simply return `getName()`.

- `isEnabled()` returns a boolean value to say whether or not this property is enabled (workable) or not. This could be used to gray-out a field.

The PropertyGroup looks as follows:



A PropertyGroup inherits from a PropertyDescriptor.

`getProperties()` returns an array of PropertyDescriptor. This is the set of properties to be shown to the end user.

`getProperty(String)` returns a single PropertyDescriptor that is keyed by the name of the property.

The PropertyType looks as follows:



- `getType()` returns the Class object for the type that this property represents
- `isHidden()` returns a boolean that indicates whether or not the property should be hidden from the user
- `isReadOnly()` returns a boolean that indicates whether or not the property can be edited
- `isExpert()` returns a boolean that indicates that this is an expert property (or not)
- `isPrimitive()` returns whether or not the data type is one of the primitive types
- `isSensitive()` returns whether or not the content of the field should be hidden such as for a password field.
- `isRequired()` returns a boolean to indicate that the field must have a value.

The name of this class must be `<DataHandlerName>Configuration`.

It must implement the Java interfaces called `EditType` and `BindingConfigurationEdit`. This causes the following methods to be injected that must themselves be implemented:

`PropertyGroup createProperties()`

```

void synchronizeFromBeanToPropertyGroup(...)
void synchronizeFromPropertyGroupToBean(...)
EditableType getEditableType()
boolean isOptional()
void setType(...)

```

Let us take these apart. The first method of interest is called `createProperties()`. This returns a `PropertyGroup` object. It is the responsibility of the DataHandler provider to implement this. The `PropertyGroup` interface itself has a bunch of methods that must be implemented. These are:

- `convertToString`
- `getProperties`
- `getProperty`
- `populateFromString`
- `addPropertyChangeListener`
- `getDescription`
- `getDisplayName`
- `getID`
- `getName`
- `isEnabled`
- `removePropertyChangeListener`
- `clone`

To explain these, `getProperties()` returns an array of `PropertyDescriptor` objects. Each property descriptor is a property to be shown to the user. Here is an example implementation of this method:

```

public class XSLTDataHandlerConfiguration implements
 EditableType,
 BindingConfigurationEdit {
...
 public PropertyGroup createProperties() {
 return new MyPropertyGroup();
 }
...
}

public class MyPropertyGroup implements PropertyGroup {
...
 public PropertyDescriptor[] getProperties() {
 PropertyDescriptor pd[] = new MyPropertyDescriptor[1];
 pd[0] = new FileProperty1();
 return pd;
 }
...
}

```

}

Each of the properties returned in the getProperties() array describes a visual property that is shown to the user. IBM supplies some helpers for these.

### Property Types – TableProperty

This property adds some new methods that need implemented:

- createNewRow
- deleteRow
- getCellProperty
- getColumns
- getRowProperties
- getCount
- other PropertyDescriptor methods

### Property Types – FileProperty

The FileProperty PropertyType.getType() must be a java.io.File

## ***HTTP DataBinding***

An HTTP DataBinding implements the `HTTPStreamDataBinding` interface. Creating an instance of this interface will result in a number of methods that need implementing and also an implied life cycle. For inbound processing (data coming into IBPM) ... the following methods are executed:

- `setControlParameters`
- `setHeaders` – A list of headers (`HTTPHeader`) where each header contains a name and a value
- `convertFromNativeData`
- `isBusinessException`
- `getDataObject`
- `getControlParameters`
- `getHeaders`

## ***References***

- DeveloperWorks - [Implementing custom databinding and custom function selectors in IBM WebSphere Adapters](#) – 2006-06-14
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Building a powerful, reliable SOA with JMS and WebSphere ESB -- Part 3](#) – 2006-04-19
- DeveloperWorks - [Creating custom data bindings in WebSphere Process Server without coding](#) – 2007-10-31
- DeveloperWorks - [Using WebSphere Integration Developer and WebSphere Flat File Adapter with custom data bindings](#) - 2007-10-10
- [EMD Specification](#)

## **DataObjects**

IBM's DataObjects provides a number of utility functions for programming

### **Creating an SCA Business Object**

A Business Object represents a structured piece of data. The Business Object can be created in an SCA application through a Java class called the BOFactory. This factory must be retrieved from a named SCA location:

```
BOFactory boFactory =
(BOFactory) ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOFactory
");
```

The BOFactory is part of the com.ibm.websphere.bo package and is fully documented along with the other related classes. At the simplest level, to create a new Business Object, we can use:

```
DataObject dataObj = boFactory.create("targetNameSpace", "complexTypeType Name");
```

As you will notice, the DataObject is created from the factory through the combination of target name-space and type name. The BOFactory appears to scan its current class path looking for XML or XSD documents that match these requirements and, from these values, then creates the associated DataObject. What this means is that if you need to create a DataObject in an arbitrary class, then the BO (.xsd file describing it) needs to be included as a dependency in the project in which the calling class is contained.

Here is an example. If you have a Library called "Lib" that contains a BO definition called "Customer" in name-space "http://mybiz", then if you want to create a DataObject instance of that type you would need to call:

```
boFactory.create("http://mybiz", "Customer");
```

If the code that creates the BO is in a Module, then "Lib" must be flagged on that Module as a dependency.

If the code that creates the BO is a JavaEE App, then the JAR file representing "Lib" must be added as a dependency to the JavaEE Deployment Descriptor.

### **Converting to/from XML**

IBPM provides a utility class to convert a DataObject to/from a specific formatted XML document. The class implementing this function is called BOXMLSerializer. An instance of this class can be retrieved through

```
BOXMLSerializer mySerializer = (BOXMLSerializer)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
```

Some of the methods of this class expect a rootElementName. To determine the root of a current DataObject, use the following:

```
dataObject.getType().getName()
```

Some of the methods of this call expect a namespace. To determine the namespace of the current DataObject, use the following:

```
dataObject.getType().getURI();
```

The following example illustrates turning a BO into an XML document:

```
BOXMLSerializer mySerializer = (BOXMLSerializer)
```

```

ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
String rootElementName = dataObject.getType().getName();
String targetNamespace = dataObject.getType().getURI();
ByteArrayOutputStream baos = new ByteArrayOutputStream();
mySerializer.writeDataObject(dataObject, targetNamespace, rootElementName,
baos);
String xmlText = new String(baos.toString());

```

To convert an XML string to a DataObject, the following code may be utilized:

```

String xmlText = "<XML to become a BO>";
...
BOXMLSerializer mySerializer =
(BOXMLSerializer)
ServiceManager.INSTANCE.locateService("com/ibm/websphere/bo/BOXMLSerializer");
ByteArrayInputStream bais = new ByteArrayInputStream(xmlText.getBytes());
BOXMLDocument document = mySerializer.readXMLDocument(bais);
DataObject dataObject = document.getDataObject();

```

### **XML documents with no namespace**

It is common to find XML documents that have no namespace information associated with them. In order to be able to support this kind of document, the corresponding Business Object should also be set to have no namespace associated with it.

When IBPM is supplied an XML document to be turned into a Business Object, it examines that document and provides a fully qualified name of the format {Namespace}Name. This is used to then find the corresponding Business Object definition.

When an XML document with no name-space is presented, then the result is {}Name. In order to find a matching Business Object definition, the BO must also appear as {}Name and this is achieved through nulling out the name-space for the BO. This will result in an ID based warning but this may be safely ignored.

Here is an example. Consider a Business Object called BO1 (no namespace) that has three fields. Converting this to its XML representation results in:

```

<?xml version="1.0" encoding="UTF-8"?>
<BO1 xsi:type="BO1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <f1>1</f1>
 <f2>2</f2>
 <f3>3</f3>
</BO1>

```

## **Advanced BPEL**

### **Variable initialization**

When a variable is declared in a BPEL process, it has no value until one is assigned to it. From within a Java Snippet, extra care must be taken as the variable will appear as a null-valued reference when accessed before assignment. If you need to assign a value to a variable for the 1st time in a Snippet, create an instance of the typed object. You can use the BOFactory to create a BO.

## **Scope**

WBI-SF, the predecessor to WPS, only supported a subset of BPEL. One of the items that was noticeably missing was the concept of Scope. Variables have a strong relationship to Scope. By default, all variables in BPEL have Global scope. This means that they can be used and accessed throughout the lifetime of the process anywhere in the process. This sounds good until you realize that variables may transiently hold large amounts of data and you don't want to have to worry about managing the size of that data throughout the whole process. In addition, a variable used in one section of your process may have no use or effect in others and you don't want to clutter the whole solution with irrelevant variables floating around.

Scope assists with these problems by allowing you to define a variable that exists only within a scope. This is similar to local variables in programming terms. Creating a variable in a scope means that it can only be accessed by activities in the same scope or lower.

## **BPEL - Java Mapping**

The simple variable types in BPEL map to XML Schema types which map to Java. The following are the more interesting:

- xsd:dateTime – java.util.Date
- xsd:date – String
- xsd:time - String

The format for the date/time Strings can be read about at the XSD page.

## **Variable Properties**

Consider a data structure called Order that has three fields in it ...

- amount – int
- customerId – String
- product - String

Consider a second data structure called Purchase that has three fields in it ...

- size – int
- currency – String
- custNum - String

Both data structures have a logical field that is used to hold the identity of a customer. In the Order data structure, it is the field called customerId. In the Purchase data structure, it is the field called custNum. In both cases, this field is the logical identity of the customer but in each case, it is named differently (presumably by the different data designers). If we write a BPEL process that uses either Order or Purchase and wishes to refer to the identity of the customer, the process is

going to be tightly bound to the name of the field in the data structure.

BPEL provides a concept that they call variable properties. A property is a named entity with associated data type. So a property could be created called customerIdentity with data type String. In a BPEL process, instead of referring to a specific named field in a variable, the property name can be used instead. In order for this to work, there must be a mapping or alias that relates the named property to the name of the field within the data structure.

Logically we can then create the following (note that this is not BPEL syntax):

```
Define: propertyName = "customerIdentity", type="String"
```

we can then create aliases:

```
Define: alias propertyName = "customerIdentity", data="Order->customerId"
```

```
Define: alias propertyName = "customerIdentity", data="Purchase->custNum"
```

Once these are defined, we can then get the customer identity by:

getVariableProperty("Order", "customerIdentity") will return Order->customerId

getVariableProperty("Purchase", "customerIdentity") will return Purchase->custNum

## ***Endpoint References***

In BPEL, a partner link is a place holder for both how the process appears to others and how the BPEL process sees partners that it may call. Associated with the logical partner link are the actual endpoints or bindings that, when used, serve as the actual endpoints for the service calls. These endpoints can be both queried and set within a BPEL process. There are two different types of endpoints. These are the endpoints that the process itself exposes or appears as (MY ROLE) and the endpoints at which the partners exist (PARTNER ROLE). Both can be queried but only the partner role endpoints may be set. It doesn't make any sense to dynamically change the location at which the process is actually listening.

Within either a JavaSnippet or an Assign activity, the endpoint values may be obtained or set.

Name	Description
Service	The name of the referenced service.
Address	The address of the endpoint. This may be a logical value as opposed to a TCP/IP address or hostname.
PortType	The WSDL port type being accessed. Think of it like the name of an interface in Java.
Port	A binding to the endpoint at the target service.

The Address object contains a field called value. This can be set to the target of the Partner Link. For a component in the same module, this can be <ModuleName>/<ComponentName>.

## **BPEL Endpoint Reference variable type**

A BPEL variable can be used to hold an endpoint reference. The data type used to contain such an entity is the ServiceRefType. This is described a series of XSD schema files that can also be used as Business Objects. The XSDs that described these data types can be automatically generated. By

opening the project's dependencies and select the WS-Addressing Schema Files from the Predefined Resources section, a set of additional XSDs are added into the project.

The two primary ones of interest are:

- ServiceRefType
- EndpointReferenceType

If we create a BPEL variable called 'myRef' and make it of type ServiceRefType, then we can use a BPEL Assign activity to set a partner reference to the variable.

## JavaSnippet access

### ***getServiceRefFromPartnerLink***

```
commonj.sdo.DataObject getServiceRefFromPartnerLink(
 String partnerLinkName,
 int role);
```

This method is used to retrieve the Service destination at the end of the named partner link. The role value must be one of:

- PARTNER\_LINK\_PARTNER\_ROLE
- PARTNER\_LINK\_MY\_ROLE

What is returned is a DataObject that contains an EndpointReference object retrieved with the name "EndpointReference".

Example:

Here is a java fragment retrieving a PartnerLink binding called RequestReply

```
DataObject myDo = getServiceRefFromPartnerLink("RequestReply",
PARTNER_LINK_PARTNER_ROLE);
EndpointReference ref = (EndpointReference)myDo.get("EndpointReference");
System.out.println("Address: " + ref.getAddress());
System.out.println("Port: " + ref.getPort());
System.out.println("Service: " + ref.getService().toString());
```

If the wired target is an Import of a Web Service, then the values are that of a Web Service. If the wired target is any other SCA component, then the only meaningful value is the Address field which contains the module and component which is the target. For example Module/Component.

## ***Query Properties***

Consider a BPEL process template. This is used to instantiate process instances. Now imagine that you want to find a particular instance of a process. Maybe the process model handles orders and when a customer submits an on-line order, an instance of the process gets generated and the customer is given an orderid. How do you find the specific process instance that has a specific order-id?

The answer to these questions is through the capability called query properties. A Query Property is

basically a property (or attribute) that can be associated with an instance of a process. While the process is running, a query can be executed to return a list of processes where the property has a specific value or, conversely given a process, to retrieve the current value of the property.

As multiple BPEL process instances execute, what differentiates one from another is the state of data maintained within the context of that process. The value of the query property is mapped from the value of a BPEL variable.

During the design of a BPEL process, one or more query properties can be created. They are created in the properties view in WID associated with variables. When the BPEL process executes, whatever is the current value of the variable in the Process will be the current value of the query property associated with the process instance.

### ***Implementation***

The implementation of query properties appears to be through the creation of additional tables in the BPEDB database. The tables identified as being associated with query properties are:

QUERYABLE\_VARIABLE\_INSTANCE\_T

### ***Performance***

When a variable's value changes within the BPEL process and that variable is associated with a query property, then the value must be updated in the query properties table. This means an extra SQL update for each modification of a query property associated variable.

### ***Activities***

Activities described:

Inline Human Tasks

Invoke

JavaSnippet

### ***Inline Human Tasks***

The majority of Human Task information can be found in the Human Task section. This section covers the BPEL specific in-line Human Task.

In-line Human Tasks are IBM additions to the BPEL implementation that create Tasks in the Human Task manager within the context of the BPEL process. This allows additional or overriding information to be supplied to the human task.

### ***Previous task owners***

In a BPEL process, an in-line Human Task can be performed and then in subsequent BPEL activities, the owner of that task may wish to be known. The previous owner of the task can be retrieved within the process using the following recipe.

In a Java Snippet (psuedo code):

```
ActivityInstanceData aid = bfm.getActivityInstance(
 PIID piid,
```

```
java.lang.String activityTemplateName);
TKIID tkid = aid.getTaskID();
Task task = htm.getTask(TKIID tkiid);
String owner = task.getOwner();
```

where:

bfm = The handle to the Business Flow Manager

htm = The handle to the Human Task Manager

## Invoke

Arguably, the Invoke Activity is the single most important construct in BPEL. Invoke is used to invoke the services of some other component outside of the BPEL process. Within nothing more than a set of Invoke activities, we can choreograph a set of services together.

## Expiration

IBM has extended the BPEL specification with an Expiration option. Understand that this is not standard BPEL. The semantics of Expiration is that for asynchronously implemented service calls, we can abandon the wait for a response. This also means that expiration does not apply to short running process definitions. This basically means that if we send a request to a target service over JMS and wait for a response, we can timeout after a period of time before the response comes back. The expiration can be supplied as either an absolute date/time (an expiration) or a relative time to wait (a duration) specified in seconds. If no response is received before the expiration triggers, a timeout fault is automatically raised.

note: Experimentation and observation seems to show that the expiration is the minimum amount of time to wait, don't expect the process to instantly see the fault after the expiration period, it can possibly take some time for the system to realize that it shouldn't wait any more.

The timeout value for the expiration can be specified as either:

Java code

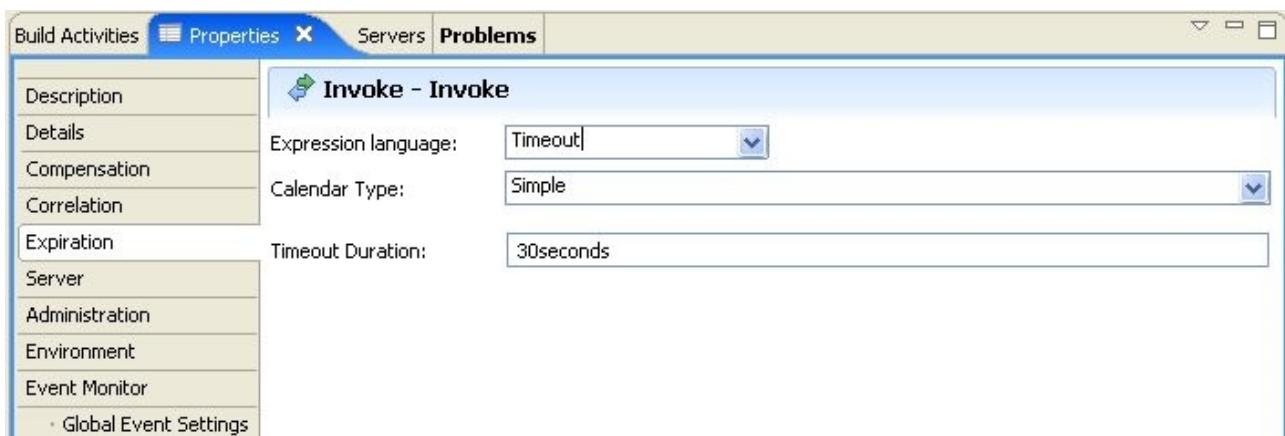
Xpath

An absolute value (using a CRON calendar)

A relative value (using a Simple calendar)

Through either Java code or XPath, access to the variables in the process can be achieved and thus the expiration can be a dynamic value based on a variable's value.

The following image illustrates setting the Expiration values.



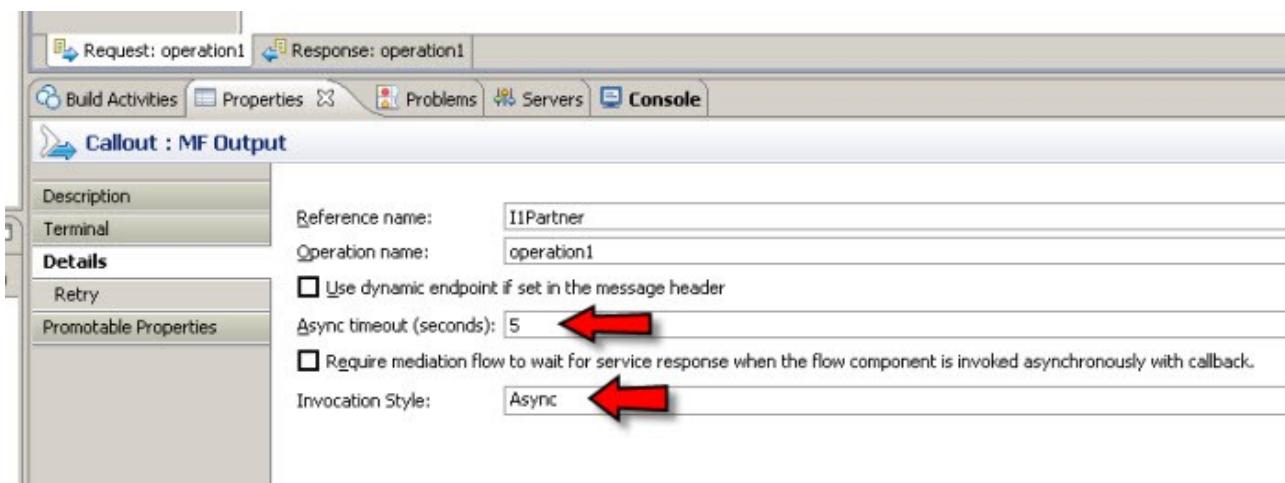
When a Timeout fires, the Timeout fault can be caught in a fault handler:



If you want to add an expiration timeout in a short running process definition, consider placing a mediation flow component between the reference terminal in the BPEL SCA component and the SCA Import to the target service. In the mediation flow component, have the interface and reference both defined as the same interface type which is that of both the BPEL process reference and the SCA Import.



In the mediation flow component, wire both the request and responses directly from their inputs to their outputs. In the Request callout node, set the a timeout value to be the required timeout and set the invocation style to be Async.



If no response has arrived within the timeout interval, the Response flow will fire on the fail terminal which can be used to signal a failure to the original caller.



## JavaSnippet

By and large BPEL allows you to perform all the tasks you might want to achieve for process choreography. IBM has extended BPEL with the concept of Snippets which can be expressed in the Java programming language. Within the Java Snippet you have access to all the variables in the process and can both set and get their values. The snippet can also be coded in the Visual Snippet language.

### Working with variables

Variables inside the BPEL Java Snippet are really Business Objects which are actually SDO DataObjects or they are boxed simple types (eg. Integer()). The recipe to create a new DataObject is documented in the SCA section of the Wiki.

### Methods available in a Java Snippet

A series of method calls are also available to you. These are:

#### **getServiceRefFromPartnerLink**

```
DataObject getServiceRefFromPartnerLink(String partnerLinkName, int role);
```

See Endpoint References for information.

#### **setServiceRefToPartnerLink**

```
void setServiceRefToPartnerLink(String partnerLinkName, DataObject serviceRef);
```

#### **getVariableProperty**

These methods either specify or return the value of a process variable's property. If either the property or the variable do not exist, a StandardFaultException of kind "selection failure" is thrown. If the value is not compatible to the type of the property, a StandardFaultException of kind "mismatch assignment failure" is thrown.

#### **setVariableProperty**

These methods either specify or return the value of a process variable's property. If either the property or the variable do not exist, a StandardFaultException of kind "selection failure" is thrown. If the value is not compatible to the type of the property, a StandardFaultException of kind

"mismatch assignment failure" is thrown.

### **getCorrelationSetProperty**

This method can be used to retrieve the properties of correlation sets that are declared at the process level. If either the property with name propertyName or the correlation set with name correlationSetName do not exist, a StandardFaultException of kind "selection failure" is thrown.

### **getProcessCustomProperty**

```
String getProcessCustomProperty(String propertyName)
```

Use these methods to access or define custom properties at the process level.

### **setProcessCustomProperty**

```
void setProcessCustomProperty(String propertyName, String PropertyValue);
```

Use these methods to access or define custom properties at the process level.

### **getActivityCustomProperty**

Use these methods to access or define custom properties at the activity level. If the activity does not exist, or activityName does not uniquely qualify an activity, a StandardFaultException of kind "selection failure" is thrown. If the value exceeds 254 bytes, an InvalidLengthException is thrown.

### **setActivityCustomProperty**

Use these methods to access or define custom properties at the activity level. If the activity does not exist, or activityName does not uniquely qualify an activity, a StandardFaultException of kind "selection failure" is thrown. If the value exceeds 254 bytes, an InvalidLengthException is thrown.

### **getLinkStatus**

This method can be used in join conditions to access the state of the incoming links.

### **activityInstance**

Use this method to return the current activity or a named activity as an object in order to access its content. The object type returned is an ActivityInstanceData. For example, assume that a preceding activity in a BPEL process was an in-line Human Task called "MyTask". Assume that it was previously executed. To obtain the user that owned that task the following code could be used:

```
ActivityInstanceData ai = activityInstance("MyTask");
String owner = ai.getOwner();
```

```
System.out.println("Owner = " + owner);
```

### **processInstance**

```
ProcessInstanceData processInstance();
```

This method retrieves the PIID object (`ProcessInstanceData`) for the callers process. This object holds a wealth of information relating to the current instance of the process. Some possible uses include:

Getting a unique name for the process. The `getID()` method returns a `PIID`. This can then have its `toString()` method called to retrieve a string representation of the process id that is unique to this instance. This could possibly be used as a correlation value or other unique id string.

### **raiseFault**

Use this method to raise a fault in the surrounding process.

### **forceRollback**

Use this to cause the compensation of the microflow. This method will not work with long running processes.

### **getCurrentFaultAsException**

```
com.ibm.bpm.api.BpelException getCurrentFaultAsException()
```

Retrieve the current fault in a fault handler as a Java object.

## ***Standard Functions***

Arrays

Converter

Date

current date and time

Returns a `java.util.Date` that represents the current time.

format locale date to string using pattern	
Inputs	<code>java.util.Date</code> – The date to be formatted <code>String</code> – The pattern to be applied to the date. The pattern is a pattern string as supplied to <a href="#">java.text.SimpleDateFormat</a> .
Outputs	<code>String</code> – A string representation of the formatted date
Description	Returns a string representation of an input date that has been formatted to a pattern.

## ***Custom Properties***

### **References**

- DeveloperWorks – [Setting custom properties for new query requirements in WebSphere Process Server](#) – 2010-06-02
- DeveloperWorks – [Selecting a human task with custom properties using WebSphere Process Server](#) – 2010-03-26

## ***BPEL Process Migration***

### **References**

- DeveloperWorks - [Migrating process instances in WebSphere Process Server V7](#) – 2010-08-11

## **Versioning**

Here are notes on versioning SCA and BPEL components.

See also:

- DeveloperWorks - [Versioning and dynamicity: Part 1: Creating multiple versions of a business process with WebSphere Process Server](#) – 2008-02-06
- DeveloperWorks - [Versioning and dynamicity: Part 2: Applying versioning and process dynamicity using WebSphere Process Server V6.1](#) – 2008-03-05
- DeveloperWorks: [Designing and versioning compatible Web services](#) – 2007-03-28
- DeveloperWorks - [Versioning business processes and human tasks in WebSphere Process Server](#) – 2008-08-13
- DeveloperWorks - [Versioning and dynamicity with WebSphere Process Server](#) – 2006-02-22
- DeveloperWorks - [Migrating process instances in WebSphere Process Server V7](#) - 2008-08-11

# User Interfaces

## ***Business Space***

Business Space is a UI framework designed for end users to interact with BPM functions. Business Space is a specialization of the IBM Lotus Mashups technologies. Business Space is built from a set of one or more "spaces". Each space contains one or more pages. Each page contains one or more widgets that are laid out on the page.

Each BPM product installed adds one or more Business Space widgets to the catalog of available widgets.

## **Events**

Widgets can communicate with each other through the publish/subscribe of events between them. When a widget is registered with Business Space, its registration describes which events it can publish to and which events it can subscribe upon. When a widget is added to a page, it can then be wired to other widgets. One widget will act as the publisher of the event, and the other act as the subscriber. Two widgets can only be wired together if they support the same type of event.

### See Also – Business Space

Flex Custom Widgets in Business Space - Using Flex within Business Space

### References – Business Space

- DeveloperWorks - [Developing a custom tree iWidget for WebSphere Business Space that retrieves data from a REST service using Apache Wink](#) - 2011-07-13
- DeveloperWorks - [Using the Java Persistence API to initialize HTML-Dojo forms for WebSphere Business Space](#) - 2010-09-08
- DeveloperWorks – [Using Business Space to manage business process applications](#) – 2010-03-24
- DeveloperWorks – [Customizing WebSphere Business Space V6.2 Human Management widgets](#) – 2010-02-24
- DeveloperWorks – [Customizing HTML-Dojo forms for Business Space powered by WebSphere](#) – 2009-04-09
- DeveloperWorks – [Customize your BPM user interfaces with business spaces](#) – 2008-12-04
- RedBook - [Building IBM Business Process Management Solutions Using WebSphere V7 and Business Space](#) – 2010-06-10

## **Custom Widgets**

The widgets provided by IBM for business space are not the only widgets possible. You can create and use your own widgets to augment the existing functions. There are a number of steps and piece-parts required to build a new widget.

A widget is configured to Business Space by providing an XML configuration file that corresponds to the iWidget specification. Contained within this file are a number of properties that are used by the Business Space runtime to control how the widget behaves and is displayed.

In addition to the visual characteristics of the widget, we should realize that a widget can send events to other widgets as well as received events from other widgets. This is a form of publish and subscribe. The XML document also holds the information on what a widget can send to other widgets and can receive from other widgets.

An example iWidget XML control file may look as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<iw:iwidget id="iWidgetEditorTests1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:iw="http://www.ibm.com/xmlns/prod/iWidget"
 supportedModes="view">
```

```

mode="view"
lang="en"
name="Hello">
<iw:content mode="view">
<![CDATA[
<div>Hello World</div>
]]>
</iw:content>
</iw:iwidget>

```

A deep understanding is recommended of the structure and semantics of this control file. IBM provides a rich function GUI editor that hides the majority of the details allowing values to be entered in a much more pleasing manner. See iWidget editor.

In addition to the iWidget XML control file, there is a second file which is a Business Space specific Catalog XML file. The purpose of this file is to tell Business Space about the nature of the widget so that Business Space can present this widget in its catalogs of available widgets.

Finally there is one more XML file of importance to us when building custom Business Space widgets. This file is the Endpoint Registration XML file. The purpose of this file is to provide information to the widget about where it should connect to at runtime in order to communicate with a WPS server instance. Business Space widgets typically make REST style communication calls to the WPS server.

### **iWidget Mode**

One of the core concepts of an iWidget is its *mode*. Think of the mode of a widget as being a "state" that the widget as a whole may be in. Depending on the value of the mode, the widget can render itself in different ways. For example, if a widget is in view mode, then it might be displaying business data. If the same widget is in configuration mode, it might be displaying configuration data for itself such as which database to read data from.

A widget can be told its mode ... which is another way of saying how the widget is to represent itself for a particular task. As an example, setting its mode to:

- `view` - The widget shows itself "normally"
- `edit` - The widget shows itself in a fashion that allows the user to customize its attributes
- `help` - The widget shows itself in a fashion that allows the user to learn how to use it

Business space defines these three modes of operation.

### **iWidget Structure**

The iWidgets supplied for Process Server can be found in the directory `<ProcessServer>/BusinessSpace/widgets`. These serve as a useful source of reference and should be examined to see how IBM has built its own widgets.

What follows next is a breakdown on the logical content of the iWidget XML documents. Note that these conform to the iWidget specification v2.0 which is **very** different from the previous version that was used in previous product releases.

The XML control file begins with an

- `<iw:iwidget>` element which may contain the following child elements
- `<iw:content>`

- <iw:resource>
- <iw:eventDescription>
- <iw:event>

## iwidget

The <iw:iwidget> element is the root container in the document used to describe the iWidget. It has many possible attributes for its configuration.

Attributes:

- id
- allowInstanceContent
- supportedModes

A **space** separated list of supported widget modes. There must be a corresponding content element with corresponding mode attribute for every mode listed. The allowable value set can contain:

- view
- edit
- help
- For every mode defined, there must be a corresponding <iw:content mode="xxx"> section.
- iscope

The name of a JavaScript Dijit widget that implements the code. The JavaScript for this widget should include a `dojo.provide` and a `dojo.declare`. The JavaScript file containing the widget should be named in an <iw:resource> tag.

- name
- lang

## Example

```
<iw:iwidget xmlns:iw="http://www.ibm.com/xmlns/prod/iWidget"
 iScope="com.kolban.AppStarter"
 id="appStarter"
 name="appStarter"
 supportedModes="view">
```

## resource

```
<iw:resource id="{resourceName}" src="{uri}" version="{version}"
 mimeType="{mimeType}" callback="" />
```

Attributes:

- id
- src
- version

- mimeType
- callback

Example

```
<iw:resource src="appStarter.js" />
```

### **eventDescription**

Attributes:

- id
- lang
- payloadType

### **event**

- Attributes:
- eventDescName
- id
- published

### **content**

This tag contains the content that will be displayed/loaded in the Widget. This is usually HTML. The `<iw:content>` element is a child of the `<iw:iwidget>` element.

Attributes:

- mode  
The mode to which this content applies. There must be a content element with associated mode for each entry listed in the supportedModes attribute of the iWidget element.

## ***iWidget Modules/Data Structures***

- ItemSet
- ManagedItemSet - Data which the widget manages (saves)
- ItemSetDescription
- ItemDescription
- iContext - Context data for the widget
- iEvents - Events processing
- iEventDescription
- iEvent
- IO data for the widget.

## Managed Item Set

The `ManagedItemSet` is part of the `iWidget` specification. It defines data that is managed by the widget itself. This includes saving the data for later retrieval. This can be used for setting the configuration/properties of the widget so that later it can be restored.

The `ManagedItemSet` for an `iWidget` can be retrieved from the widget's `iContext` using the `getiWidgetAttributes()` function.

```
ManagedItemSet getiWidgetAttributes();
```

For example,

```
var myAttributes = this.iContext.getiWidgetAttributes();
```

Although the `iWidget` spec shows that the `setItemValue` function can take an object to save, reality is that there is a limitation of the `iWidget` container. To help with this, we should encode the object that we want to save as a string and save the string value as the attribute. In JavaScript, we can employ the JSON encoding and leverage the `dojo.toJson` function to encode a JavaScript object. When we retrieve the value, we can use the `dojo.fromJson` function to get the JavaScript object back again.

For example:

```
var x = new Object();
x.a = "Hello";
x.b = "World";

var myAttributes = this.iContext.getiWidgetAttributes();
myAttributes.setItemValue("test", dojo.toJson(x), false);
myAttributes.save();

// Retrieve ...
var y = dojo.fromJson(myAttributes.getItemValue("test"));
```

Also note that saving attributes is recommended when the widget is in 'edit' mode. The actual persisting of these attributes back to the server only occurs when the widget goes from the edit mode back to view mode.

## Functions on ManagedItemSet

```
ManagedItemSet setItemValue(in String itemName,
 in Object value,
 in boolean readOnly /*optional*/);
Object getItemValue(in String itemName);
ManagedItemSet removeItem(in String itemName);
String[] getAllNames();
boolean isReadOnly(in String itemName);
null save(in Function callbackFn /*optional*/);
boolean addListener(in Function listener);
boolean removeListener(in Function listener);
ManagedItemSet clone();
```

- `setItemValue`

This method sets an item within the `ManagedItemSet`, creating or replacing an existing entry as needed. Since only one value is allowed for any one name, the supplied value replaces any existing value for the supplied name. Note that while an arbitrary Object can be specified for

the value, the `iContext` will only persist data fields and therefore any functions defined on the Object will be lost. It is also likely that any specific type used will also be lost across the persistence operation. Optionally marking an item as `readOnly` indicates to the `iContext` that while this item may be shared with other components on the page, the access of those components SHOULD not include changing or removing the item. When successful, this method MUST return a handle to the `ManagedItemSet`. On failure, it MUST return null.

- `getItemValue`  
This method returns the value for the named item from the set. On failure, this method MUST return null.
- `removeItem`  
This method removes the named item from the set. When successful, this method MUST return a handle to the `ManagedItemSet`. On failure, it MUST return null.
- `getAllNames`  
This method returns an array of strings, providing the name each item currently in the set. If the set contains no items, this method MUST return null.
- `isReadOnly`  
This method returns a boolean indicating whether or not the item specified by the supplied name can be modified by the user (or code acting on their behalf).
- `save`  
This method requests the `iContext` save the current state of the `ManagedItemSet`. The `iContext` MAY also choose to save the `ManagedItemSet` at times other than when iWidgets request a save. This method MAY operate asynchronously. The `iContext` MUST invoke any supplied `callbackFn` upon completion of the save attempt. The signature for the function is:

```
function(in String managedItemSetName, in boolean success);
```

`addListener`

This method returns a boolean indicating whether or not the request to add a listener to changes in iWidget v1.0 Specification the `ItemSet` was successful. Note that the signature for such a listener is:

```
module
{
 null listener(in iEvent ev);
}
```

`removeListener`

This method returns a boolean indicating whether or not the request to remove a listener was successful.

`clone`

This method returns a new `ManagedItemSet` that is a duplicate of the current `ManagedItemSet`. While this method does provide for cloning all the values within the `ManagedItemSet`, in general this will only clone the data fields for complex Objects, both type information and any embedded logic will most likely be lost.

## iWidget iContext

```
ManagedObjectSet getiWidgetAttributes();
ManagedObjectSet getUserProfile();
ManagedObjectSet getiDescriptor();
ItemSet getItemSet(in String name, in Boolean private);
Object iScope();
String processMarkup(in String markup);

null processWidgets(in DOMNode node);
Element getElementById(in String id);
Element[] getElementByClass(in String className);
Element getRootElement();
null requires(in String requiredItem,
 in String version /*optional*/,
 in String uri,
 in Function callbackFn /*optional*/);

iEvents iEvents;
IO io;
xml xml;
String widgetId;
```

- `getiWidgetAttributes`

This method returns the ManagedItemSet that provides access to the iWidget's customization attributes. If there is no ManagedItemSet related to the iWidget's customization attributes, this method MUST create an empty set and return it.

- `getUserProfile`

This method returns the ManagedItemSet that provides access to the user's profile data. If there is no ManagedItemSet related to the user's profile, this method creates an empty set and returns it. If access to the user's profile is denied, this method returns null. While it is a matter left to the iContext's policy, it is likely that most items within the user profile will be marked as "readOnly". Among the properties returned in this ManagedItemSet there are:

- `j2ee_principalname`
- `principalName`
- `uid`

- `getiDescriptor`

This method returns the ManagedItemSet that provides access to attributes that both the iContext and the iWidget need to understand. If there is no ManagedItemSet related to the iWidget's descriptive items, this method MUST create an empty set and return it.

- `getItemSet`

This method returns an ItemSet corresponding to the requested name. If it does not already exist, an ItemSet will be created and associated with the supplied name. If a new ItemSet is created, the "private" parameter controls whether or not the ItemSet can be exposed to other components. If the ItemSet already exists, the "private" parameter is ignored. If access to the desired ItemSet is denied or the ItemSet cannot be created, null is returned.

- `iScope`

This method returns an instance of type Object that was initialized prior to the loading of the iWidget (either as a generic Object or an instance of the encapsulation Object referenced by declarative means. Its purpose is to support proper encapsulation of the iWidget's assets (variables and methods) such that multiple instances of the iWidget can be loaded into a single

page's DOM without stepping on each other.

- `processMarkup`

This method requests the iContext to process the markup such that it can be inserted into the iWidget's markup and properly interact with the page (that is, logically extends the processing related to loading the page). Where, when, and how the markup is inserted into the page is the responsibility of the iWidget. On success, this method MUST return the processed markup. On failure, it MUST return null.

- `processWidgets`

This method requests the iContext to process the subtree under the supplied node for the purpose of resolving and instantiating any referenced iWidgets.

- `getElementById`

This method provides the same semantics as the DOM method by the same name with the distinction that this method will restrict the search to the iWidget's markup rather than the entire page. This is an extremely important function. It is not uncommon for a widget to want to replace (add/delete) content in the DOM to update the widget's appearance. If instead of using this function, one used the "normal" element finders for DOM one may very well find a different element in some either unrelated widget or else in a different instance of the same widget. Either way, this resolves the problems.

- `getElementByClass`

This method returns an array of Elements within the iWidget's markup that have the supplied value as one of those specified by the Element's "class" attribute.

- `getRootElement`

This method returns the root element of the iWidget such that the iWidget can easily do things such as searching its own markup. This also means the root element can be a convenient location to place items that the iWidget wishes to access later.

- `requires`

This method provides the means for an iWidget to declare dependency on a set of non-required items. For these items, no URI should be specified as the iContext is responsible for loading any dynamic portions of itself. In addition, sharable resources (for example "dojo.js") can be loaded just once for all iWidgets using the resource via this method. As there may be issues with having multiple versions of such resources loaded at the same time, this method includes a version parameter to inform the iContext of any version dependency. If no value is supplied for the required version, the iWidget author is indicating that any version is acceptable. As iContexts are likely to do such dynamic loads in an asynchronous manner, the callbackFn provides a means for the iWidget to become aware that the required resource is available. It should be noted that the callbackFn may be invoked prior to the return from this request, especially for those resources already loaded. The callbackFn signature is of the form:

```
function(requiredItem, uri, resourceHandle /* when appropriate */)
```

The following names refer to optional functionality defined here:

`io`

`xml`

`iEvents`

This field contains an object that provides access to event services in a manner allowing the iWidgets on the page to interact in a loosely coupled manner while also providing control to whomever is defining the overall page/application. Types related to eventing are defined in a

separate section below.

#### io

This optional field contains an object that provides access to io services in a manner allowing the page as a whole to remain consistent, particularly in the face of server-side coordination between related application components. See also: iWidget IO.

#### xml

This optional field is a placeholder for future definitions providing explicit support for XML-oriented processing. This version of the specification provides no such definitions, but they are expected in future versions. Extensions supported by more advanced iContexts SHOULD follow the pattern used for the iEvents and IO portions of these definitions. This allows iWidgets to determine the support for a "foo" extension defined by a group named "bar" using a construct of the form:

```
var fooSupported = (iContext._bar & iContext._bar.foo);
```

#### widgetId

This field appears to be the Dojo Dijit ID for the Dijit Widget

## Constants

```
iContext.constants.mode.VIEW = "view"
iContext.constants.mode.EDIT = "edit"
iContext.constants.mode.HELP = "help"
iContext.constants.ATTRIBUTES = "attributes"
iContext.constants.IDESCRIPTOR = "idescriptor"
iContext.constants.USERPROFILE = "userprofile"
iContext.constants.keys.SHIFT = 1
iContext.constants.keys.ALT = 2
iContext.constants.keys.CTRL = 4
iContext.constants.keys.META = 8
iContext.constants.keys.CAPSLOCK = 16
```

- `iContext.constants.mode.VIEW`  
The generated markup fragment(s) should reflect normal interaction with the iWidget.
- `iContext.constants.mode.EDIT`  
The generated markup fragment(s) should reflect iWidget interactions for editing iWidget attributes.
- `iContext.constants.mode.HELP`  
The iWidget should generate markup fragment(s) to assist the user with interacting with the iWidget.
- `iContext.constants.ATTRIBUTES`  
Name for the ManagedItemSet holding the customization attributes.
- `iContext.constants.IDESCRIPTOR`  
Name for the ManagedItemSet holding the items describing the iWidget.
- `iContext.constants.USERPROFILE`  
Name for the ManagedItemSet holding data about the user.

## iEvents

```
module iEvents
{
 null publishedEvents(in iEventDescription eventDesc[]);
 null handledEvents(in iEventDescription eventDesc[]);
 null fireEvent(
 in String name, /* event name, preferably a serialized QName */
 in String type, /* optional reference to type,
 preferably a serialized QName */
 in Object payload /* optional ... the event's data */);
}
```

- `publishedEvents`

This method informs the iContext of events that the iWidget might originate.

- `handledEvents`

This method informs the iContext of events that the iWidget is capable of handling.

- `fireEvent`

This method informs the iContext that distributing an event of the type described by the parameters is appropriate. Whether or not such an event will actually be created and distributed will depend on whether the iContext has any handlers for that event.

## io

```
XMLHttpRequest XMLHttpRequest();
URI rewriteURI(in URI uri);
XMLHttpRequest request(in requestVerb,
 in URI uri,
 in Function callbackFn,
 in String message /*optional*/,
 in [{headerName, value}] requestHeader /*optional*/);
```

- `XMLHttpRequest`

This object wraps the native XMLHttpRequest support to provide a consistent page in the face of asynchronous updates, especially those that may cause server-side coordination between the server-side components related to multiple iWidgets on the page. Note that the need to proxy URIs to domains outside the domain that sourced the page MAY also need the iContext to modify the specified URI prior to supplying it to the native XMLHttpRequest implementation. Note that either the iContext or proxy implementation MAY refuse to activate the requested URI due to security or policy reasons. All of the semantics/syntax of the native support apply with the exceptions/additions noted below:

- None at this time.

- `rewriteURI`

This method takes a URI as a parameter and returns a URI that the browser can resolve for accessing the supplied URI. Examples of usage include resolving a relative URI against the source of the iWidget's definition and resolving other URIs to address any intermediate gateways, such as a proxy server.

- `request`

This convenience method creates a new XMLHttpRequest, registers any supplied callbackFn, sets it to asynchronous only if a callbackFn is provided, sets any supplied request headers and sends the supplied message as the request body if the requestVerb is "post" or "put" and executes the supplied requestVerb against the supplied URI. If the requestVerb is not one of "get", "post", "put" or "delete," then "get" is used.

## Notes

### Getting the URL for the widget

The URL for the widget can be obtained using:

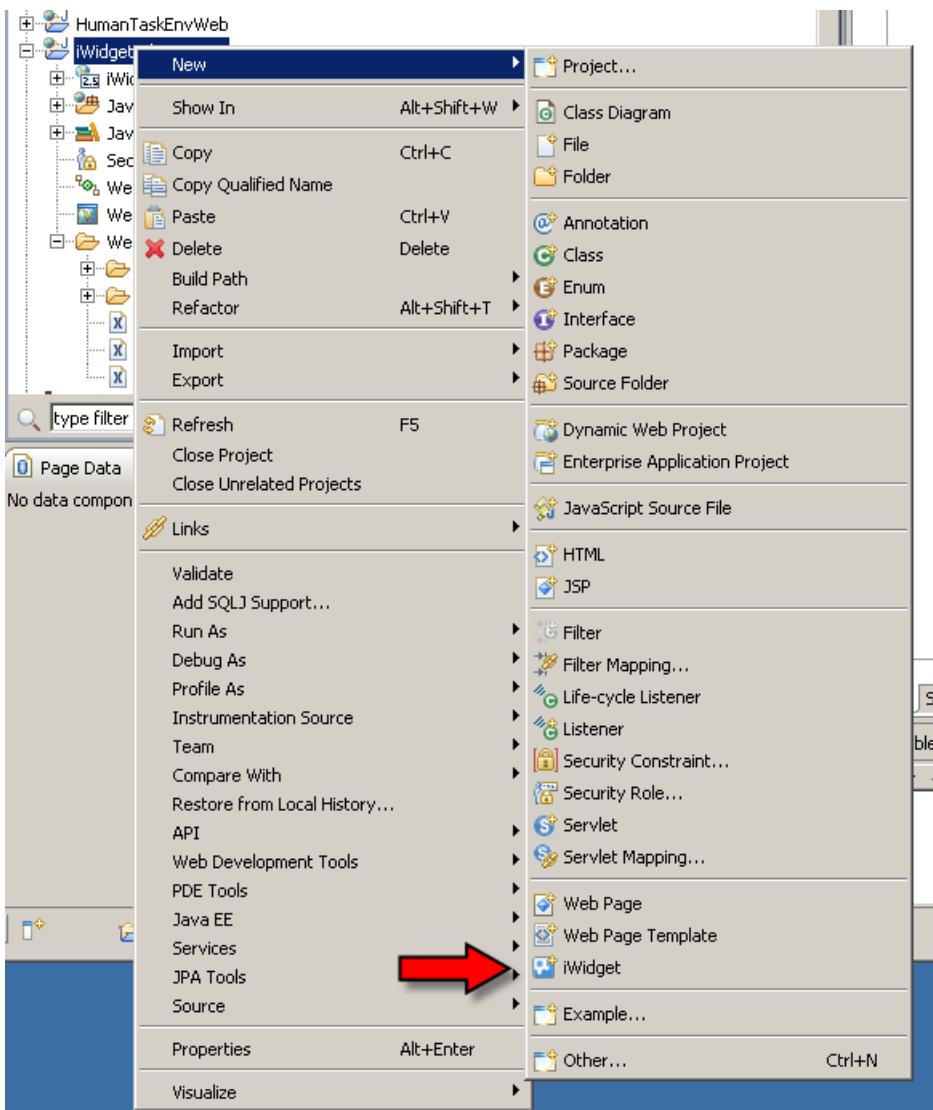
```
var rootString = io.rewriteURL("");
```

## Misc

After changing the Business Space Registry, much of the documentation says to execute an expensive restart of the server. Experience so far seems to be showing that simply stopping and then restarting the application called `IBM_BSPACE_WIDGETS` appears to be sufficient.

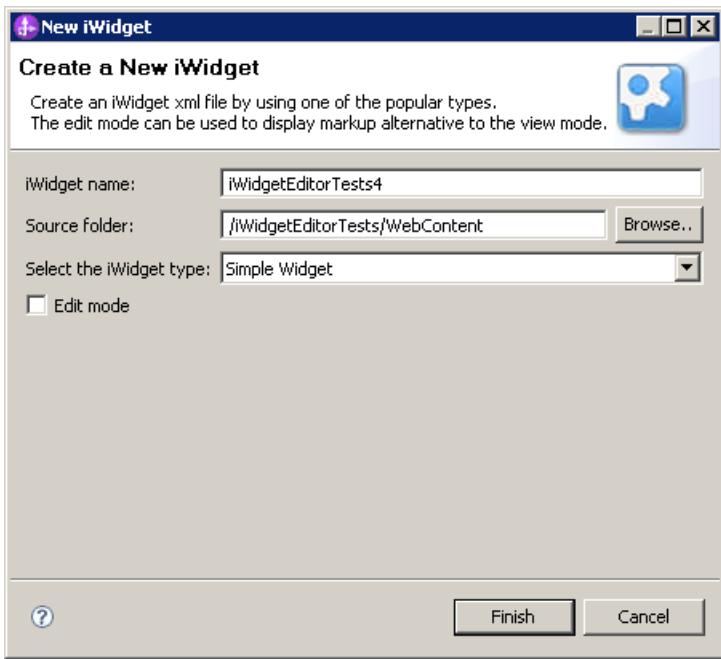
### *iWidget editor*

In the latest versions of ID and RAD, there is a an iWidget editor built into the development tooling. The iWidget editor is documented in the RAD InfoCenter. This editor visualizes the iWidget XML description file in a custom editor that is designed to show the values of an iWidget. In addition, an iWidget control file can be created as a new artifact. To create a new iWidget, select New from a Web project. You must be in the Web Perspective. In the list of available artifacts, the iWidget entry can be found:



If the entry does **not** show up in the list, choose it from the New artifact list. It can be found in the Web folder.

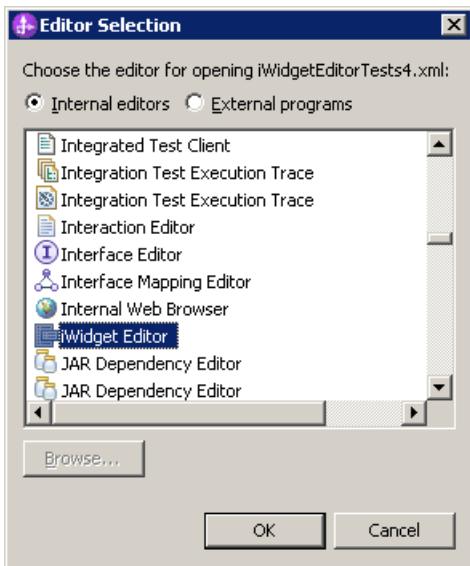
This launches a wizard in which details of the new iWidget can be entered:



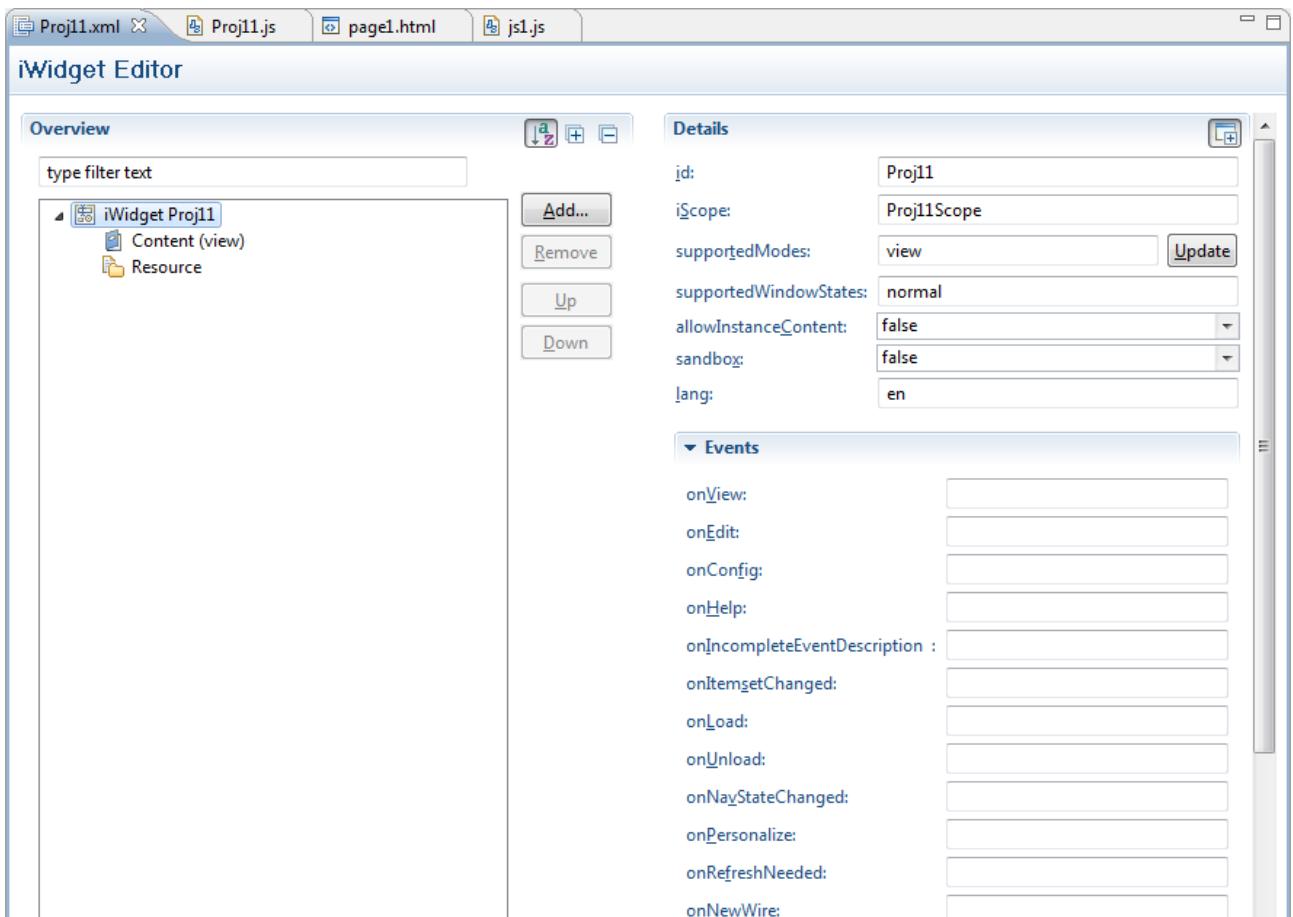
In the editor, the iWidget has a drop-down selection for the type of iWidget. The selections available are:

- Simple Widget – A simple Hello World widget
- Event Publisher Widget – An iWidget instance which includes an event to send data
- Event Subscriber Widget – An iWidget instance which includes an event to receive data

Once created, a new XML artifact appears in the project. To open this file in the iWidget editor, select Open With → Other and select iWidget Editor



Here is an example of an iWidget file opened in the editor with some data entered:



## **BSpaceWidgetRegistry**

`getWidgetRegistry()`

Returns the business space registry for the widget that contains the instance of this class.

`getWidgetRegistryServiceURI()`

Returns the service URI that returns all the widget registries.

`getWidgetRootURI()`

Returns the URI to the location of the widget definition XML.

`getWidgetInfoByWidgetId(widgetId)`

Undocumented.

`getServiceEndpoint(key)`

Returns the endpoint that matches the given key.

`getServiceURLRoot()`

Returns the service URI root of the business space.

`isInBSpace()`

Returns true if this widget is hosted in business space. False otherwise.

`getDefinitionXMLPath()`

Returns the path to this widget's definition XML.

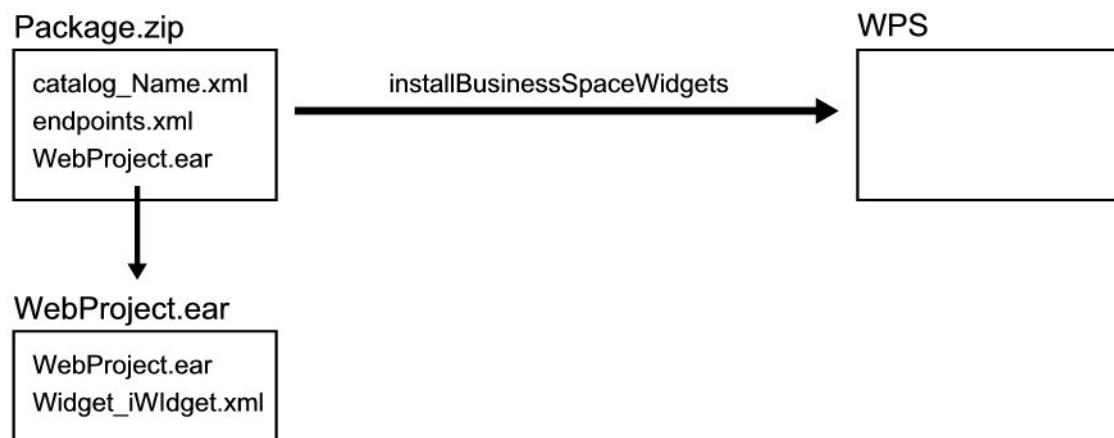
## **BSpaceGeneralHelper**

- `getCurrentSpaceId()`  
Returns the current business space id. If no id is found then the DEFAULT\_SPACE\_ID is returned.
- `getCurrentPageId()`  
Returns the current page id. If no id is found then null is returned.
- `createWidgetMenuItem(caption, action, iconClass)`  
Create a menu item for itself under BSpace platform. No effect in other platforms.
- `setWidgetMenuItemEnabled(action, booleanEnabled)`  
Enable/disable a menu item for itself under BSpace platform. No effect in other platforms.
- `setCreateTaskMenuItemEnabled(booleanEnabled)`  
Enable/disable the widget create attachment task (send widget) menu item
- `removeIWidget()`  
This method allows the iWidget to close itself
- `isSpaceOwner(spaceid)`  
This method returns true if the current user is owner of a specific space.
- `isCurrentUser(userid)`  
This method returns true if the userid is current user .
- `isSuperuser()`  
This method returns true if the current user is a superuser in BSpace.
- `showIWidgetRequiresConfiguration(container)`  
This method shows a message informing the user that the iWidget requires to be configured before use.
- `createIEventObject(eventName, payloadType, payload)`  
This handy utility method create an iEvent object for you from the information you passed in as parameter.
- `fireSwitchSpacePageEvent(spaceId, pageId, iEventArray)`  
This method allows widget to fire event with known space id and page id and result in switching to that specific space and page. Once this method is called, it will switch to the space and page specified and broadcast the event to all widgets.
- `retrieveIEventFromEventHolder()`  
This method allows widget to retrieve any waiting event in the global event holder queue. The method will return an array of all waiting iWidget iEvent if there is any.
- `getPageIdFromResourceType(resourceType)`  
Returns the hidden linked page id from resource type. If no id is found then the null is returned.
- `safeHandleCallback(callbackContext, callbackFunction)`  
Provide a safe way for handling async call callback. This will check to see if the iWidgetContainer still has the iWidget before proceeding with the real callback function.
- `printDebug(debugString)`  
General debug method so that only printing debug statement if isDebug in djConfig is true.

- `handleException(className, method , errorMessage ,exception)`  
This function allows the caller to display an error caught from a widget.

## Registering Widgets

In WPS 7.0, a new wsadmin task was introduced that performs the installation actions more elegantly than previous releases. The command is called `AdminTask.installBusinessSpaceWidgets`.



The syntax for the new WSAdmin task is as follows:

```
AdminTask.installBusinessSpaceWidgets(' [
 -nodeName nodeName
 -serverName serverName
 -widgets pathToWidgetZipFile]')
```

The input to this file is a manually created ZIP file. To create the ZIP file, first create a new folder. In that folder, create the following sub-folders:

`ear`

Contains the EAR file for the Widget Web Project. This EAR contains the iWidget XML file and the JavaScript amongst possible others.

`catalog`

Contains the `catalog_nameOfWidget.xml` file. The structure and content of this file is described at The Catalog File.

`endpoints`

Contains the endpoint XML file

`help (optional)`

Once done, ZIP up the data so that these folders are the immediate children in the ZIP. This is the ZIP file to be passed to the `installBusinessSpaceWidgets` script in the `-widgets` parameter.

The effect of running this command will be to install the EAR contained within the ZIP as well as register the new Business Space widget. Experience shows that although the EAR application is installed, it is not automatically started. It must be started before the widget can be used in Business Space.

An almost identical command is available to remove (un-install) a previously installed widget. Like the install script, the un-install script takes the ZIP file as input.

```
AdminTask.uninstallBusinessSpaceWidgets(' [
 -nodeName nodeName
```

```
-serverName serverName
-widgets pathToWidgetZipFile]')
```

Updating the information about the widget in WPS can also be achieved through scripting:

```
AdminTask.updateBusinessSpaceWidgets('[
-nodeName nodeName
-serverName serverName
-catalogs pathToCatalogXMLFile]')
```

The same command can be used to update the endpoints XML file:

```
AdminTask.updateBusinessSpaceWidgets('[
-nodeName nodeName
-serverName serverName
-endpoints pathToEndpointsXMLFile]')
```

After installing or updating a widget (at least in test), restart the server.

When installing a widget, the catalog file is copied into:

```
<Profile>/BusinessSpace/Node/Server/mm.runtime.prof/config
```

Deleting the file from this location will also delete the widget. The file called catalog\_default.xml should also be edited to remove the associated includes.

Similarly, a directory called:

```
<Profile>/BusinessSpace/Node/Server/mm.runtime.prof/endpoints
contains the endpoint files.
```

## ***The Catalog File***

The XML Catalog file that is used to register a widget does not appear to be any known industry standard. It appears to be IBM specific and very technical in nature at that. No publicly understood documentation is known to exist that describes this mysterious content. By examination (and guesswork), the Catalog XML file seems to contain the following information.

In WPS v7.0, a new catalog structure was designed that differs from that of previous releases. The name of the XML file should be:

```
catalog_nameOfWidget.xml
```

Documentation on this is currently poor. Its general structure appears to be:

```
<catalog id="???">
 <resource-type>Catalog</resource-type>
 <category name="???">
 <title>Text</title>
 <description>Text</description>
 <entry id="{namespace}name" unique-name="{namespace}name">
 <title>Text</title>
 <description>Text</description>
 <definition>Path???.</definition>
 <preview>Path???.</preview>
 <icon>Path???.</icon>
 <previewThumbnail>Path???.</previewThumbnail>
 <shortDescription>Text</shortDescription>
 <metadata name="com.ibm.bspace.version">1.0.0.0</metadata>
 <metadata name="com.ibm.bspace.owner">IBM</metadata>
 <metadata name="com.ibm.bspace.serviceEndpointRefs">
 [

```

```

 "name": "serviceUrlRoot",
 "required": "false",
 "refId": "endpoint://{{namespace}}name",
 "refVersion": "1.0.0.0"
 }]
</metadata>
</entry>
<category>
</catalog>

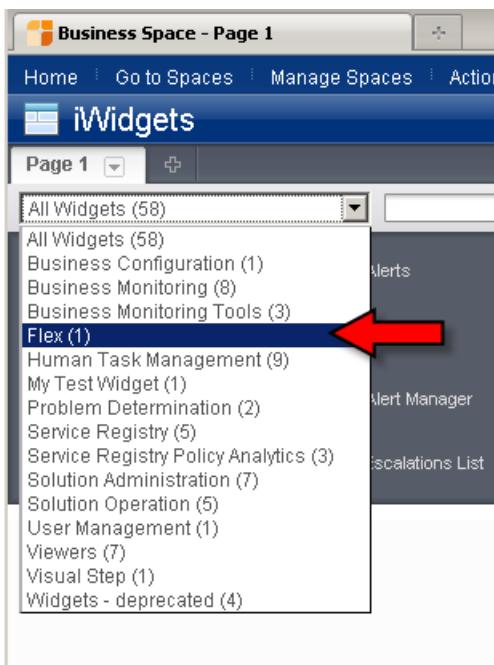
```

The text entries in the XML Catalog can be NLS encoded using the following format:

```
<nls-string lang="en">Human Readable Text</nls-string>
```

The fields in the Catalog structure are as follows:

- catalog/@id  
The name of the widget seems to work fine here. It is not known how this property is used.
- catalog/resource-type  
Examples seem to show this to be the constant called Catalog.
- catalog/category/@name  
The category is the "folder" or "group" in which this Business Space widget will appear and be housed.
- catalog/category/title  
This is the text that is shown in the Category selection.

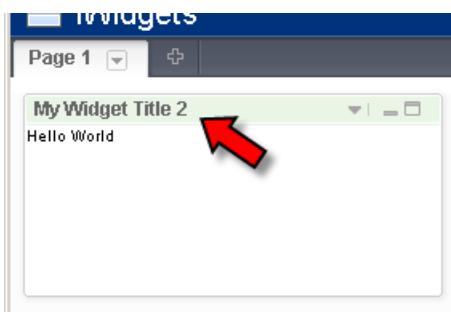


- catalog/category/description  
It is not known where this description shown up in Business Space.
- catalog/category/entry/@id  
This entry should be a namespace/name (eg. {MyNameSpace}MyName) formatted entry. It is not known how this is used.
- catalog/category/entry/@unique-name

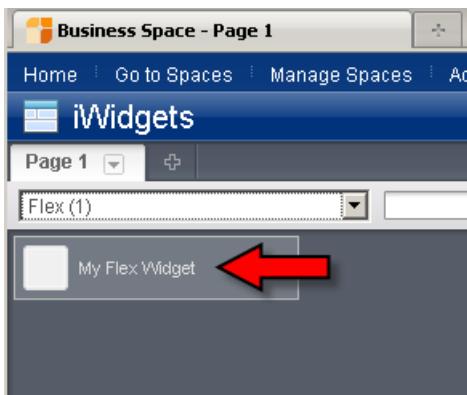
This attribute seems to hold the same value as catalog/category/entry/@id

- catalog/category/entry/title

This attribute is used to set the initial title of an instance of the Widget when created in Business Space.



It is also used in the category selection in a couple of places:



and



catalog/category/entry/description

The description shows up in the category selection



#### `catalog/category/entry/definition`

This entry points to the path of the iWidget XML file available from a URL. This links together the Catalog data and the iWidget XML description data. It is from here that the runtime can obtain the iWidget XML description document.

- `catalog/category/entry/preview`  
This is a web path to an image that will be used as a preview of what the widget will look like in a page. Link to a 160x125 pixel sized image.
- `catalog/category/entry/icon`  
This is a web path to an image that will be used as an icon for the new widget. Link to a 20x20 pixel sized image.
- `catalog/category/entry/previewThumbnail`  
This is similar to the previous preview image but smaller. Link to a 64x48 pixel sized image.
- `catalog/category/entry/shortDescription`
- `catalog/category/entry/metadata/@name="com.ibm.bspace.version"`  
It is not known how this attribute is used.
- `catalog/category/entry/metadata/@name="com.ibm.bspace.owner"`  
It is not known how this attribute is used.
- `catalog/category/entry/metadata/@name="com.ibm.bspace.serviceEndpointRefs"`  
Examples seem to show this to be a piece of encoded JSON objects with fields that include:
  - `name`
  - `required`
  - `refId`  
This appears to be a string of the format:  
`endpoint://{{?}}???`
  - `refVersion`

## **Registry Endpoints XML**

The endpoint registry XML example looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessSpaceRegistry
 xmlns="http://com.ibm.bspace/BusinessSpaceRegistry"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://com.ibm.bspace/BusinessSpaceRegistry
BusinessSpaceRegistry.xsd ">
 <Endpoint>
```

```

<id>{namespace}name</id>
<type>{namespace}name</type>
<version>1.0.0.0</version>
<url>/WebPath</url>
<description>Description</description>
</Endpoint>
</BusinessSpaceRegistry>

```

#### BusinessSpaceRegistry/Endpoint/id

The ID of this endpoint information. This is what is references in the Registry Widget XML in the tag serviceEndpointRef/refId.

- BusinessSpaceRegistry/Endpoint/type  
Examination has shown that the type element and id element should be set to the same value.
- BusinessSpaceRegistry/Endpoint/version  
It is not known how the version is to be used but it appears that the value should be of the form "1.0.0.0" and must match the corresponding version in the catalog.
- BusinessSpaceRegistry/Endpoint/url  
A URL that will be the root of the iWidget. For example myWidget/. When building a Dynamic Web Project that contains a custom Widget, this will normally be the name of that project prefixed and followed by the "slash". For example /MyWidget/.
- BusinessSpaceRegistry/Endpoint/description  
A human readable description of this entry

### **JavaScript for a new widget**

When a widget is loaded by Business Space, that widget usually supplies some JavaScript code to be executed. As the Business Space widget is loaded and used, callbacks are executed into this code. It is this code that controls the user interactions and makes REST calls to back-end servers and updates the HTML of the page to display or change content. JavaScript does not natively have similar concepts to Java of package names and class names. Dojo provides a similar mapping using the two Dojo methods called `dojo.provide` and `dojo.declare`. Using these methods provides the ability to declare the Java equivalent of a Class type in a package. This is used with custom Widgets in the iWidget specification. In the iWidget definition, there is an attribute called `iScope`. This takes the *name* of the declared JavaScript *class*. When an instance of a Business Space widget is created on the page, a corresponding instance of the JavaScript class object is also created.

```

dojo.provide("com.kolban.SampleWidget");
dojo.declare("com.kolban.SampleWidget",
[dijit._Widget],
{
 "onLoad": function() {
 dojo.mixin(this,
 new com.ibm.bspace.common.util.widget.BSpaceCommonUtilityLoader());
 this.require("com.ibm.bspace.common.util.widget.BSpaceGeneralHelper");
 },
 "onUnload": function() {},
 "onReload": function() {},
 "onRefreshNeeded": function() {},
 "onSizeChanged": function() {}
});

```

## Functions

The JavaScript written to implement the widget contains callback functions that are called by the Business Space framework. The names of these functions are architected. What follows is a brief description of the different callbacks available.

- `onLoad`  
This appears to be called when the widget is loaded. This may not be the same as the `dojo.onLoad()`. The page may not be fully loaded at this time. It is likely that a call to `dojo.addOnLoad()` may be called here to ensure that the page is properly loaded.
- `onUnload`
- `onReload`
- `onAdd`
- `onDrag`
- `onDrop`
- `onModeChanged`
- `onSizeChanged`  
An object is passed as a parameter. The object contains:
  - `name: "onSizeChanged"`
  - o `payload:`
    - + `newHeight: int`
    - + `newWidth: int`
- `onNavStateChanged`
- `onItemSetChanged`
- `onRefreshNeeded`  
Called when the widget needs to be refreshed.

In addition, for each of the modes, a callback function is available for when the mode is selected:

`onview`

`onedit`

## Modes

A widget can be in any one of a number of modes. The common modes are view and edit. To switch mode, the `iContext` event called `onModeChanged` can be invoked.

```
this.iContext.iEvents.fireEvent("onModeChanged", null, "{newMode: 'view'}");
```

## ***HTML rewriting***

It is possible that there is HTML re-writing going on in the environment. I seem to see that the

widget.xml contains:

\_IWID\_

is being replaced with a UUID which I believe to be the iWidget ID (what ever that means). This same widgetID can be obtained from `iContext.widgetId`.

### ***Debugging iWidget JavaScript***

Assuming you are running the iWidget in FireFox with FireBug installed, adding the JavaScript code statement:

```
debugger;
```

causes a breakpoint to be reached which stops the execution and throws you into the FireBug JavaScript debugger.

Adding the JavaScript:

```
console.log("Text");
```

caused the text to be logged to the FireBug console.

### ***Event handling***

A widget can send (publish) and receive (handle) events from other widgets. IBM's supplied Business Space widgets both publish and handle events. This means that custom widgets can be wired together with the IBM widgets to react or cause reaction. In order for two widgets to be wired together, one needs to publish an event interface and the other needs to handle the exact same event interface. The configuration that describes published and handled events is performed in the iWidget XML document. There are two tags of interest. These are:

```
eventDescription
event
```

The `eventDescription` contains:

```
<iw:eventDescription id="{eventDescName}"
 payloadType="{payloadType}"
 description="{description}"
 lang="{locale}"
 title="{title}">
 <!-- one per locale -->
 <iw:alt description="{description}"
 title="{title}"
 lang="{locale}" />
</iw:eventDescription>
```

`id`

This appears to be a unique ID for the `eventDescription`. It must match the value used in the `eventDescName` in the `event` tag.

`payLoadType`

This describes the type of data that can be incoming.

`lang`

A locale (eg. "en")

`description`

A textual description of the event

`title`

Unknown. Although this shows up in the spec, it does not show up in the iWidget editor.

The `event` contains

```
<iw:event id="{eventName}"
 eventDescName="{eventDescName}"
 published="{boolean}"
 handled="{boolean}"
 handlerItemName="{attributeValue}"
 onEvent="{handlerName}" />
```

`id`

The identity of the event. This must match between event sender and receiver.

- `eventDescName`  
The name of the id value of a `eventDescription` tag.
- `published`  
A value of either true or false. If true, then this widget can be the sender/source of this type of event.
- `handled`  
A value of either true or false. If true, then this widget can be the receiver/destination of this type of event. If true, an `onEvent` method must be provided.
- `onEvent`  
The name of a JavaScript function in the `*.js` file for the widget that will receive the event.

To send an event one uses the method called:

```
iContext.iEvents.fireEvent()
```

The function defined in the `onEvent` attribute of the `event` tag accepts a single parameter defined as follows:

```
module iEvent
{
 String name;
 String type;
 Object payload;
 String source;
}
```

- `name`  
The name of the event, preferably a QName.
- `type`  
The type of any payload. If this is set to null, no information is being provided. This is useful for both simple signals or events with opaque payloads (i.e. a minor attempt at making it private)
- `payload`  
The data, if any, being provided by the source of the event.
- `source`  
The iWidget supplied name for the source iWidget. Note that some iContexts will consistently

null this field for security reasons.

When the JavaScript function named in the `<iw:event onEvent=...>` entry is called, it is passed a JavaScript Object as a parameter. This object contains at least the following:

- `name`  
The name of the event type that was received. This allows a single event handler to be able to process many types of handled event as the code can determine which event caused the callback.
- `payload`  
The data passed by the event publisher.

The iWidget specification provides a set of predefined events that callback into the JavaScript code without event specifications in the XML file having to be coded. These are:

- `onLoad` – No parameters. Called when the widget has been loaded and is "ready".
- `onUnload` – No parameters. Called when the widget is about to be unloaded.
- `onModeChanged` – Called with:

```
{
 "newMode": value
}
```

Called when the mode of the widget has changed

- `onSizeChanged` – Called when the size of the widget has changed. Parameters:

```
{
 "newWidth": value,
 "newHeight": value
}
```

- `onNavStateChanged` - ?
- `onItemSetChanged` - ?
- `onRefreshNeeded` - ?
- `onNewWire` - ?
- `onRemoveWire` - ?

### ***Dojo Level Workarounds***

The level of Dojo distributed with Business Space is back-level compared to the latest possible Dojo release. At the time of writing, the level of Dojo supplied is 1.4.3. Many of the expected Dojo functions simply aren't there. Here are some of them (but by no means all) and some suggested workaround:

None at this time.

### ***Debugging and Problem determination***

If after building a custom widget, things are not working as expected, here are some tips to follow to see what might be going wrong:

In the catalog file, there is an entry called <definition>. This defines a WEB path to the iWidget xml file. Open a browser and attempt to access this file. For example:

```
http://localhost:9080/PathToWidgetXMLFile
```

After making changes to the definitions, consider restarting the Business Space server. When changes actually take effect is not completely known.

When logging JavaScript objects, consider using the FireBug command:

```
console.debug(object)
```

This will log the object to the console in an expanded form that allows one to interrogate the object's contents very easily.

While developing Custom Widgets, changes are frequently made to the code files. If a browser caches code, then re-testing can be a challenge. In FireFox 3, the Function Key 5 (F5) causes a reload of the current page bypassing any cached files.

Assuming you are running the iWidget in FireFox with FireBug installed, adding the JavaScript code statement:

```
debugger;
```

causes a breakpoint to be fired when reached which stops the execution and throws you into the FireBug JavaScript debugger.

If the widget appears but there is no flex content, check that the expected SWF file is the one named.

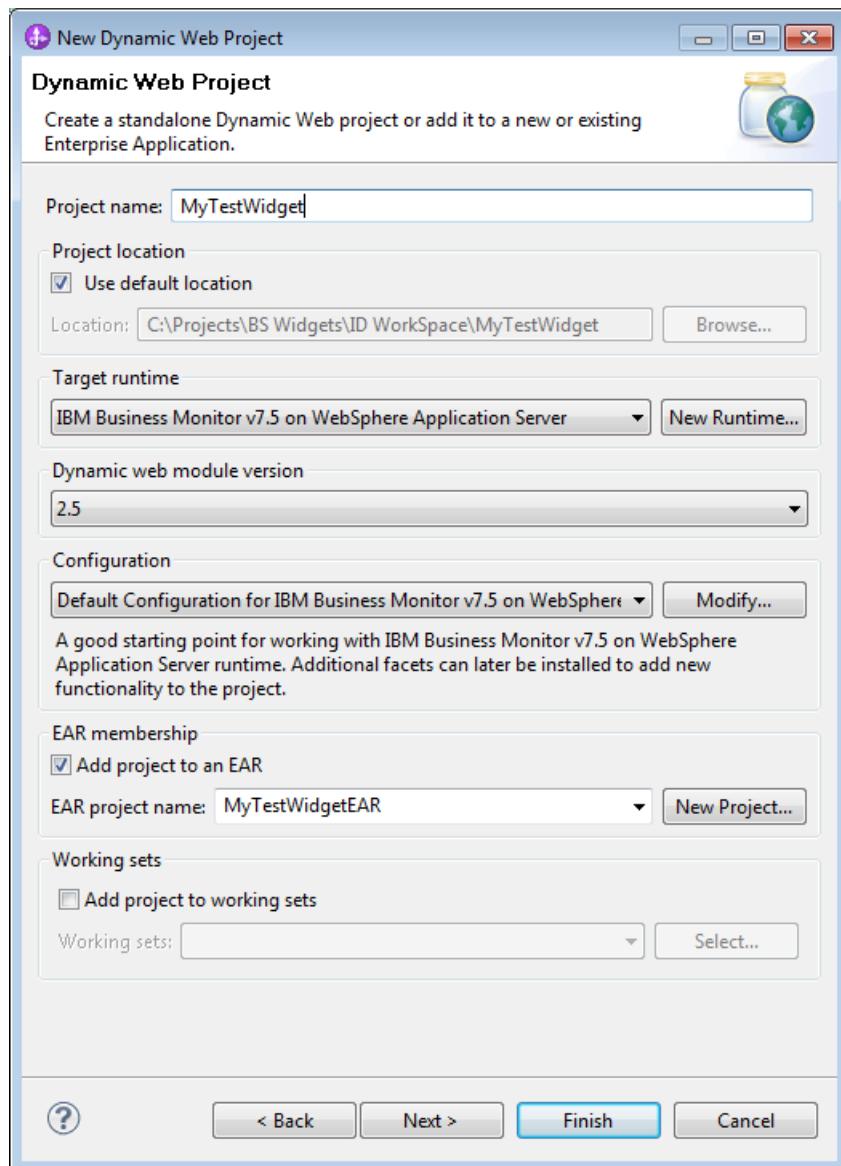
## See Also:

- DeveloperWorks – [Develop custom widgets for Business Space with Rational Application Developer or WebSphere Integration Developer](#) – 2009-12-20
- DeveloperWorks - [Integrating Flex applications with IBM Mashup Center](#) - 2009-06-29
- DeveloperWorks – [Creating a Flex-based widget to display WebSphere Business Monitor data in Business Space](#) – 2009-04-29
- [The iWidget 1.0 specification](#)
- [The iWidget 1.0 primer](#)
- [The iWidget 2.0 specification](#)
- [The iWidget Programming Guide](#)
- [Lotus Mashup home page for iWidget programming](#) – Includes links to the majority of iWidget documents
- [Train yourself to develop, configure, and debug widgets](#)
- RedBook - [Building IBM Business Process Management Solutions Using WebSphere V7 and Business Space](#) – 2010-06-10 - Contains large chapter on building custom widgets

## Custom Widget Walk through

In this section we will walk through the creation of a trivial custom widget from beginning to end to demonstrate the creation of such an entity. It assumes familiarity with the previous topics

We create a new Dynamic Web based project to hold our new widget. We call the project MyTestWidget and have it associated with a deployable EAR called MyTestWidgetEAR. This project will host the JavaScript and iWidget XML file.



Now we need to create the JavaScript file that will be called to handle events.

```
dojo.provide("com.sample.MyTestWidget");

dojo.declare("com.sample.MyTestWidget", null, {
 onLoad: function ()
 {
 console.log("onLoad called");
 },
 onUnload: function()
 {
 console.log("onUnload called!");
 },
 onview: function()
 {
 console.log("onview");
 },
 onedit: function()
 {
 console.log("onEdit called!");
 }
});
```

```

} ,

onReload: function() {
 console.log("onReload called");
} ,

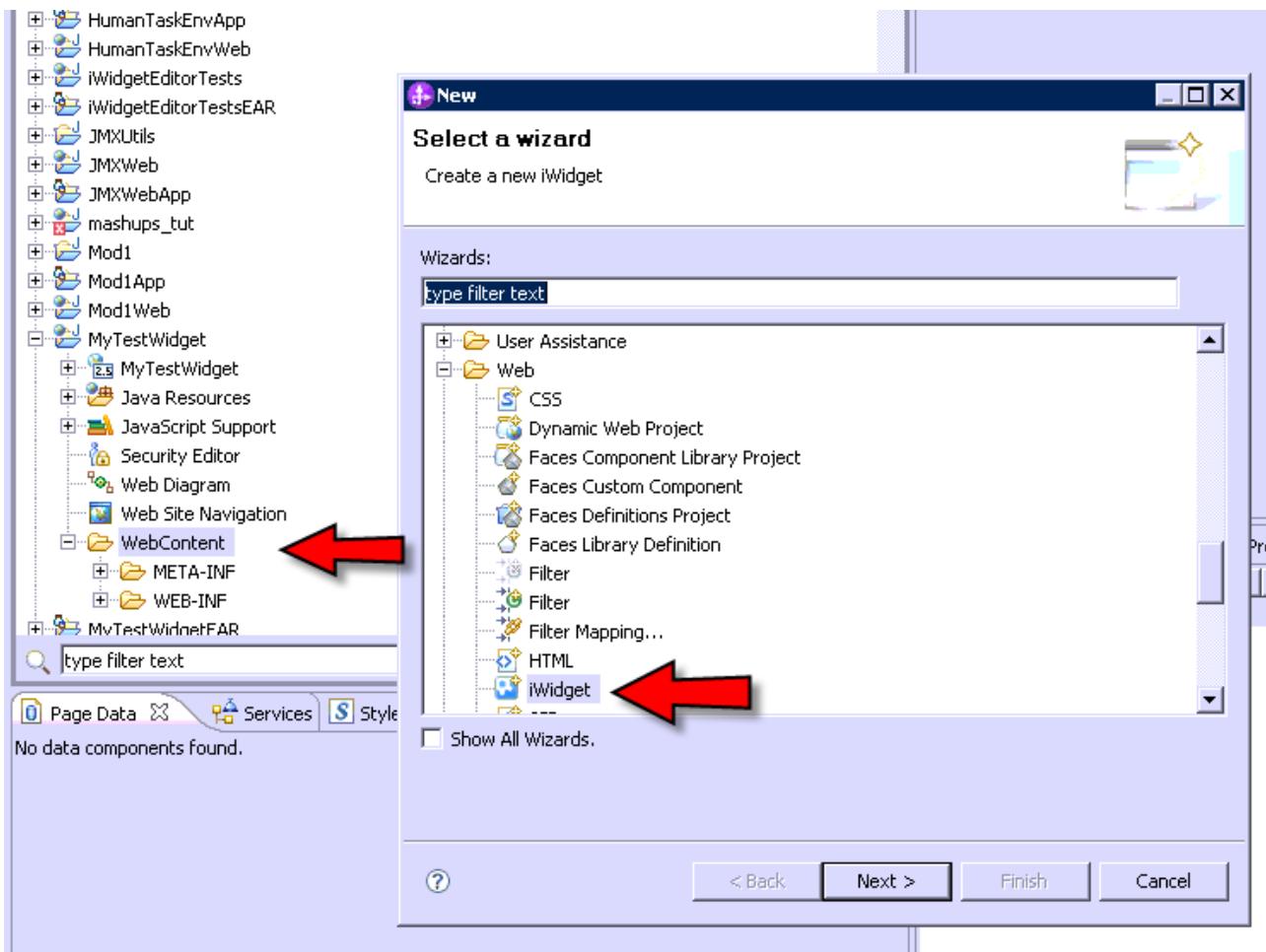
onRefreshNeeded: function() {
 console.log("onRefreshNeeded called!");
 console.log("Root: " + this.getWidgetRootURI());
 console.log("here: " + this.iContext.io.rewriteURI(""));
} ,

onSizeChanged: function(event){
 console.log("onSizeChanged called");
 console.log("New sizes: width=" + event.payload.newWidth + " height=" +
event.payload.newHeight);
} ,

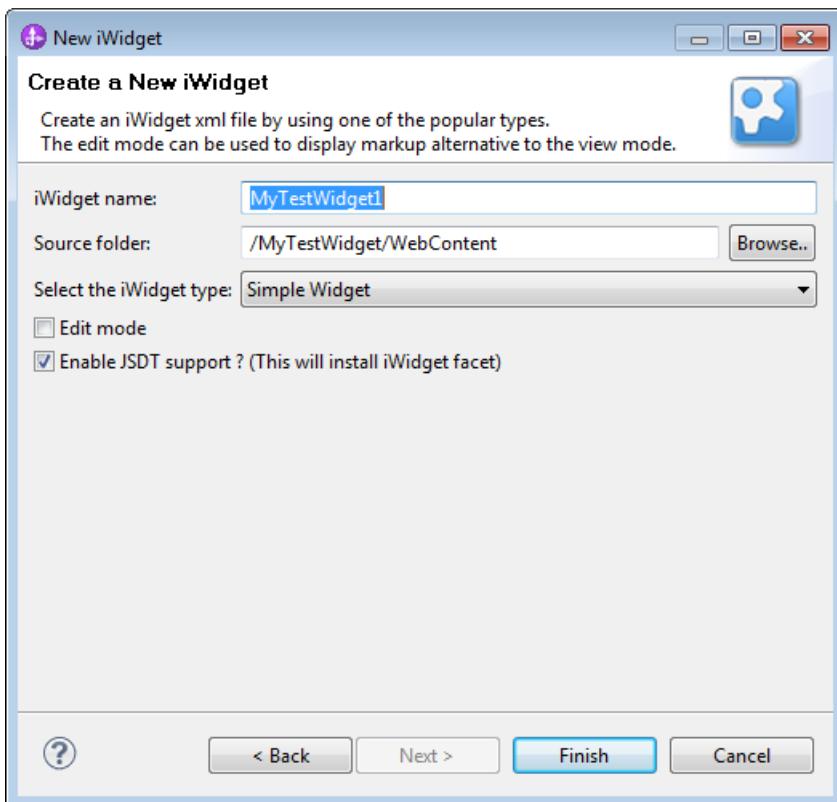
$$/*
 * Sample Event handler ...
 */
handleEvent: function(event)
{
 console.log("handleEvent called! : " + event.payload);
}
);$$

```

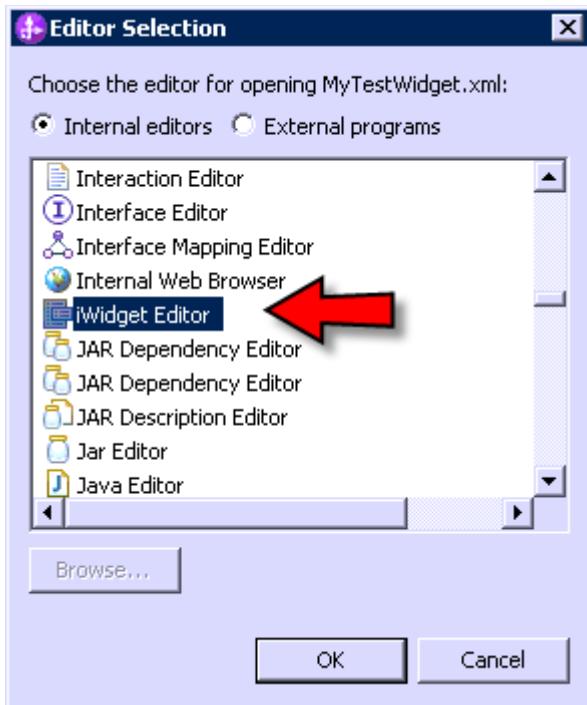
Create the iWidget XML file. Right click in the Web Content folder of the new project and select create New > Other. From the Web folder of the New dialog, find and select iWidget.



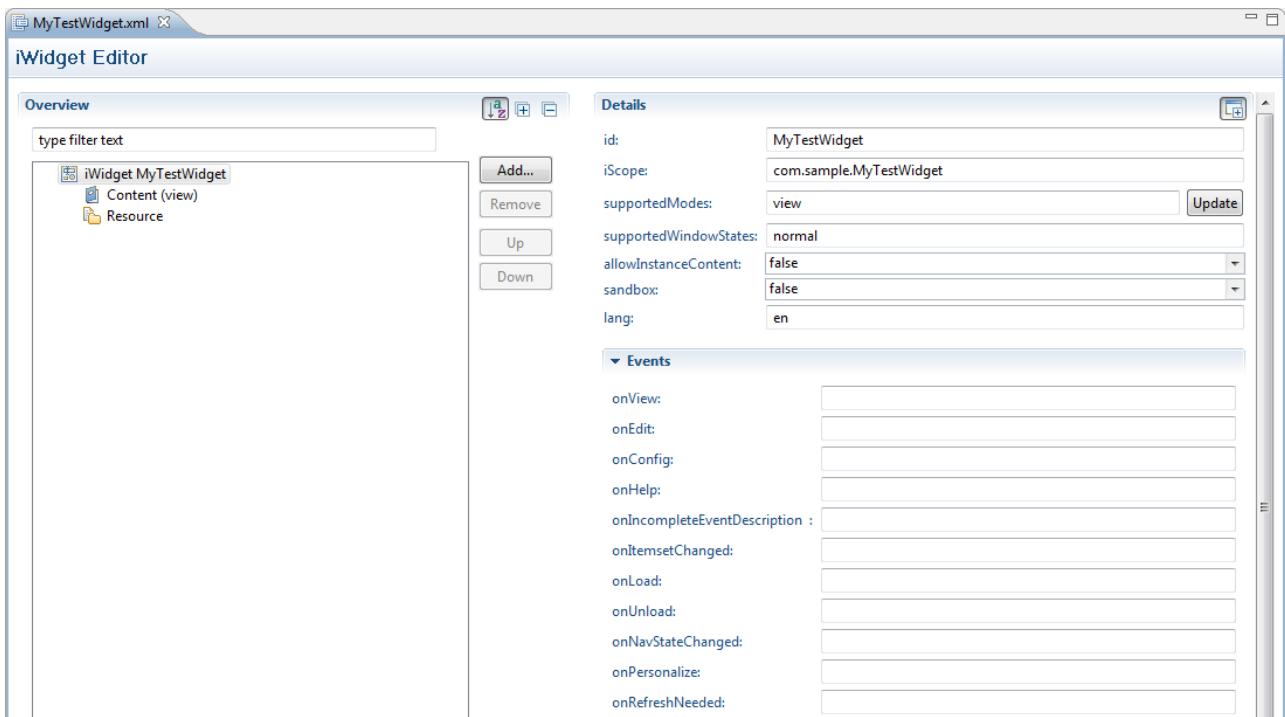
The name of the new iWidget should be MyTestWidget.



Open the newly created iWidget XML file with the iWidget editor. This may have to be done with the Open > Other and then select the editor explicitly.



Give values to some of the iWidget attributes such as name and iScope. The iScope parameter MUST have the Dijit name of the widget. Save the result and close the editor.



Here is the data contained in the iWidget XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<iw:iwidget id="MyTestWidget" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 xmlns:iw="http://www.ibm.com/xmlns/prod/iWidget"
 supportedModes="view" lang="en" iScope="com.sample.MyTestWidget">
 <iw:content mode="view">
 <![CDATA[
 <div>Hello World</div>
]]>
 </iw:content>

 <iw:resource src="MyTestWidget.js"/>
</iw:iwidget>
```

Create a General Project called MyTestWidgetPackage. This project will be used to hold Business Space widget installation and packaging artifacts.

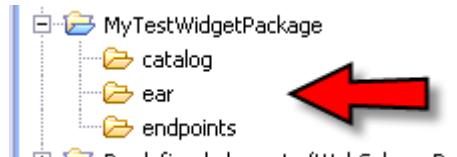
In the new project, create three simple folders called:

catalog

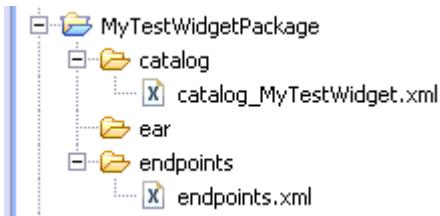
ear

endpoints

The result should look as follows:



In the catalog folder, create a file called catalog\_MyTestWidget.xml and in the folder called endpoints create a file called endpoints.xml. These should be created as simple files.



Copy the following XML fragment into the content of the file `catalog_MyTestWidget.xml`. This fragment is a template for what we need and will be further edited.

```

<?xml version="1.0" encoding="UTF-8"?>
<catalog id="myWidget">
 <resource-type>Catalog</resource-type>
 <category name="myWidget">
 <title>
 <nls-string lang="en">My Widget</nls-string>
 </title>
 <description>
 <nls-string lang="en">My Widget description.</nls-string>
 </description>
 <entry id="{mynamespace}myWidget" unique-name="{mynamespace}myWidget">
 <title>
 <nls-string lang="en">My Widget Title 2</nls-string>
 </title>
 <description>
 <nls-string lang="en">My Widget Description 2</nls-string>
 </description>
 <definition>/WebRoot/myWidget.xml</definition>
 <preview>/WebRoot/images/preview_myWidget.gif</preview>
 <icon>/WebRoot/images/icon_myWidget.gif</icon>
 <previewThumbnail>/WebRoot/images/thumb_myWidget.gif</previewThumbnail>
 <shortDescription>
 <nls-string lang="en">My Short Description</nls-string>
 </shortDescription>
 <metadata name="com.ibm.bspace.version">1.0.0.0</metadata>
 <metadata name="com.ibm.bspace.owner">IBM</metadata>
 <metadata name="com.ibm.bspace.serviceEndpointRefs">
 [
 {
 "name": "serviceUrlRoot",
 "required": "false",
 "refId": "endpoint://myWidget",
 "refVersion": "1.0.0.0"
 }
]
 </metadata>
 </entry>
 </category>
</catalog>

```

Change the following:

All references to `myWidget` to be `MyTestWidget`

All references to `WebRoot` to be `MyTestWidget`

The result will be:

```

<?xml version="1.0" encoding="UTF-8"?>
<catalog id="MyTestWidget">
 <resource-type>Catalog</resource-type>
 <category name="MyTestWidget">
 <title>

```

```

 <nls-string lang="en">My Test Widget</nls-string>
 </title>
 <description>
 <nls-string lang="en">My Test Widget description.</nls-string>
 </description>
 <entry id="{mynamespace}MyTestWidget" unique-
name="{mynamespace}MyTestWidget">
 <title>
 <nls-string lang="en">My Test Widget Title 2</nls-string>
 </title>
 <description>
 <nls-string lang="en">My Test Widget Description 2</nls-string>
 </description>
 <definition>/MyTestWidget/MyTestWidget.xml</definition>
 <preview>/MyTestWidget/images/preview_myWidget.gif</preview>
 <icon>/MyTestWidget/images/icon_myWidget.gif</icon>

<previewThumbnail>/MyTestWidget/images/thumb_myWidget.gif</previewThumbnail>
 <shortDescription>
 <nls-string lang="en">My MyTestWidget Short Description</nls-string>
 </shortDescription>
 <metadata name="com.ibm.bspace.version">1.0.0.0</metadata>
 <metadata name="com.ibm.bspace.owner">IBM</metadata>
 <metadata name="com.ibm.bspace.serviceEndpointRefs">
 [
 {
 "name": "serviceUrlRoot",
 "required": "false",
 "refId": "endpoint://mynamespace}MyTestWidget",
 "refVersion": "1.0.0.0"
 }
]
 </metadata>
 </entry>
</category>
</catalog>
```

Copy the following XML fragment into the endpoints.xml file:

```

<?xml version="1.0" encoding="UTF-8"?>
<BusinessSpaceRegistry
 xmlns="http://com.ibm.bspace/BusinessSpaceRegistry"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://com.ibm.bspace/BusinessSpaceRegistry
BusinessSpaceRegistry.xsd ">
 <Endpoint>
 <id>{mynamespace}myWidget</id>
 <type>{mynamespace}myWidget</type>
 <version>1.0.0.0</version>
 <url>/WebRoot/</url>
 <description>Location of MyWidget</description>
 </Endpoint>
</BusinessSpaceRegistry>
```

Change all occurrences of myWidget to MyTestWidget and the WebRoot to also be MyTestWidget.  
The result will be:

```

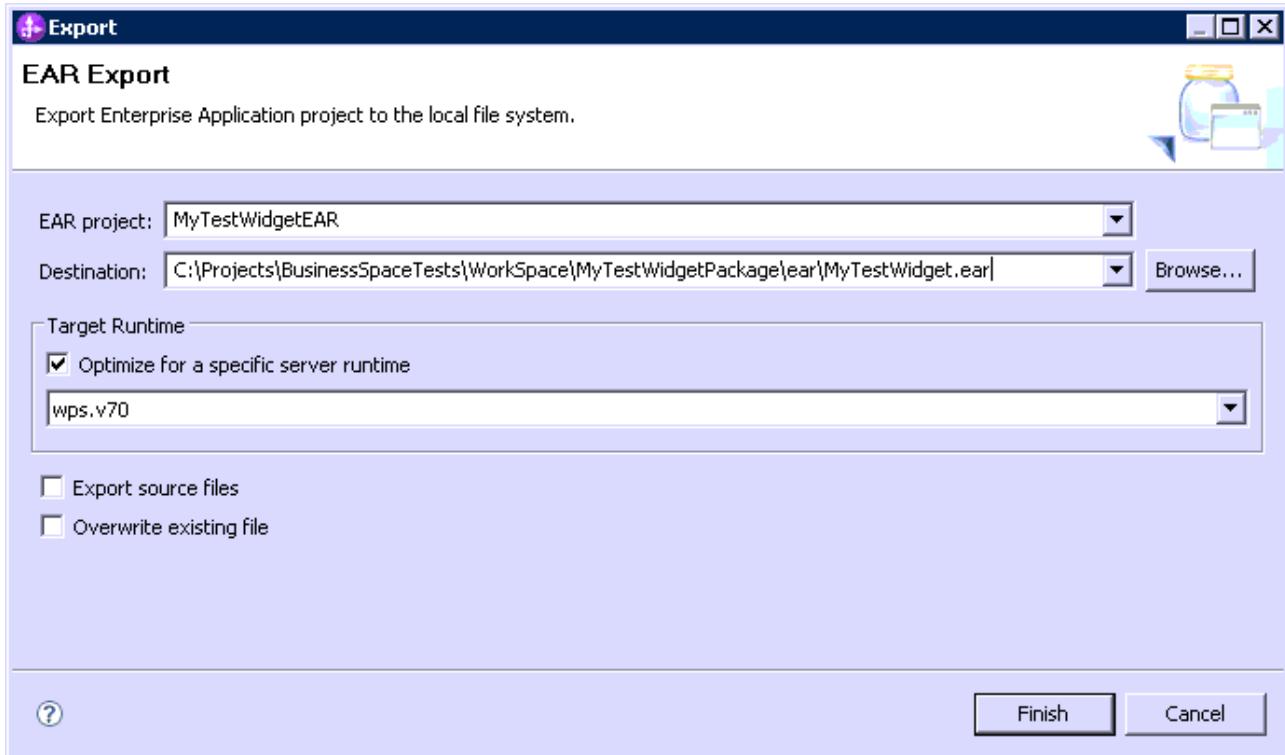
<?xml version="1.0" encoding="UTF-8"?>
<BusinessSpaceRegistry
 xmlns="http://com.ibm.bspace/BusinessSpaceRegistry"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://com.ibm.bspace/BusinessSpaceRegistry
BusinessSpaceRegistry.xsd ">
 <Endpoint>
 <id>{mynamespace}MyTestWidget</id>
```

```

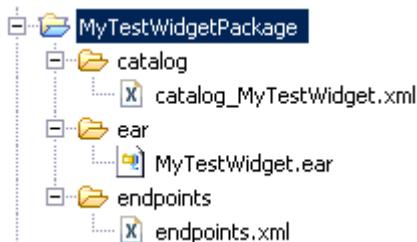
<type>{mynamespace}MyTestWidget</type>
<version>1.0.0.0</version>
<url>/MyTestWidget/</url>
<description>Location of MyWidget</description>
</Endpoint>
</BusinessSpaceRegistry>

```

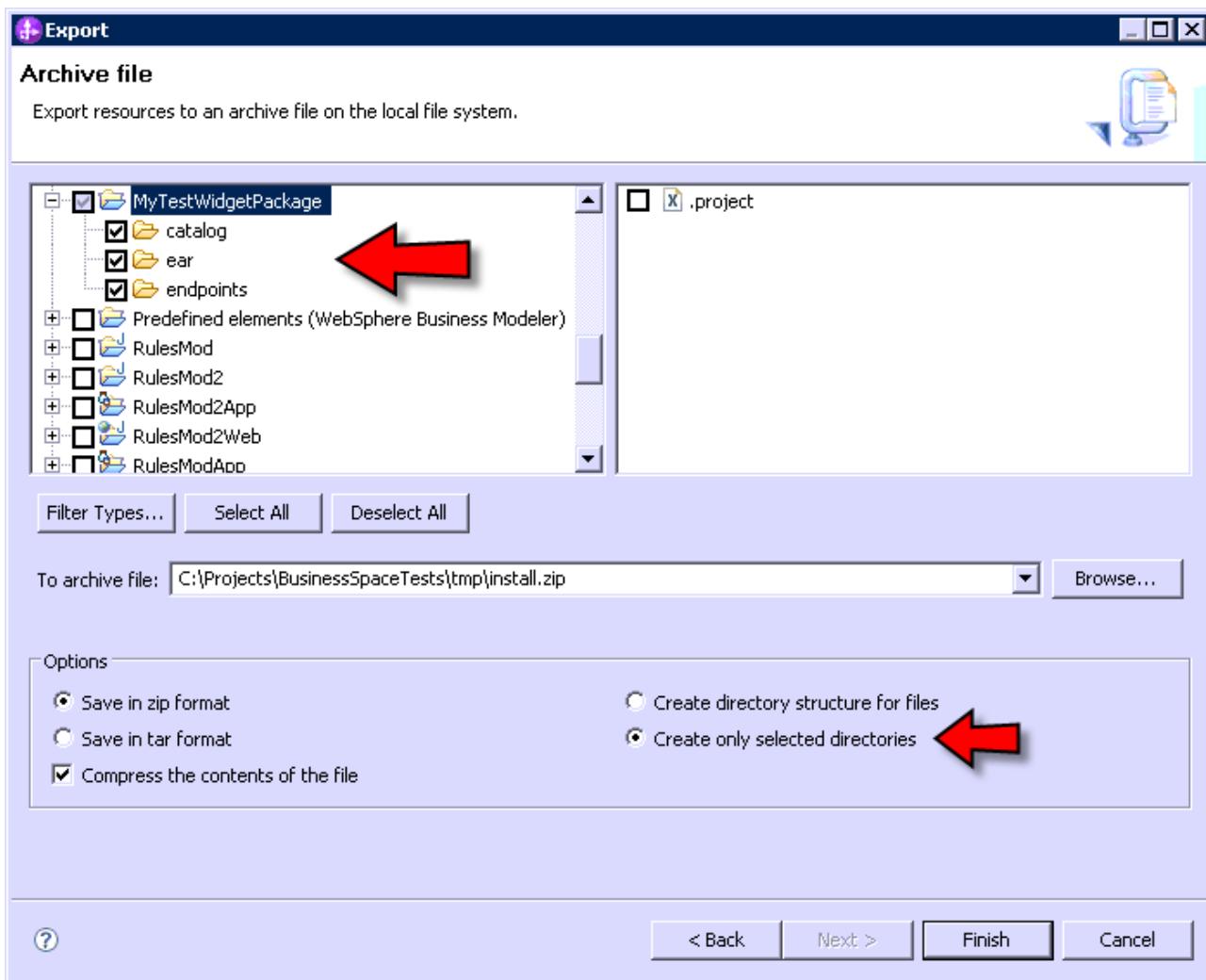
Now we want to generate the EAR file for the MyTestWidget Web project and save the result in the file system underneath the ear folder in the packaging project.



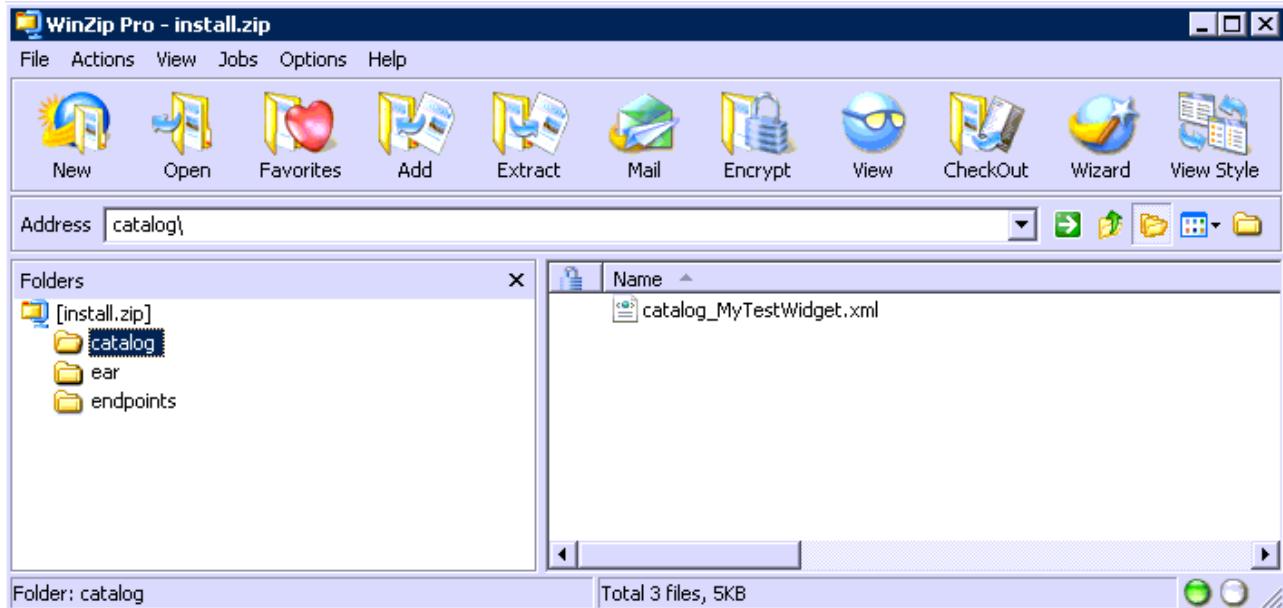
Refreshing the packaging project now shows that the EAR is contained in the ear folder:



Generate a ZIP file containing just these folders and their contents. This can be done through the ID Export option. Remember to select "Create only selected directories" to ensure that no extra directories are created that we do not want.

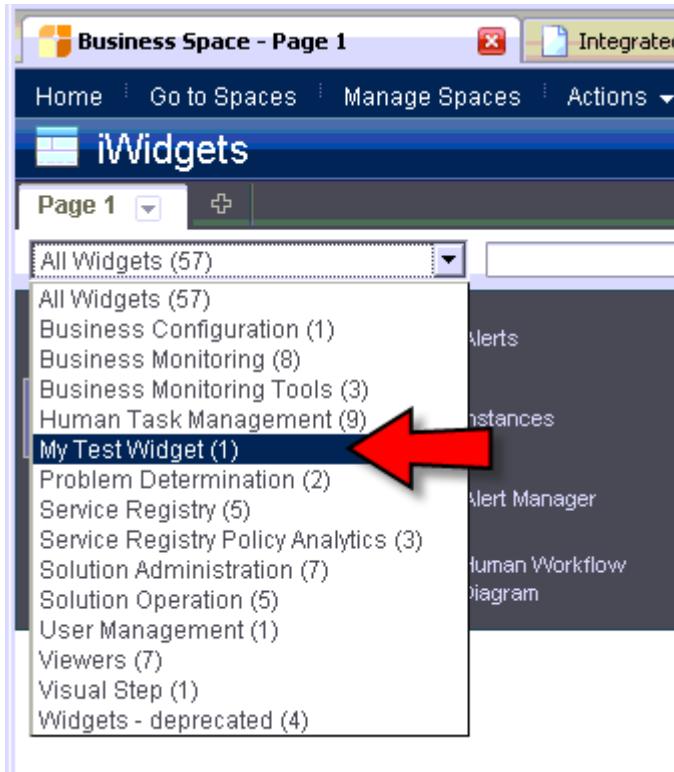


If examined in a ZIP tool, the following would be shown:

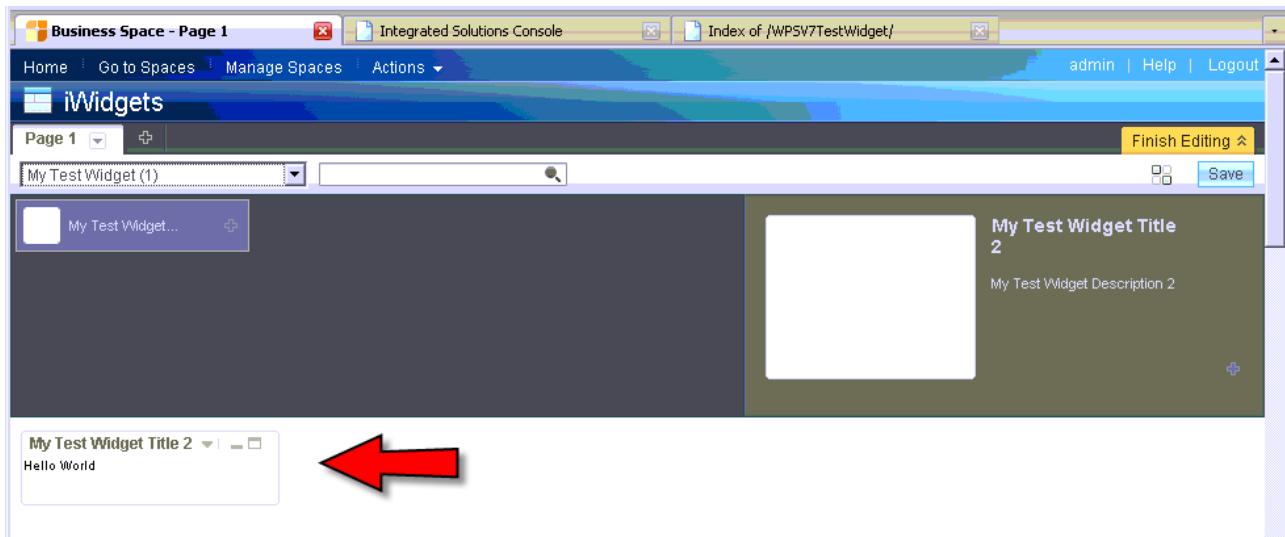


Once the ZIP file has been created, the Widget is ready to be installed. Use the wsadmin script to install the widget. Make sure that the script is pointing to the ZIP file that was just created. See

Registering Widgets. After installation has been completed, restart the WPS server to ensure that all changes have taken effect. Once the server has been restarted, Business Space can be launched. In the page editing section, we will now see a new widget that can be added to a page:

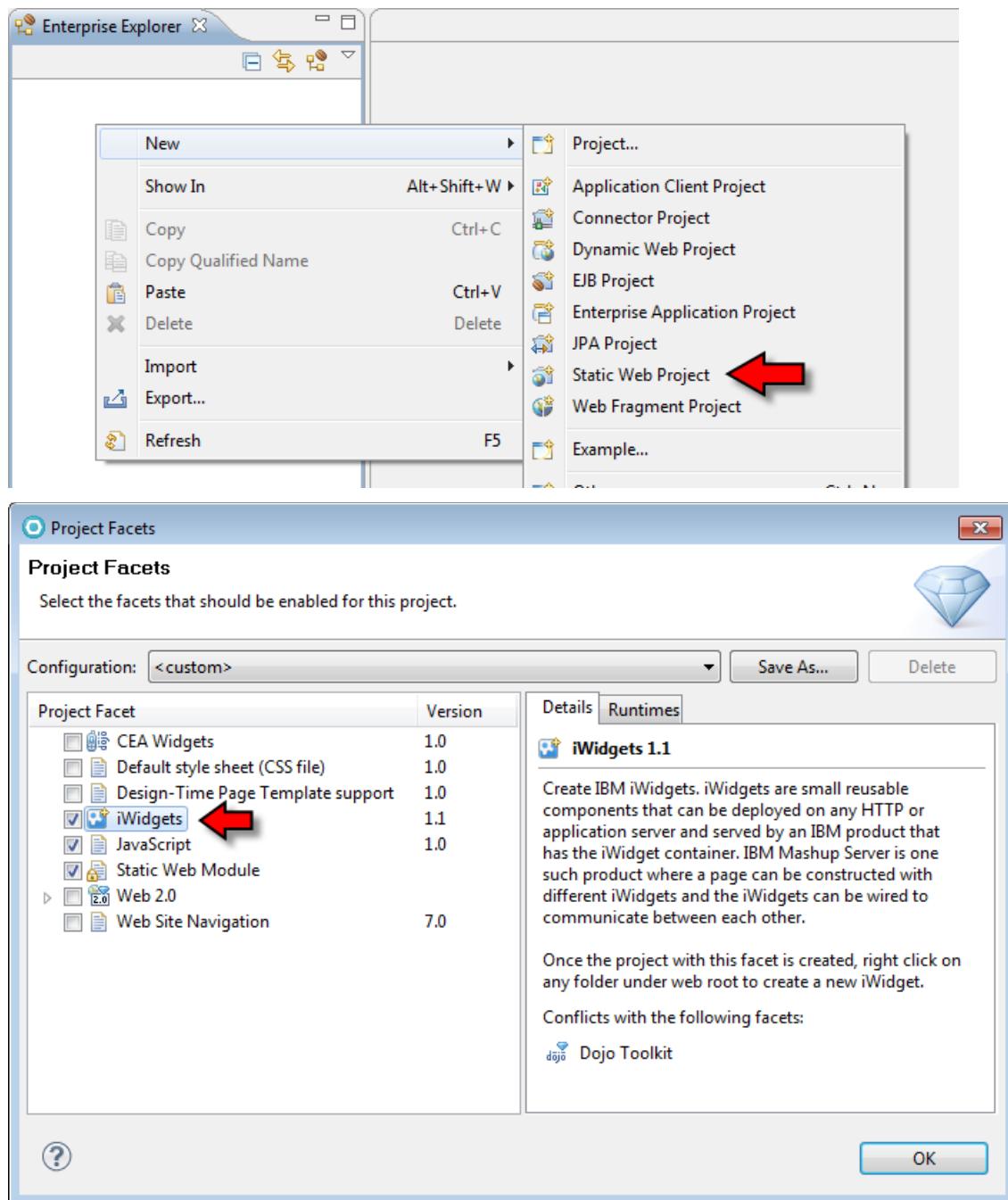


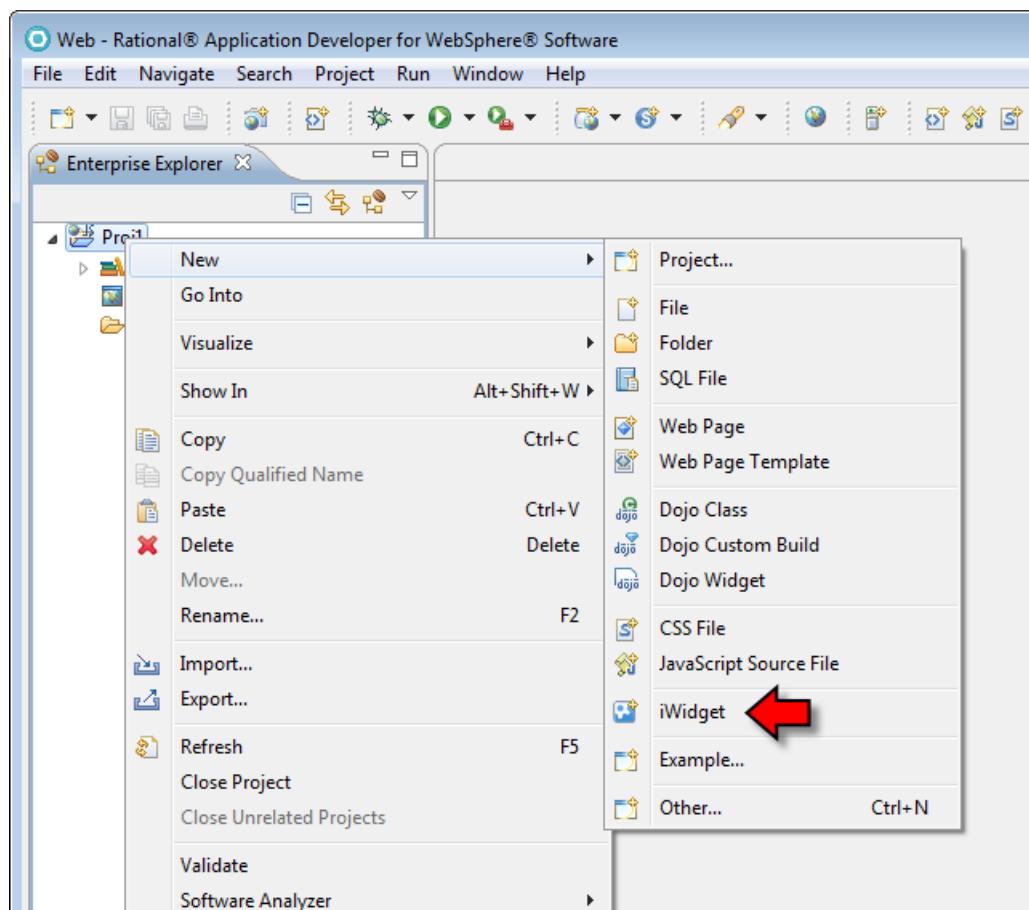
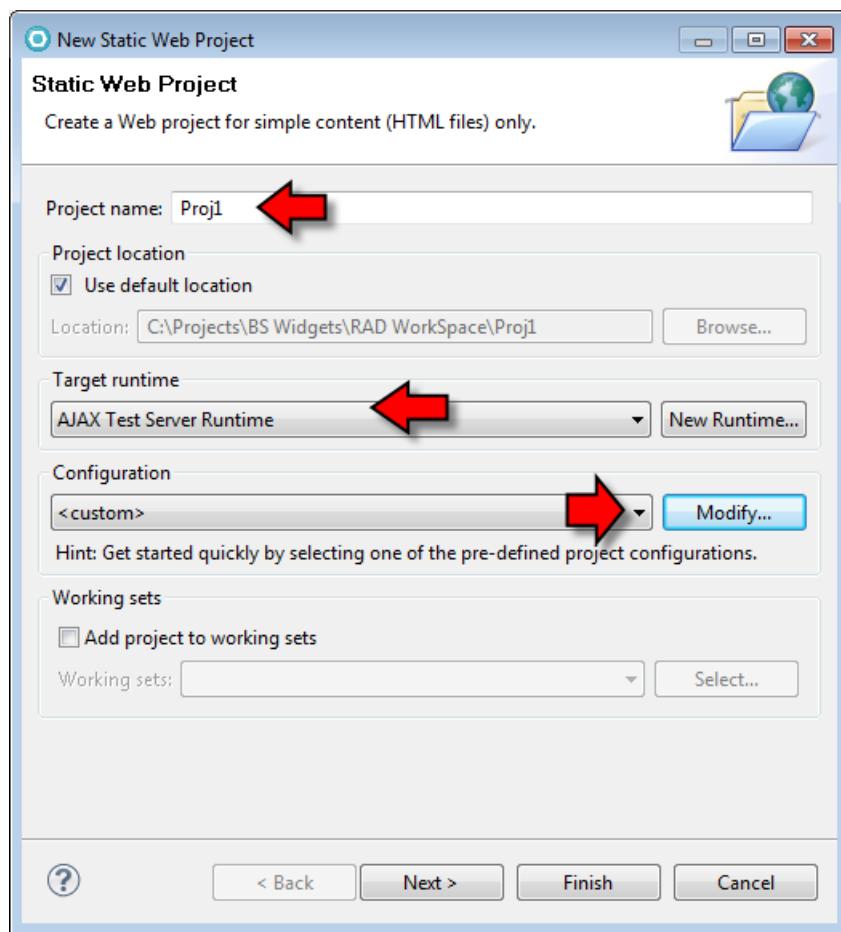
Once added, we can see it in place within the Business Space environment:

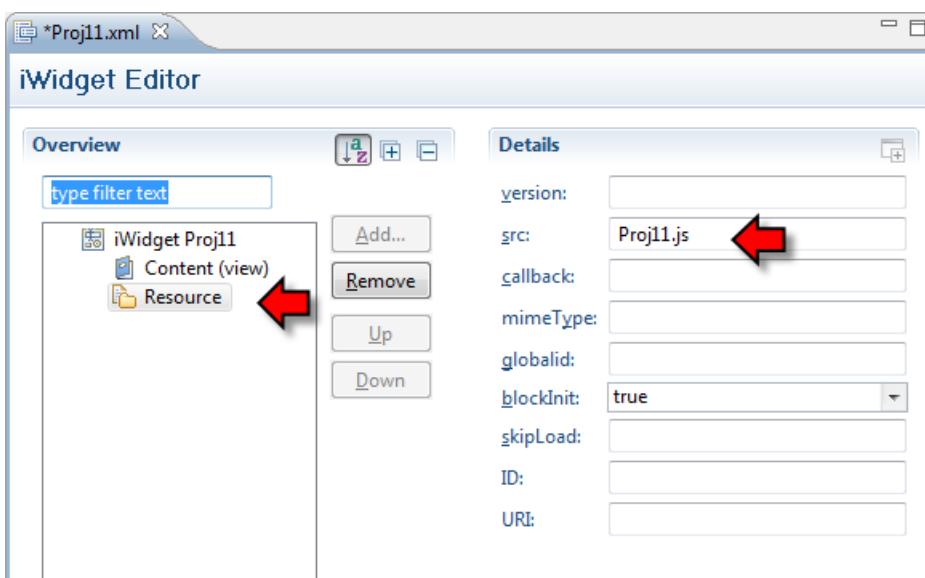
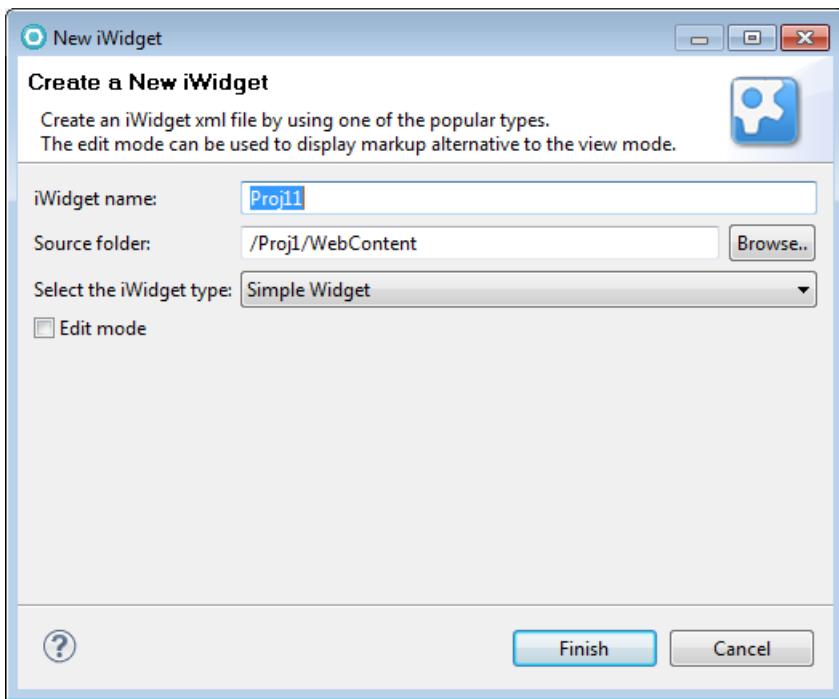


It isn't a very exciting widget ... but it is indeed a custom widget.

## Using RAD 8.0.3 to create an iWidget







## Mashup Center

See also:

- [IBM Mashup Center home page](#)

## Multiple Browser instances and BPM

When developing or working with BPM solutions, it is not uncommon to want to work with multiple browsers open on your desktop to a variety of BPM applications. Such applications can include:

- Business Space
- BPC Explorer
- WAS Admin Console

- Business Rules Manager
- others ...

In a secure WAS environment, each of these applications requires you to authenticate with the target WAS server before the application can be used. Unfortunately, this can cause a problem. To understand this, let us examine how WAS authentication through a browser works.

When you login to WAS and provide a user name and password, that data is sent to WAS only once. WAS validates that the user name and password match and generates a security **token**. This token is probably no more than a long sequence of characters. When the browser subsequently interacts with WAS, the token is passed back from the browser back to WAS. On seeing this token, WAS now knows who you are and trusts the previous authentication. This means that the userid and password only ever flowed to WAS once. All transmissions between the browser and WAS occur over the encrypted HTTPS (SSL) transport and hence the content of the token is never exposed on the network and can't be sniffed and replayed by other users.

When the token is originally generated by WAS and sent back to the browser, the token is saved locally by the browser in the form of a **cookie**. This cookie containing the token is what is sent back to WAS when the browser makes subsequent calls back to WAS.

So far ... no problems.

Now imagine bringing up two browser windows or tabs running in the same browser. If we sign-on to a WAS based application on one window, a token is created and saved in the local cookie store. If we sign-on to WAS in the second window, again a token is created and saved in the local cookie store. The problem is ... is that there is only **one** cookie store shared by all browser windows/instances. So if you sign-in as user "A" in one browser instance and then sign-in as user "B" in a second browser instance, the token/cookie for user "A" is replaced with the token/cookie for user "B".

From a users perspective, this manifests itself as either getting unexpected results when working with WAS applications in different browser windows or else we get additional requests to authenticate as the cookies are constantly being expired and refreshed.

Fortunately, there is a solution.

When we use the FireFox browser, each instance of the browser can have its own private **profile**. Think of this as a complete set of configurations including the storage area where cookies are saved. If we want to have multiple browsers up and running then we can launch multiple instances of FireFox, each with its own profile.

The FireFox program has two command line flags that should be supplied:

```
-ProfileManager
-no-remote
```

The first parameter causes FireFox to display the ProfileManager dialog which allows us to create and select the profile to be used.

The second parameter forces FireFox to use a second profile even if an existing instance of FireFox is already running. Without this flag, a second profile will be ignored and FireFox will use the profile in effect for the first instance started.

The ProfileManager dialog is shown next:



Use the Create Profile button to create as many profiles as desired. To start FireFox with a specific profile, select it and click the Start Firefox button.

Note that multiple tabs in a single instance of FireFox will always share the same profile so utilize multiple profiles and hence multiple browser instances to keep them separate.

## ***Business Space Supplied Widgets***

Business Space comes with a variety of Widgets. As other BPM products are installed, so too are additional sets of Widgets.

## **Business Configuration**

### ***Business Rules***

### **Human Task Management**

The Human Task Management category of Business Space contains the following widgets:

- Business Categories
- Business Category Information
- Escalations
- Inbox
- My Team's Tasks
- My Work Organizer
- Process Definitions
- Processes
- Process Information
- Task Definitions
- Task Filter
- Task Information
- Tasks
- Work Basket Information

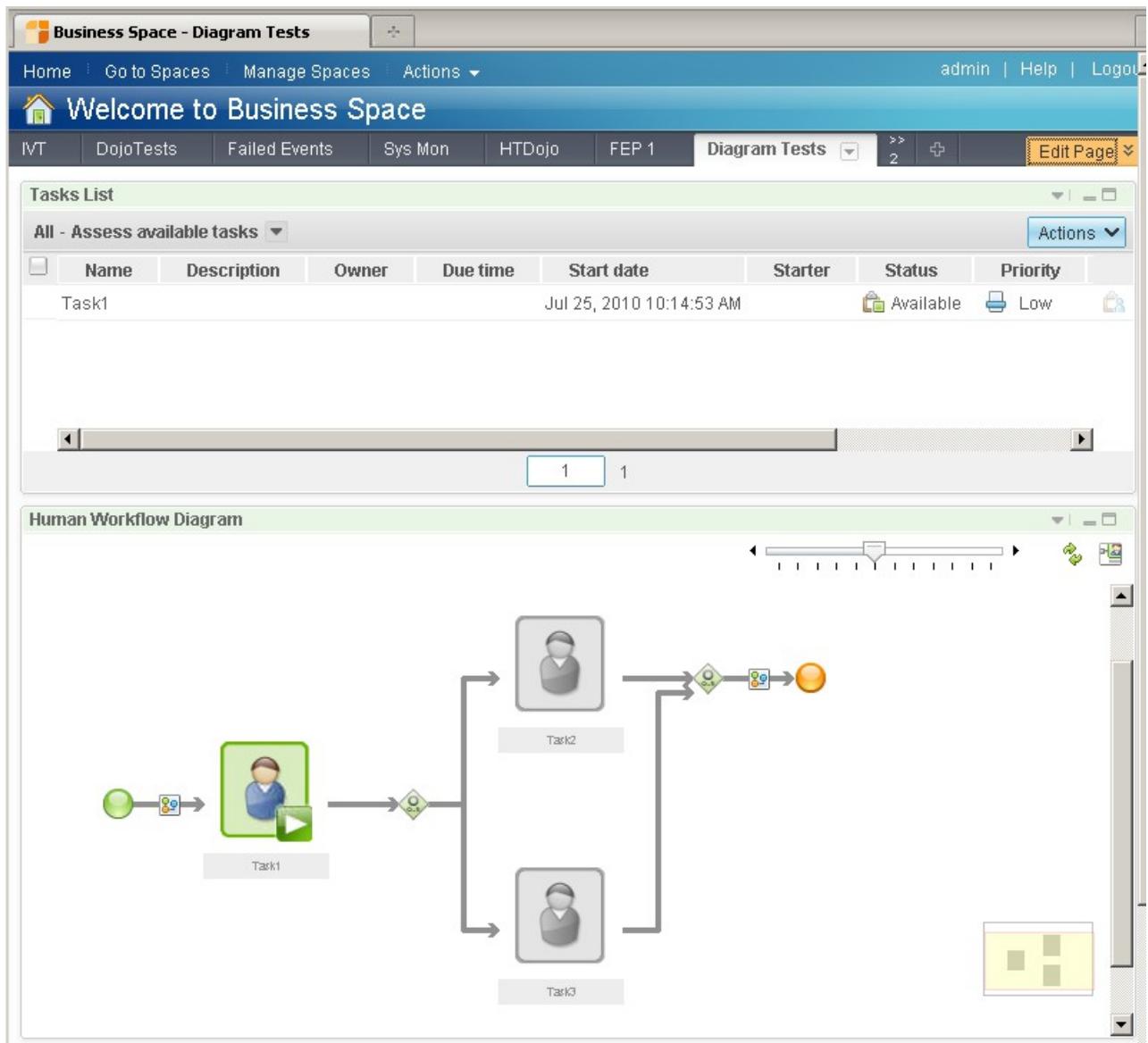
- Work Baskets

## ***Escalations List***

### ***Human Workflow Diagram***

A BPEL process can contain a set of Human Tasks. The Human Workflow Diagram widget allows us to visualize the human tasks within instances of BPEL processes and see where we are within the process with respect to the human tasks being executed. It dynamically examines the process associated with a selected task and draws a picture of where that task is within the process relative to other tasks.

The following illustrates an example of the Human Workflow Diagram.



The widget responds to the following events:

- Open – com.ibm.widget.Open
- Close – com.ibm.widget.Close

- Tab Changed – com.ibm.widget.TabChanged

An example wiring of this widget would be to a Tasks List widget with:

Tasks List (Item Selected) → Human Workflow Diagram (Open)

### ***My Team's Tasks***

### ***My Work Organizer***

### ***Process Definitions List***

### ***Processes List***

### ***Task Definitions List***

### ***Task Information***

### ***Tasks List***

Action Requested	
Focus Changed	
Items Selected	
Task Released	
Task Delegated	
Task Terminated	
Task Deleted	
Task Claimed	
Escalation Started	

## **Problem Determination**

## **Solution Administration**

## **Solution Operation**

## **User Management**

## **Viewers**

### ***Web Viewer***

The Web Site viewer widget displays the contents of a web site. The web site displayed can be configured in the settings for the widget or can be passed in via an event definition.

### ***Script Adapter***

The Script Adapter widget can be wired to receive events published by any other widget. When it receives an event, it can then forward that event onwards to any other wired widget. The value of this is two-fold. First, it can be used during widget development for debugging. When one widget publishes an event, the fact that the event was published and its payload can be logged by the Script Adapter. Secondly, JavaScript source code can be supplied as a configuration parameter to the Script Adapter widget. This JavaScript can operate against the payload data and potentially massage or transform its content. The passed in data is available in a variable called `payload`. Payload will be a String. If it contains JSON encoded data, then the data will have to be expanded. An example of this is:

```
var myObject = eval('(' + payload + ')');
```

## **Visual Step**

## ***JSPs***

See Also:

- DeveloperWorks - [Building a human task-centric business process with WebSphere Process Server, Part 2: Customizing JSPs](#) - 2007-03-21

## ***WebSphere Portal***

See also:

- developerWorks - [Realizing the value of IBM Business Process Manager in the WebSphere Portal environment, Part 1: Integrating WebSphere Portal with IBM Business Process Manager](#) - 2011-09-29

## Liferay

Liferay is an Open Source Portal environment.

### Installing Liferay on WAS

Liferay can be installed on WAS v8 environment. Since we are working with IBM BPM, chances are good that our familiarity with WAS v8 will be higher than other Java EE platforms.

We start our installation by creating a new WAS profile. I'll assume you know how to do that. Create the profile with no special augmentations.

Download the Liferay package without a bundled Java EE environment. This can be found in the Additional Files page at the Liferay web site:

The screenshot shows the Liferay website's 'Get Liferay Portal' page. On the left, there's a sidebar with a 'DOWNLOADS' section containing links for Liferay Portal, Social Office, Sync, Projects (Alloy UI, FACES, IDE), and a 'Releases' link. The main content area has a title 'Get Liferay Portal'. Below it, a paragraph describes the two versions of Liferay Portal. A navigation bar at the bottom of the content area has three tabs: 'Releases', 'CE vs EE', and 'Additional Files'. A large red arrow points to the 'Additional Files' tab. The 'Additional Files' section contains a heading 'Additional Files' and a paragraph about optional files. It includes a link to 'Liferay Portal 6.1 GA 2 Dependencies' and a note about application servers. Another red arrow points to this note. The section also contains a heading 'Dependencies' and a note about integrating with external systems. A third red arrow points to the 'Dependencies' heading. The 'Mobile Device Detection Plugin' section is also visible, with a fourth red arrow pointing to its heading.

A Zip file called "liferay-portal-dependencies" is supplied which contains JAR files that need to be added to the WAS class path. As of writing, these are:

- hsql.jar
- portal-service.jar
- portlet.jar

These should be copied to <WAS>/lib/ext

Now we start the WAS Server. Experience has shown that the default max heap size of 256MB is

not enough. I increased to 512MB. Once started, bring up the WAS Admin console and install the Liferay WAR file. Accept all the default options. The WAR is pretty large, it make take a few moments to upload and then parse for installation.

## **Using the Liferay Tomcat bundle**

One of the bundles available for Liferay is an integrated Tomcat environment. After extracting the content to a folder, we can launch the Tomcat environment using:

```
<Liferay>/tomcat/bin/startup
```

## **Developing Liferay Portlets**

First we will want to install the Liferay development environment. The full recipe for this can be found in the Liferay documentation but here are some notes that I used to get it working. First I downloaded Eclipse for Java EE 4.2.1. Once launched, I used the Eclipse in-built updated to install the Liferay IDE from the following Eclipse update site:

<http://sourceforge.net/projects/lportal/files/Liferay%20IDE/1.6.1/updatesite/>

Liferay supports the JSR-286 Portlet Specification

See also:

- [Understanding the Java Portlet Specification 2.0 \(JSR 286\): Part 1, Overview and Coordination Between Portlets](#) – 2010-01
- [JSR 168 Portlets & JSR 286](#)

## ***Lotus Forms***

See Also:

- DeveloperWorks – [Build and deploy a business process model using WebSphere Business Process Modeler Advanced and Lotus Forms, Part 3: Implement a mediation flow](#) – 2009-10-14

## ***Dojo***

### **Custom Dojo development environment**

At times we want to work with the Dojo generated by WID. WID places these files in a Web Project for deployment to the WPS server for subsequent retrieval by Business Space. If we want to modify the generated HTML files, we find that we are constantly redeploying the Web project so that the changes become visible. During development of UIs, we constantly find ourselves wishing to make changes so this update/publish sequence can be annoying. Fortunately, there is an elegant solution.

When the web project containing the HTML is published to WPS, the files for the HTML can be accessed from the file system under the WPS profile directory. Changes made to these files are immediately picked up.

In the following walk-through, assume that the Web Project hosting the Dojo/HTML files is called `DojoHolder` and that an example HTML file is called `MyHTML.html`. `DojoHolder` is hosted by the EAR called `DojoHolderApp`.

If we navigate to:

```
{WPS Profiles}/{Profile Name}/installedApps/{Cell
Name}/DojoHolderApp.ear/DojoHolder.war
```

we will find the files in the `DojoHolder` Web Project. Editing these will provide real-time changes to the files as served by WPS.

Rather than editing these files with notepad or some similar editor, we ideally want to edit them with a JavaScript editor such as is found in WID. It has been found that if we create a new WID Static Web project we can then link in these file system files to the web project and edit within WID. This Web Project will **never** be published and is only a wrapper to allow us access to the files.

See also:

- DeveloperWorks – [Developing custom widgets for Business Space using Dojo, Part 1: Generating Dojo markup using a generic markup handler](#) – 2009-12-09
- DeveloperWorks – [Integrating a Dojo client with an SCA application via SCA HTTP binding](#) – 2009-11-25
- DeveloperWorks – [Customizing HTML-Dojo forms for Business Space powered by WebSphere](#) – 2009-04-09

See also:

- DeveloperWorks - [Integrating IBM WebSphere Portal V7 with IBM WebSphere Process Server V7 using the WebSphere Process Server client](#) - 2010-09-29
- DeveloperWorks – [Building a process task portlet application using the Portlet Generator in WebSphere Integration Developer 6.2](#) – 2009-04-08
- DeveloperWorks – [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 1: Solution overview](#) – 2008-04-16
- DeveloperWorks – [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 2: Developing a human-centric business process](#) – 2008-04-16
- DeveloperWorks – [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 3: Developing user interfaces for originating and participating tasks](#) – 2008-04-30
- DeveloperWorks – [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 4: Developing a task list](#)

[application](#) – 2008-05-28

- DeveloperWorks – [Implementing a human-centric business process application using WebSphere Portlet Factory: Part 5: Deploying the user interfaces](#) – 2008-06-18
- DeveloperWorks – [Setting up and testing a secured, two-cell process portal environment](#) – 2007-12-12
- DeveloperWorks - [Building a human task-centric business process with WebSphere Process Server, Part 4: Portlet development](#) – 2007-07-30
- DeveloperWorks - [Building a human task-centric business process with WebSphere Process Server, Part 3: Adding portlets](#) - 2007-06-06

## ***Flex and Business Space***

Flex is UI technology from Adobe that allows a developer to create fantastic user interfaces very easily. Flex is the combination of:

- The free Adobe Flash player
- An open source set of visual components
- An open source scripting language called Action Script that is based on ECMA Script (JavaScript)
- (Optional) The Adobe Flex Builder Integrated Development Environment

### ***Widgets that use Flex***

Here is a list of widgets that are known to use Flex. There may very well be others. This list is useful if reverse engineering is needed to see how a technique was done.

- Business Monitoring > KPI History and Prediction
- Business Monitoring Tools > Alert Manager

## ***Flex Custom Widgets in Business Space***

Business Space is the BPM framework for hosting “Widgets” that allow end users to interact with IBM’s BPM products.

Business Space provides the ability to create new customer written widgets that can add or replace functions for end user interaction. These new widgets are generally called ‘custom widgets’. The creation of a general custom Business Space widget is described in detail at Custom Widgets.

The goal of this section is to provide additional knowledge on creating end user interfaces using Flex and that will be hosted as Custom Widgets in a Business Space environment.

Using Flex components, a user interface can very quickly be created. The next question is how can this be visualized in the IBM Business Space?

The answer to this is to understand that a Business Space Custom Widget is a combination of the following technologies:

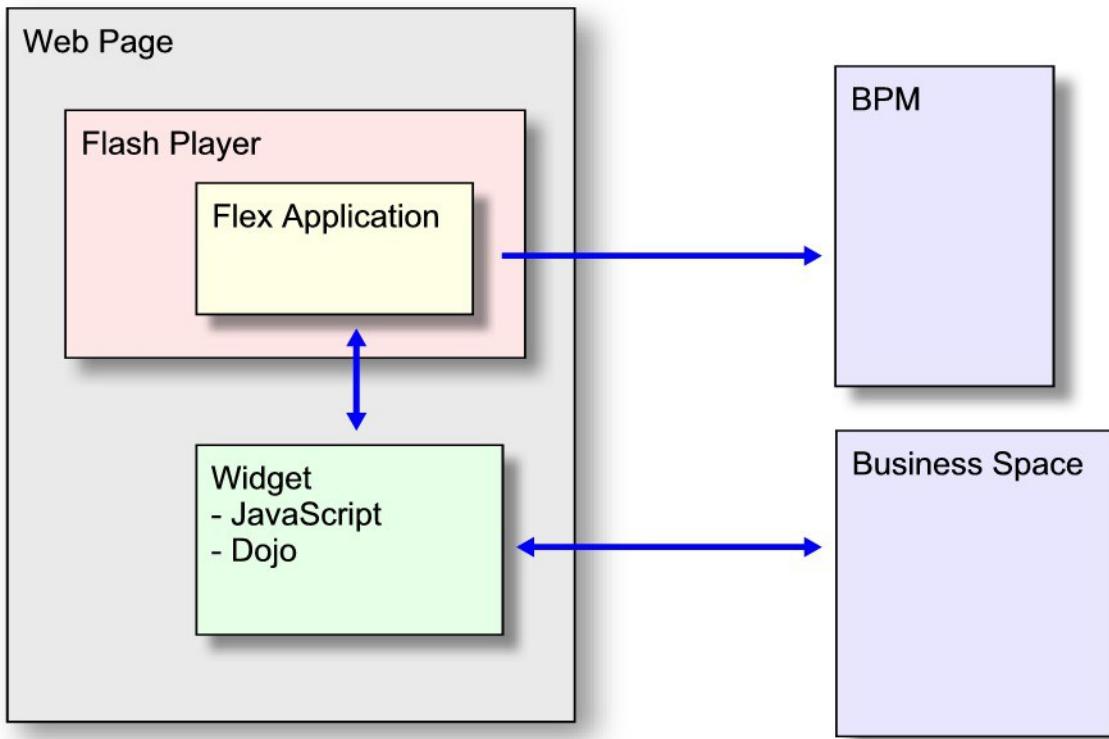
- JavaScript
- HTML
- Dojo/Dijit
- iWidget specification
- Business Space framework contracts

To create a custom widget, the developer must be aware of all of these technologies and more. This is a steep learning curve.

Fortunately, there is a relatively easy solution.

What if we could create a Business Space custom widget that could *generically* host a Flex application? If designed and implemented properly, the task of creating a new custom widget utilizing Flex would solely be that of building a Flex application solution and the custom widget could simply be *told* which Flex application to load. This would dramatically reduce the burden of implementation and allow the UI designers to focus exclusively in their area of comfort and expertise ... that being Flex.

At a high level, this solution would look as follows:



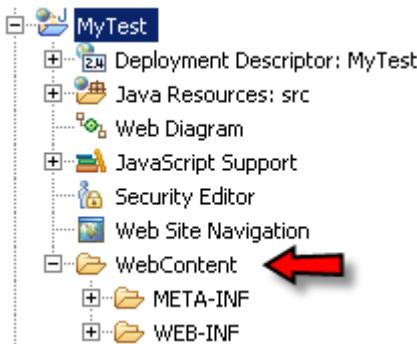
A Business Space user will open the web page to the Business Space environment and either add a new widget or see existing widgets on their page. These widgets are written in the combination of JavaScript and Dojo. The widget will insert an Adobe Flash Player object into the page using HTML and the flash player will then load the Flex application. The Flex application can communicate directly with the BPM environment using REST or other communication calls as well as interact with the Widget that loaded it.

Again, it needs to be stressed, that the designer of the user interface need **only** concentrate on their logical function and not be aware at all of the Widget and Business Space interactions. All they see is the clean world of Flex programming.

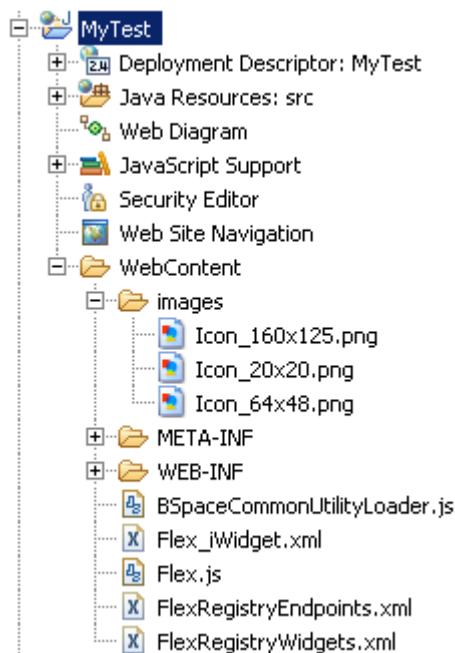
We will call such a generic Flex wrapper for Business Space by the name BSFlex.

### ***Building a new Custom Widget for Flex***

The construction of a new Custom Widget needs WID for its design and configuration. Create a new Dynamic Web project. In that Dynamic Web Project, add the following supplied files into the WebContent folder:



After adding the files to the project, the file structure should look as follows:



The files that were added were:

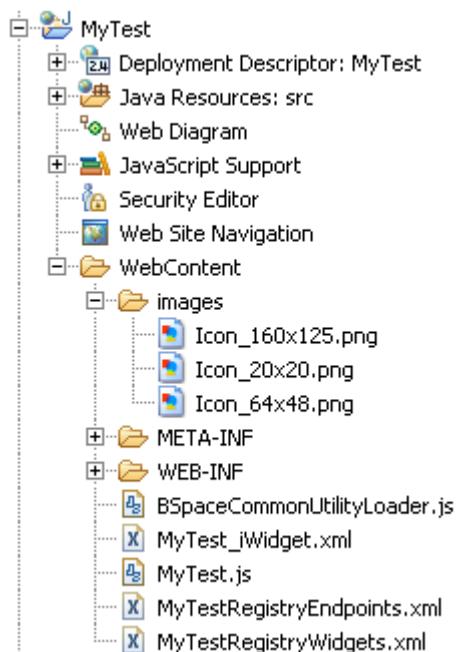
- images/Icon\_160x125.png
- images/Icon\_20x20.png
- images/Icon\_64x48.png
- BSpaceCommonUtilityLoader.js
- Flex.js
- Flex\_iWidget.xml

Now we need to rename these files. Let us assume that our new widget is going to be called "MyTest"

The mapping of file names becomes:

images/Icon_160x125.png	unchanged
images/Icon_20x20.png	unchanged
images/Icon_64x48.png	unchanged
BSpaceCommonUtilityLoader.js	unchanged
Flex.js	MyTest.js
Flex_iWidget.xml	MyTest_iWidget.xml

The resulting contents of the project after renaming looks like:



Next we must edit the content of some of these files.

The files that must be edited are:

MyTest.js

MyTest\_iWidget.xml

Within these files, the places to be changed are marked with

**\*\* CHANGE \*\***

The change is to replace the word “flex” or “Flex” with “myTest” or “MyTest”.

Copy the `xxxRegistryEndpoints.xml` and `xxxRegistryWidgets.xml` file to the WPS `<Profile>/BusinessSpace/registryData` folder/directory.

Stop and restart the WAS application called BusinessSpaceManager.

From the Flex perspective, an Adobe Flex Builder project has been supplied which contains the necessary functions for the framework to execute. Rename the `FlexWidget.mxml` file to be the same name as the target widget.

## **Environment of the Flex application**

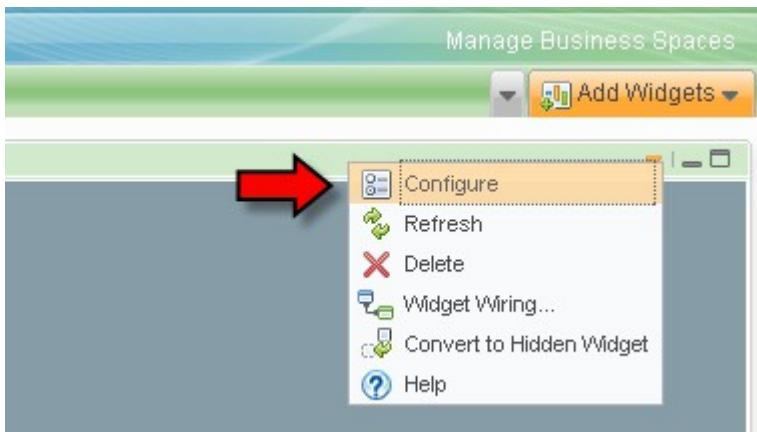
When the Flex application has been loaded, it is passed some properties from the Business Space widget. These properties can be used by the Flex programmer to communicate with BPM and Business Space.

The properties passed in can be found in the `Application.application.parameters` object

<b>Property</b>	<b>Type</b>	<b>Description</b>
iWidgetId	String	The ID of the Dijit widget in the Business Space
widgetBaseUri	String	The URL of the Web application associated with the Business Space widget
baseUri	String	Protocol/host/port to the server hosting business space

## Widget Instance Configuration

Every Business Space widget has configuration data available to it. The BSFlex wrapper needs to be informed when the user has selected configuration in Business Space.



In our Flex application, we must register a callback from Business Space which is informed when a configure request has been issued. The following code should be added that is invoked early:

```
ExternalInterface.addCallback("doConfigure", doConfigure);
```

The function registered is called (with no parameters) when configuration is requested because the user has selected Configure from the menu for the widget. The name of the callback must be doConfigure. When the doConfigure function in Flex is called, it should visualize the configuration to allow the user to change properties. This is the next section.

Our BSFlex wrapper provides an object called <local:Configuration>. This visual component loads and saves data from the Business Space persistent store.

Custom configuration for the BS Widget is achieved through modification of this widget to display the configuration desired.

The exposed (public) information for this object is:

```
function loadAttributes(): void
```

Load the attributes for this instance of the widget.

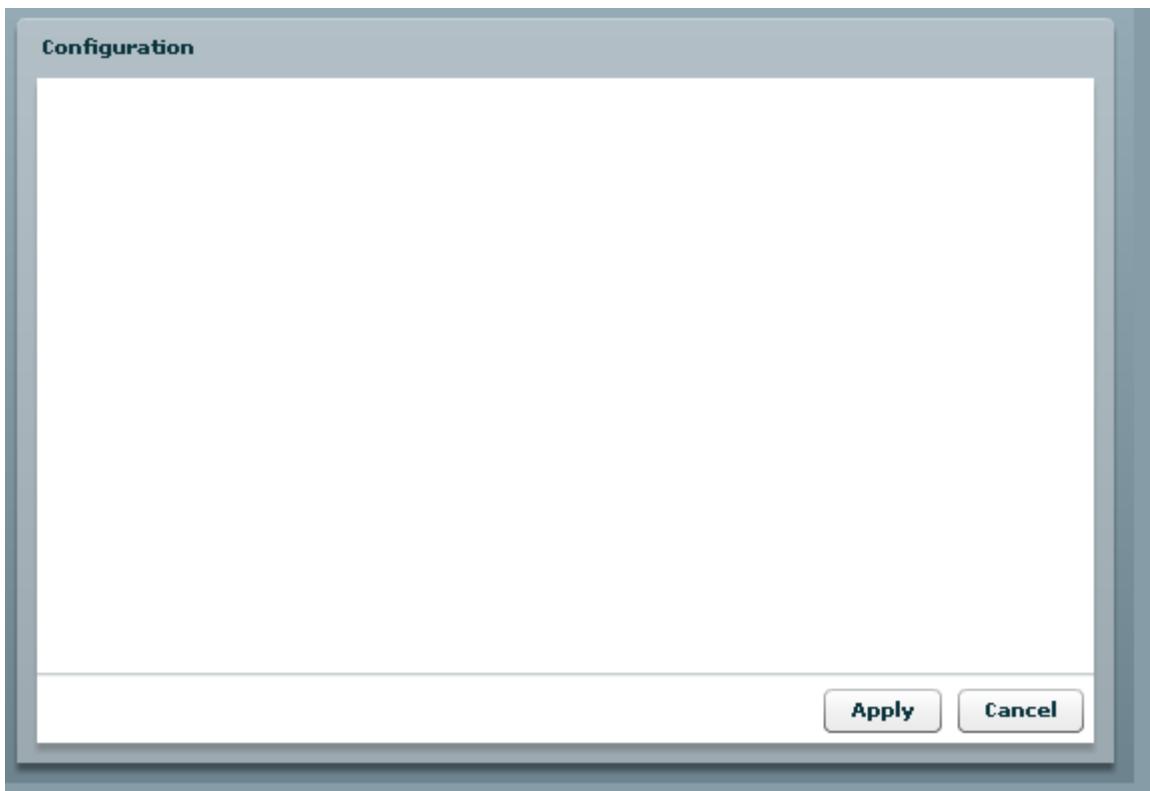
### **var attributes:Object**

This variable holds the attributes of the widget. This object can be used globally to get the values saved for this widget during configuration. The object should **never** be written to outside of the context of a configuration edit.

### **event: endConfigure**

Event issued when the configuration has completed (Apply or Cancel pressed).

The basic visual looks as follows:



When the **Apply** button is pressed, the values of the attributes object are saved back to Business Space. If the **Cancel** button is pressed, nothing is saved back to the server.

When either **Apply** or **Cancel** are pressed, an event is generated called `endConfigure` that flags the configuration as over.

It is anticipated that this component will be included in a `ViewStack` container and shown when configuration of the Widget is requested. When the `endConfigure` event is received, the widget will be hidden from the stack.

### **Example:**

Imagine that our widget has three attributes called "a", "b" and "c". The `Configuration.mxml` Flex source file would be modified to present visualization of these values. The data for the values **must** be held off the "attributes" variable as this is the data that is loaded and saved when the widget is used. The "attributes" variable is public from this flex file and hence can be read and written to globally. From a Flex programmer's perspective, the only customization needed is to code the function called `setAttributes()` which is also in `Configuration.mxml`. This function should populate the attributes object from the current settings of the configuration visuals.

### **Making REST requests**

In many cases, the Flex BS Widget will want to make REST requests back to the server to send and receive information. This is easily achieved in Flex through the `HTTPService` object.

Here is an example:

```
import mx.rpc.http.HTTPService;
import mx.rpc.AsyncToken;
import mx.rpc.events.ResultEvent;

var h1:HTTPService = new HTTPService();
var baseUri:String = Application.application.parameters["baseUri"];
```

```

h1.url = baseUri + "/rest/bpm/monitor/models";
var asyncToken:AsyncToken = h1.send();
asyncToken.addResponder(new mx.rpc.Responder(
 function(result:ResultEvent): void
 {
 var respData:Object = JSON.decode(String(result.result));
 // Do something ...
 return;
 },
 function(fault:FaultEvent): void
 {
 trace("Fault caught: " + ObjectUtil.toString(fault));
 }));

```

Although this may look a little scary at first, it can be quickly understood. First we create an instance of the `HTTPService` object which knows how to make a REST request. Next we get the base URI that will be the server target of the request. We then append to this the target specific data and set this to be the `url` property on the `HTTPService` object. We then send the request which returns an `asyncToken` object. To this we add the callback function that will be invoked when the response is available.

When using REST, we often find that we want to subvert browser security, thankfully that is handled by the Ajax Network Proxy supplied with Business Space. This can be reached at:

`/mum/proxy`

For example:

`http://localhost:9080/mum/proxy/http/www.google.com:123`

Prior to 7.5.1, the Proxy was wide open meaning that any redirections were possible. From 7.5.1 onwards, Proxy configuration was tightened. The proxy configuration file is called "proxy-config.xml" and can be found at:

`<Profile Name>/BusinessSpace/<nodeName>/<serverName>/mm.runtime.prof/config`

After making these changes, the `AdminTask.updateBlobConfig wsadmin` command must be executed. Details of this can be found in the BPM InfoCenter. An example would be:

```

AdminTask.updateBlobConfig('[-serverName server1 -nodeName win7-x64Node01
-propertyFileName
"C:\IBM\WebSphere\AppServer\profiles\ProcCtr01\BusinessSpace\win7-
x64Node01\server1\mm.runtime.prof\config\proxy-config.xml" -prefix "Mashups_"]')

```

`AdminConfig.save()`

To switch off security for the Proxy, an entry such as the following may be added to `proxy-config.xml`:

```

<proxy:policy url="*" acf="none" basic-auth-support="true">
 <proxy:actions>
 <proxy:method>GET</proxy:method>
 <proxy:method>POST</proxy:method>
 <proxy:method>PUT</proxy:method>
 <proxy:method>DELETE</proxy:method>
 </proxy:actions>
 <proxy:headers>
 <proxy:header>Cache-Control</proxy:header>
 <proxy:header>Pragma</proxy:header>
 <proxy:header>User-Agent</proxy:header>
 <proxy:header>Accept*</proxy:header>
 </proxy:headers>
</proxy:policy>

```

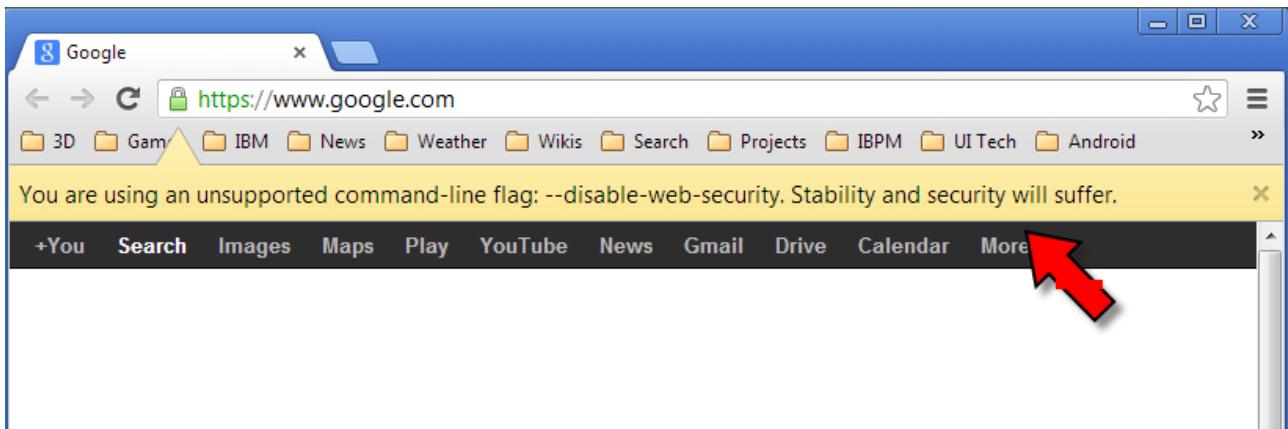
```

<proxy:header>Content*</proxy:header>
<proxy:header>X-Method-Override</proxy:header>
<proxy:header>X-HTTP-Method-Override</proxy:header>
<proxy:header>If-Match</proxy:header>
<proxy:header>If-None-Match</proxy:header>
<proxy:header>If-Modified-Since</proxy:header>
<proxy:header>If-Unmodified-Since</proxy:header>
<proxy:header>Slug</proxy:header>
<proxy:header>SOAPAction</proxy:header>
</proxy:headers>
<proxy:cookies>
 <proxy:cookie>LtpaToken</proxy:cookie>
 <proxy:cookie>LtpaToken2</proxy:cookie>
 <proxy:cookie>JSESSIONID</proxy:cookie>
</proxy:cookies>
</proxy:policy>

```

If you are using the Chrome web browser, it has a command line option called "`--disable-web-security`" that switches off web security (Same Origin Policy) for that instance of the browser. This is useful for development but should **never** be used or suggested for production.

You will know it is working because you will see a warning message:



See also:

- [Configuring the HTTP proxy for AJAX applications \(V1.0\)](#)

## Handling returned XML

Some of the REST requests return XML data. By default, this XML is parsed into an Object. If we want the pure XML text, we need to set the `resultFormat` property of the `HTTPService` object to "text". It has been found that the XML returned contains carriage-return/line-feed characters (i.e "`\r\n`"). To remove the carriage-return, the following AS3 can be used:

```

var data1:String = String(result.result);
var p1:RegExp = new RegExp("\r", "g");
data1 = data1.replace(p1, "");

```

## Cross domain security

A Flex application that wishes to call a service in a machine other than that from which the Flex application was loaded will be disallowed by default. This protects the Flex sandbox from breaking

security protocols. For development testing, create a file called crossdomain.xml and have it available for download from the machine that the REST request is targeted to. The content of the file should look as follows:

```
<?xml version="1.0"?>
<cross-domain-policy>
 <allow-access-from domain="*" />
</cross-domain-policy>
```

Be aware that this will allow ANY Flex application to connect to the server. This is usually fine for testing. Before setting up cross domain security, first test to see if the application works without it. For BPM apps it seems to work fine probably due to explicit authentication.

## Problems with PUT and DELETE requests

Unfortunately, Flex's HTTPService object disallows both PUT and DELETE requests. What this means is that a complete section of WPS REST interfaces appear to be inaccessible. Fortunately, WPS claims to accept the X-Method-Override capability.

### Bug

The problem with this is that it appears not to actually work. PMR 79493,004,000 has been raised to address the problem. As of 2009-07-02 this has been partially fixed with an APAR available from support called IZ52208. This problem no longer exists in the latest versions of WPS with the latest fixpacks applied.

An example of this would be

```
h1.method = "POST";
h1.headers = {"X-Method-Override": "PUT"};
```

## Widget to Widget Event Handling

Event handling is the idea that a Business Space widget can be a source or destination of events. An event is simply the transmission of some data from one widget to another. The data can be anything that the Flex programmer chooses to transmit.

### Sending Events

A Flex application can send an event directly to another widget. The way to achieve this is illustrated in the following code fragment:

```
var iWidgetId:String = FlexGlobals.topLevelApplication.parameters.iWidgetId;
ExternalInterface.call("_" + iWidgetId + "_iContext.iEvents.publishEvent",
 "eventName",
 "parms");
```

### Old technique

Sending an event is accomplished in two parts. First the flex application will make a call back to the JavaScript wrapper:

```
var iWidgetId:String = FlexGlobals.topLevelApplication.parameters.iWidgetId;
ExternalInterface.call("_" + iWidgetId + "_iContext.iScope().funcName", parms);
```

In the JavaScript file, we need to code something like the following to actually execute the event publish.

```
this.iContext.iEvents.publishEvent(eventName, payload);
```

This needs to be wrapped in the function that the Flex application will invoke through the ExternalInterface call.

```
eventPublish_markerClicked: function(payload)
{
 this.iContext.iEvents.publishEvent("com.kolban.map.MarkerClicked", payload);
}
```

Convention has us declare the function with the name:

```
eventPublish_<event name>
```

The iWidget.xml file for the custom widget must be modified to declare that the widget is capable of publishing events. Code similar to the following should be added:

```
<iw:eventDescription
 id="com.sample.MyEventDescription"
 payloadType="JSON"
 lang="en"
 description="My Event Description" />

<iw:event
 id="com.sample.MyEvent"
 description="com.sample.MyEventDescription"
 published="true" />
```

The name of the event to be published is found in the iw:event@id location.

Unfortunately, we can't call the fireEvent wrapper function directly. This is because the function is defined on the JavaScript object that represents the widget and not on the page itself. In WPS 7.0, when a Business Space widget is on a page, the JavaScript object that represents that widget is hooked off the root of the HTML DOM tree. It appears to have a name of

```
_iWidgetId_IContext
```

This object has a function upon it called iScope(). This returns the JavaScript object that we wish to access.

From this a Flex application can call the functions in that widget directly. From within Flex, we can then call functions on that object as shown in the following example:

```
var iWidgetId:String = FlexGlobals.topLevelApplication.parameters.iWidgetId;
ExternalInterface.call("_" + iWidgetId + "_iContext.iScope().testFunc", "Hello
World");
```

The iWidgetId can be passed into flex from the JavaScript variable

```
this.iContext.io.id
```

## **Receiving Events**

Receiving events sent from another Widget is a little trickier.

In the receiving Flex application, a function needs to be created with the signature:

```
private function <name>(payload:Object) : void
```

This is the function that is to be called when an event transmitted by a partner widget has been

received. It is in this function that the Flex code for processing the event will be written.

This function needs to be exposed to the surrounding JavaScript with a call to the Flex provided class called `ExternalInterface` such as:

```
ExternalInterface.addCallback("<name>", name);
```

This will allow the surrounding JavaScript to call the method when an event arrives. A suitable place for adding this code is in the `init()` method.

In the iWidget XML file, the fact that this widget can now subscribe to incoming events must also be registered:

```
<iw:eventDescription id="com.kolban.map.MapInputDesc"
 description="MapInput Desc1"
 lang="en"
 payloadType="JSON">
</iw:eventDescription>
<iw:event id="com.kolban.map.MapInput"
 description="com.kolban.map.MapInputDesc"
 handled="true"
 published="false"
 onEvent="eventSubscribe_mapInput"/>
```

In the JavaScript for the iWidget, a callback handler must be defined:

```
eventSubscribe_mapInput: function(event)
{
 var swfId = this.iContext.widgetId + "_" + this.widgetName;
 if (dojo.isIE) {
 window[swfId].eventSubscribe_mapInput(event.payload);
 } else {
 document[swfId].eventSubscribe_mapInput(event.payload);
 }
}
```

## Working with WPS Human Task / Process Data

WebSphere Process Server has the notion of Business Objects. Both a BPEL process and a Human Task can take one or more Business Objects as input and return one or more Business Objects as outputs. On occasion (and especially with Human Tasks) we may wish the input to be presented to the user and have a response returned to the process or task (we will focus now on just tasks, but the same applies to processes).

WPS internally represents a Business Object definition as an XML Schema and a Business Object instance as an XML document corresponding to that schema. When we make REST calls to WPS to get Human Task input data, we are returned an instance of an XML document. If we want to send new data to the Human Task as output data for that task, we would supply an XML document.

Through the REST APIs we can ask for an instance of task data. The following code fragment illustrates how this can be done:

```
h1.url = "http://localhost:9080/rest/bpm/htm/v1/task/" + tkid + "/input";
h1.method = "GET";
h1.resultFormat = HTTPService.RESULT_FORMAT_E4X;
var asyncToken:AsyncToken = h1.send();
asyncToken.addResponder(new mx.rpc.Responder(
 function(result:ResultEvent) : void
```

```

{
 var myXML:XML = XML(result.result);
 // Do something with the XML
},
function(fault:FaultEvent) : void
{
 trace("Got a fault!");
}
));

```

What we end up with is a Flex XML object. If the variable holding this data is held at the highest level and defined as bindable ... for example:

```
[Bindable]
private var myXML:XML;
```

Then the fields in the data can be accessed through Flex bindings.

Let us now work through an example of this. Imagine we have a Business Object that is defined as follows:



It is used in a Human Task Interface that looks like:

	Name	Type
▼	mailing	
Inputs	input	Address
Outputs	output	Response

The XML document retrieved via a REST call to WPS might look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<p:mailing_._type xsi:type="p:mailing_._type"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://HTTestPerson/Mailing">
 <input>
 <street>9200 Glenhaven</street>
 <city>Fort Worth</city>
 <state>TX</state>
 <zip>76182</zip>
 </input>
</p:mailing_._type>
```

It looks pretty scary ... but if we take it apart we have the following:

```
<root>
 <input>
 <street>
```

```
<city>
<state>
<zip>
```

What we find is that the immediate children of the document are the input parameters. The tags are named after the names of the parameters. In this case the tag called `input` matches the parameter called `input` found on the Interface.

The children of the parameter tags are the data. In this case, a data type called `Address` which matches the BO definition.

When this is assigned to a Flex XML object, we can then address the fields contained within such as:

```
[Bindable] var myData:XML = ...

var street:String = myData.input.street;
var state:String = myData.input.state;
```

If we then use Flex Binding, we can display a field in a `TextInput` Flex component with the following:

```
<mx:TextInput text="{myData.input.street}" />
```

## Mapping Data types and controls

A WPS Business Object field has a finite collection of possible common scalar data types associated with it. These include:

- string
- int
- date

When we look at the input and output data associated with a task, we must realize that the Business Object is serialized to and from an XML document. This then asks the question, what is valid for a given BO field type in the associated XML item?

### ***String***

Any character string is allowable.

### ***Int***

Any positive or negative numeric characters are allowable

### ***Date***

Date is more interesting. The format of the date expected in XML is “YYYY-MM-DD”. This is not the default format of a Flex `DateField`. `DateField` has a property called `formatString` which can be set to “YYYY-MM-DD” to generate the correct format for the XML document.

## **Working with repeating/array data**

## **Working with AnyType**

### **Performing input validation**

#### **Single User Workflow**

Single User Workflow is the idea of being presented with a page, completing that page and then immediately being presented with a succeeding page without having to manually claim the task.

InitiateAndClaimFirst

In: processTemplateName

In: InputMessage

Out: AIID

Out: ActivityName

Out: InputMessage

CompleteAndClaimSuccessor

In: AIID

In: OutputMessage

Out: AIID

Out: ActivityName

Out: InputMessage

CompleteAndClaimSuccessor

...

CompleteAndClaimSuccessor

<End>

## **Flex related REST Problems**

Validate that the format of data being sent or received matches the expected format for the HTTPService request.

Use the FireFox 3.5 FireBug tool which can show Flash related communications.

Only a large subset of the BFM and HTM APIs are available via the REST interface. If the other BFM and HTM calls are needed, we seem to be stuck. As a possible workaround, we can create our

own REST based servlet that when called, makes the correct local EJB calls. This is not an easy/obvious proposition.

## Elixir

### **References – Elixir**

- [IBM Home page for Elixir](#)
- DeveloperWorks - [IBM ILOG JViews Enterprise Overview](#) – 2010-03-02
- DeveloperWorks - [Create an Elixir application using Adobe Flex Builder](#) - 2009-11-04

### **References – Flex Widgets**

- DeveloperWorks – [Develop a rich Web client for a business process application with WebSphere Integration Developer, WebSphere Process Server, and Adobe Flex](#) – 2010-03-24
- DeveloperWorks - [Integrating Flex applications with IBM Mashup Center](#) - 2009-06-29
- DeveloperWorks – [Creating a Flex-based widget to display WebSphere Business Monitor data in Business Space](#) – 2009-04-29
- [Flex Video Training](#)
- [Tour-De-Flex](#) – An online set of examples of all the different flex capabilities

## Custom Process Portals

The IBM BPM framework provides what it calls the "Process Portal". This is an IBM supplied web based application that provides users with the abilities to:

- See the list of tasks waiting for them to work against
- Start new instances of processes
- Interact with the screens of individual tasks

It looks as follows:

The screenshot shows the 'My Tasks' section of the IBM Process Portal. At the top, there are tabs for 'My Tasks' and 'Saved Searches'. Below this is a search bar and a filter dropdown set to 'Open Tasks | Completed Tasks'. A section titled 'Overdue (25+)' is expanded, showing four task items. Each item has an exclamation icon, a title like 'Step: Untitled' or 'Step: Step A', a due date ('December 12, 2012 1:24 AM' or '8:44 PM'), and a small note about the BPD version (e.g., 'BPD2:175'). A scroll bar indicates there are more results. At the bottom left is a 'Show More' button. On the right side of the screen, there is a sidebar with buttons for 'Launch', 'Following', and '@Mentions', and a list of applications: BPD2, ClaimNext, and JavaScript Search Code Builder.

The IBM supplied Process Portal is feature rich and extremely powerful. However there is a commonly encountered problem with consumers actually using it. Simply put, it may not be the company's strategic direction for interacting with users. Imagine a clerk responsible for processing insurance claims. This clerk has been trained to use the existing user interfaces of the existing applications. It would not be surprising if multiple distinct applications could be worked upon through this one user interface. If IBM BPM is to be considered as a component in a solution, it may not be desired to use Process Portal for a variety of non-functional business reasons. From a raw technical perspective, it will work just fine ... but if Process Portal were used, we have the unfortunate story of the clerk user having to switch from one application to another to perform their work. In addition, extra training will have to be supplied to the clerks to teach them how to use the new user interface. All of this adds up to expense and introduces opportunities for errors. There is low quality in saying "*Enter new claim data in this application*" but "*View claim history in that application*".

Fortunately, there is an elegant solution. The IBM BPM product provides programming interfaces (APIs) that can be used by technical programming staff to perform the tasks that are achieved through IBM's supplied Process Portal. These APIs are exposed through the REST style of remote function call. By having such an API available, consumers can build or augment their own existing user interface applications to include access to the features provided by IBM BPM. This section will continue to discuss and dive deeper into that notion.

First of all let us ask ourselves what are the key concepts that we must take into account:

- Task lists – The core notion of IBM BPM is that there are tasks which are to be performed by end users. The process models define when and to whom these tasks are to be assigned. When a user is "free" to perform a task, they are expected to pull-up a task list and pick (or be given) the next task to work upon. This means that there has to be a capability to search for and retrieve a list of tasks that are associated with a user.
- Data presentation and input – For each task, there is data provided by IBM BPM that is to be made known to the user as well as data or decisions that are to be entered by the user. With Process Portal these screens are described through the "coach" technology and shown to the end users. With custom user interfaces we have more UI choices. We can either launch a coach based screen or else allow the consumer to build their own screen's from scratch.
- Starting a new Process – A user may wish to initiate a new process. Perhaps a telephone clerk has received a call from a customer for a new order and now wishes to start an instance of the order handling process.
- User authentication – For a user to be able to interact with IBM BPM, the run-time must know who the user is and be convinced that they are in fact who they claim to be. This means that we must provide a "sign-on" mechanism which includes propagation of the user's identity/credentials with each REST request. A solution for this should be integrated with any existing sign-on provided by existing user interface applications.

See also:

- REST Integration

## Types of Custom Portal

Let us start with some simple thoughts on different types of portals. At the highest level we have two basic kinds. Those known as "thick clients" and those that are "browser based". Thick clients are native executables installed locally on consumers desktops. These are applications that may use Windows/.NET APIs, Java Swing or Java SWT (as examples). The second type of process portal are those that run within the browser environment. These are the more common types of custom portals. These can also be broken down into categories based on whether the web page (HTML) is built in the browser (Web 2.0 style) or built on the server.

Browser built HTML technologies includes:

- Dojo
- jQuery
- YUI
- Flex

Server built HTML includes:

- JSP
- JSF
- Portlets (JSR168, JSR286) hosted by portal providers
  - IBM WebSphere Portal
  - Liferay

Irrespective of the type of UI technology used, the high level concepts and design remain the same.

1. Use REST API to obtain a list of tasks
2. Use the UI technology of choice to show the list to the user
3. Allow the user to select the single task on which they are to work

Once the task has been selected, we now choose how to display the task data itself. This will either be the Coach UI as an in-line web page perhaps using iFrame technology or else it will be custom controls/UI using the REST API to get the data for a specific task.

See the following for the appropriate REST APIs for these tasks:

- Working with REST Search Queries
- Finishing a Task
- Getting Task variable values

## Generic Widgets for Custom Portals

With the idea of Custom Portals in mind, maybe we can design a framework for generic portals? This framework would be as flexible as possible and hide the majority of details from the consumer. To make the solution as flexible as possible, let us assume that we introduce the notion of a "Widget". A Widget is the user interface building block that is responsible for some function.

### The Task List Widget

Here is a picture of the IBM Process Portal task list:

The screenshot shows a 'My Tasks' page with a search bar and navigation links for 'Open Tasks' and 'Completed Tasks'. Below this, a section titled 'At Risk (3)' displays three task items. Each item has a yellow warning icon, the title 'Step: Step A', a dropdown menu, and a due date of 'December 4, 2012 6:11 PM'. The tasks are associated with 'All Users' and have IDs 'BPD2:115', 'BPD2:116', and 'BPD2:117' respectively. At the bottom, a message says 'Showing 3 of approximately 3 results'.

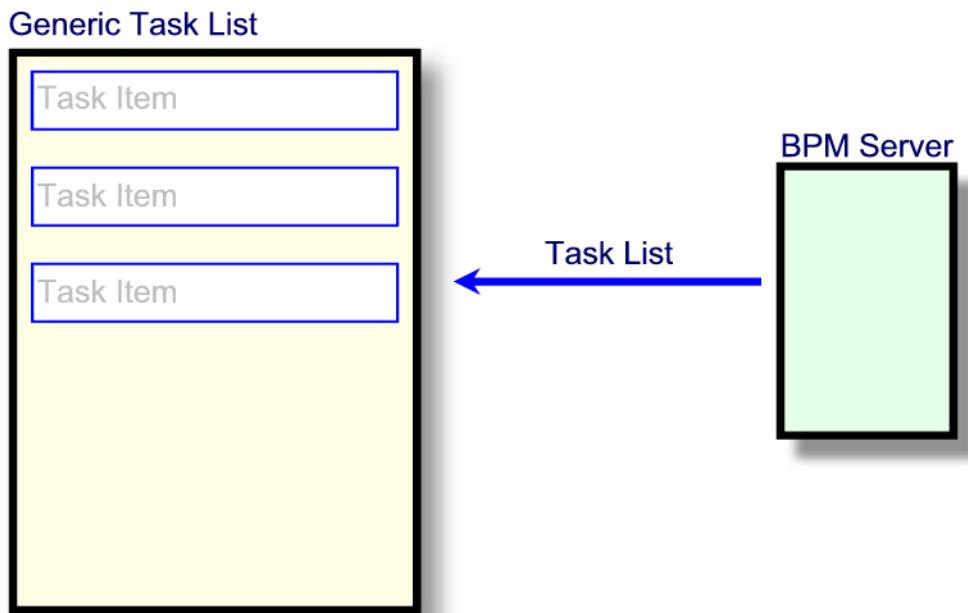
As we see, it is very attractive but what does it "logically" contain? If we strip away the superficial colors and styling, what is left?

The answer is that it is a visual container that lists a series of tasks that a user can work upon. This is a core thought. The concept of the Task List is a "container of tasks". It is responsible for determining a list of tasks and then showing each task. If a building block Widget of a Task List were to be created, what would it consist of?

Ignoring its visual appearance, we could define it to have:

- `refreshTasks()` – A function which, when called, refreshes the list of tasks shown to the user. This could be called when the page loads to populate the list of tasks. Alternatively, a "refresh" button could be supplied which, when clicked, would call this function to refresh the list.
- `taskSelected(taskId)` – A task selected by the user. This would be a callback function. When the user selects a task, this function is called. It is expected that this function is implemented by the custom portal designer. It could then be "wired" to another widget to show the selected task.

With this minimal interface, the Task List widget would then be functional.



After a Task List requests tasks from a BPM Server, the Task List must now display each Task retrieved.

We will now assume the existence of a second type of Widget that we will call the TaskItem. The Task List will create an instance of a TaskItem for each Task that it wishes to display. Since the TaskItem is a widget, it is responsible for displaying its content. Again ignoring the visual appearance of the TaskItem, we could define it as having the following signature:

- `taskDetails` – A task item returned from the BPM Server describing the task. This is a property.
- `onSelected(taskId)` – A callback function called when the task is selected by the user

By generalizing these two functions, we now have all we need to create arbitrary visualizations of a Task List.

Here is an example visualization of a new Task List using these principles:

The screenshot shows a "Task List Title" widget with a refresh button at the top. Below it is a list of four tasks, each represented by a card:

- Step: Step A** Due: 2012-12-05T00:11:39Z All Users BPD2:115
- Step: Step A** Due: 2012-12-05T00:11:40Z All Users BPD2:116
- Step: Step A** Due: 2012-12-05T00:11:42Z All Users BPD2:117
- Step: Step A** Due: 2012-12-05T05:32:08Z All Users BPD2:118

### The Task Table Widget

When we think of a custom portal, we must consider the selection of tasks by a user. The user wishes to see these tasks in a desirable form. Given that the user may have some number of tasks available for them to work upon, one style of task list presentation is that known as the table. The table is pretty much what it sounds like. It is a rectangular array of information composed of rows and columns. Each row will represent a single task and each column will hold some attribute of that task.

The Task Table widget is an alternative to the Task List widget. It presents a table of tasks to the user and allows the user to select a task through the "Open" button.

Task Name	Status	TaskId	Actions
Step: Step B	Received	133	<button>Open</button>
Step: Step B	Received	134	<button>Open</button>
Step: Step A	Received	135	<button>Open</button>
Step: Untitled	Received	153	<button>Open</button>

This widget has a callback function called "onTaskSelected" which is invoked and passed an object which contains the taskId of the task that was selected.

The widget exposes a method called "refreshTasks ()" which, when invoked, will retrieve the current set of tasks and refresh the table with the new information.

An example of usage would be:

```
<div data-dojo-type="kolban.widget.TaskTable">
 <script type="dojo/method">
 this.refreshTasks();
 </script>
 <script type="dojo/connect" data-dojo-event="onTaskSelected"
 data-dojo-args="value">
 console.log("Connect called: " + value.taskId);
 myCoachViewer.set("taskId", value.taskId);
 </script>
</div>
```

This fragment will create the TaskTable widget, call its `refreshTasks()` method and, when a task is selected, call a CoachViewer to show the Coach associated with the task.

### **The CoachViewer widget**

The CoachViewer widget is used to load and view the Coaches for an associated task. It allocates an area of the browser display and when asked to show a Coach, that area then hosts the Coach details. It has a property called "taskId" that **must** be set using the widget's "set()" method. When a new taskId is supplied, the widget will display the content of the associated Coach.

An example of usage would be:

```
<div data-dojo-type="kolban.widget.CoachViewer" data-dojo-id="myCoachViewer">
</div>
```

This can be combined with the TaskTable widget (see previous) to select which task to show.

The CoachViewer also has a callback function called "onCompleted" which is called when the user completes the task. This can be used to signal a refresh of the task list or some other function. The parameter passed to the callback is an object which contains:

- status – The status of the completion. This may be
  - "completed" – to indicate that the coach completed successfully
  - "error" – to indicate that an error was encountered. If the status is error, two additional properties will be available:
    - errorMessage – A message describing the error
    - errorNumber – The number of the error

A property of the CoachView called "showStandby" can be set to "true" to show a "standby" overlay while the Coach is loading.

A property called "showError" is defined which is a boolean. If set to true and an error is encountered, a dialog box showing the nature of the error will be displayed. If set to false, no such dialog will be shown. In either setting, an error will be returned via the `onCompleted` callback.

- taskId – The taskId of the task who's coach is to be shown in the CoachViewer. Must be set by the CoachViewer `set()` method.
- showStandby – A boolean value which, if true, will show a standby message while the coach is loading. The default is false.
- showError – A boolean value which, if true, will show an error dialog if an error is encountered. The default is true.
- onCompleted – A callback function executed when the coach completes

### **Determining when the current Coach has completed**

When writing a custom Portal, the current Coach is usually shown in an iFrame. When the Coach completes, we somehow need to tell our portal framework that this has happened. It seems that when a Coach completes, it posts a message to the DOM "window" containing the iFrame. This is

the same window being used by the portal. The following code will register a callback for such an entry:

```
var onCompleted = function (m) {
 debugger;
}
if (typeof window.addEventListener != 'undefined') {
 window.addEventListener('message', onCompleted, false);
} else if (typeof window.attachEvent != 'undefined') {
 window.attachEvent('onmessage', onCompleted);
}
```

A more modern solution using Dojo would be:

```
on(window, "message", function(m) { ... });
```

Unfortunately, the callback is invoked on a number of occasions by different parties, not just when we expect, so we have to be careful to check the response.

The passed in "m" message contains a field called "data" which appears to contain a single string which is a JSON encoded piece of data that contains:

- name – so far we have seen instances of:
  - onInitialized – Called when the Coach has been loaded
  - onCompleted – Called when the Coach has been unloaded
  - onError – Called when there was an error showing the coach
  - notifyBoundaryEvent – Return from a boundary event?
- taskID
- applicationInstanceId
- parameters – Can be array of strings. For onError it appears to be a message and a message code.

## ***Using the Custom Portal widgets***

Now that we have described the functions of the custom portal widgets, what remains is to describe how to use them.

A Widget is added to a Web Page (a source HTML file). Since the widgets are Dojo, the first thing we have to do is to add the mandatory boiler plate to load the Dojo environment:

```
<script
 type="text/javascript"
 src="/teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js"
 data-dojo-config="async: true, parseOnLoad: true">
</script>
<link
 rel="stylesheet"
 href="/teamworks/script/coachNG/dojo/1.8.3/dojo/resources/dojo.css" />
<link
 rel="stylesheet"
 href="/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/claro.css" />
```

The above must be added to in the <head> section of the page. The URL to Dojo is good as of IBM BPM 8.5.

Because we are using custom widgets that are not part of the default Dojo distribution, we need to tell the environment where to load those widgets from. This is achieved using the AMD technology. In addition to saying *where* the custom widgets are located, we also have to say *which*

of those widgets are going to be used in the page. Again in the <head> section, we add the following:

```
<script type="text/javascript">
 require({
 packages : [{
 name : 'kolban',
 location : '/StaticTests/kolban'
 }]
 });
 require(["kolban/widget/ProcessStart"]);
</script>
```

Because we are using Dojo, we must also specify the Dojo theme that we are going to use. The HTML <body> tag should have the Dojo theme added to it:

```
<body class="claro">
```

We are now ready to include any widgets we wish, for example:

```
<div
 data-dojo-type="kolban.widget.ProcessStart"
 data-dojo-id="myProcessStart" >
 <script
 type="dojo/connect"
 data-dojo-event="onStarted"
 data-dojo-args="processDetails">
 console.log("Process Started!");
 console.dir(processDetails);
 </script>
</div>
```

Putting it all together, a sample would look like:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Test_ProcessStart</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<script
 type="text/javascript"
 src="/proxy/teamworks/script/coachNG/dojo/1.8.3/dojo/dojo.js"
 data-dojo-config="async: true, parseOnLoad: true">
</script>
<link
 rel="stylesheet"
 href="/proxy/teamworks/script/coachNG/dojo/1.8.3/dojo/resources/dojo.css" />
<link
 rel="stylesheet"
 href="/proxy/teamworks/script/coachNG/dojo/1.8.3/dijit/themes/claro/claro.css" />
<script type="text/javascript">
 require({
 packages : [{
 name : 'kolban',
 location : '/StaticTests/kolban'
 }]
 });
 require(["kolban/widget/ProcessStart"]);
</script>
</head>
<body class="claro">
<h1>Test ProcessStart</h1>
<p>The ProcessStart Widget presents a visible list of start-able processes. Clicking on one of these entries causes a new instance of that process to start.</p>
<hr />
<div
 data-dojo-type="kolban.widget.ProcessStart"
 data-dojo-id="myProcessStart">
 <script
 type="dojo/connect"
 data-dojo-event="onStarted"
 data-dojo-args="processDetails">
 console.log("Process Started!");
 console.dir(processDetails);
 </script>
</div>
```

```
</div>
<hr />
</body>
</html>
```

## Writing custom portals using Coaches

In principle, a custom portal can itself be written as a Coach.

Within a custom portal we can imagine the user being presented with a list of tasks upon which they can work. After selecting such a task, a new window might open which will show the details of the new task. The URL to be used to open the window can be found using the REST API to query upon the client settings of the taskId to be launched. This REST API returns a URL for exactly this purpose. However, there appears to be a problem (as of 8.0.1.1). When the newly opened Coach completes, it seems to cause a "refresh" or "reset" of the Coach that launched it. This means that if we write a Coach which acts as a process portal, select a task and then open that task in a new window, when that new window ends, the original process portal Coach is reset. We do not yet know of a circumvention.

Addition: 2013-10-15: It appears that if we open a new window for a Coach if we set that window's "opener" property to null, the refresh will not happen.

## Apache Flex

Flex is a function rich UI development environment that runs within a browser, native on the desktop and within Mobile platforms including Android and iOS. Originally developed by Adobe, it was Open Sourced and provided to the Apache group. The latest information on Flex can be found here:

<http://flex.apache.org/>

There are a variety of fee and free development tools for building Flex applications. These include:

- [Adobe Flash Builder](#) – fee
- [FDT](#) – fee
- [Tofino](#) – free
- [FlashDevelop](#) - free

## Flex for Mobile

One of the key reasons for considering Flex is for building mobile applications (i.e. those that can run on Android or iOS). Flex provides a write-once/run-anywhere model. It should be stressed that a logical "runtime" of the Adobe AIR environment is used but to all intents and purposes building such apps may be considered closer to native Apps than attempting to build HTML5 based applications.

The Flex support for Mobile appears to be first class but restricts the components that can be used. Here is a quick summary of those supported:

StageWebView	Web Browser
BusyIndicator	
Button	
ButtonBar	

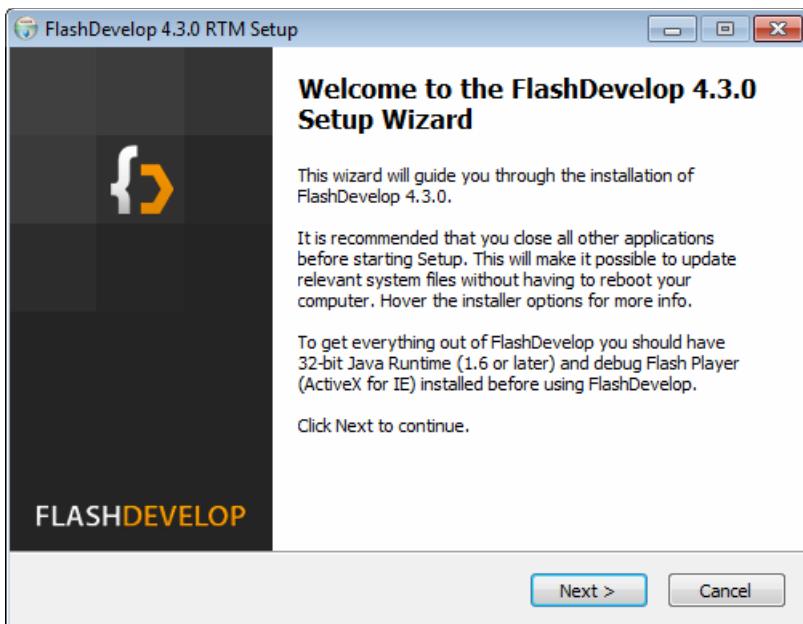
Callout	
CalloutButton	
CheckBox	
DateSpinner	
HSlider	
Image	
Label	
List	
RadioButton/RadioButtonGroup	
SpinnerList	
TextArea	
TextInput	
ToggleSwitch	

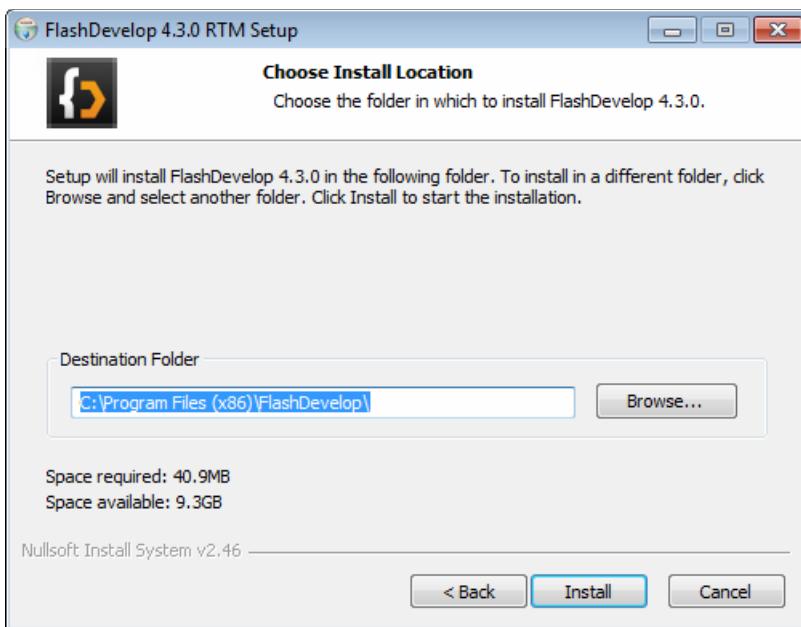
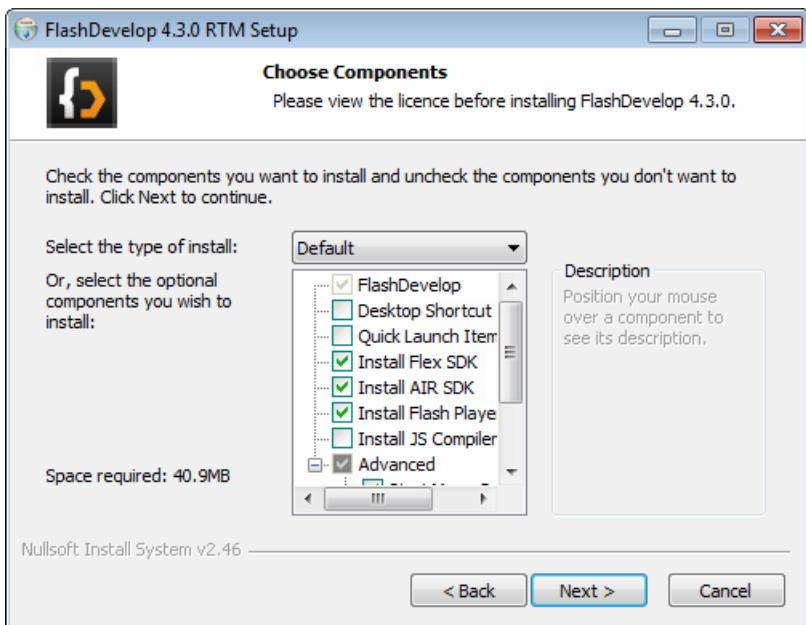
DataGroup	
Group	
HGroup	
Scroller	
Spacer	
TileGroup	
VGroup	

## Installing FlashDevelop

FlashDevelop is a free, Open Source development environment for Flex based applications:

<http://www.flashdevelop.org/>





## **Google Charts**

Google provides a charting package. The URL for this is:

<https://developers.google.com/chart/>

To use this technology, we need to perform some work:

1. Load the Google JavaScript APIs

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

2. Load the packages

```
google.load('visualization', '1.0', {'packages': ['corechart']});
```

3. Provide a callback function to execute when the packages are loaded

```
google.setOnLoadCallback(function() {
 ... code goes here ...
});
```

## **Providing data to Google Charts**

All Google charts expect data encapsulated in a DataTable object. The name of this object is "google.visualization.DataTable". There are a number of ways to build and populate this object. We will touch upon some of them. For full details, see the full Google documentation.

```
var data = new google.visualization.DataTable();
data.addColumn('string', 'Name');
data.addColumn('number', 'age');
data.addRows(1);
data.setCell(0, 0, 'Bob');
data.setCell(0, 1, 34);
```

## **Google Chart – Gauge**

The Gauge control provides a "dial" or "gauge" which contains a label and a value. The value is shown as well as a needle that provides a relative indication of where the value falls:



Package: `google.load('visualization', '1', {packages: ['gauge']});`

```
var chart = new google.visualization.Gauge(node);
chart.draw(data, options);
```

## **Google Web Toolkit – GWT**

Google Web Toolkit (GWT) is a modern Web 2.0 toolkit/framework and development environment. It has the ambitious (yet apparently achieved) goal of allowing the page designer to write their code in the Java programming language which is then cross-compiled to JavaScript for execution within the browsers. In addition, a rich set of visual widgets are available as well as a free UI Integrated Development Environment (IDE) based on Eclipse. This is arguably the future of web based programming and will give Dojo a serious run for its money especially since it has a

superb and free UI designer built into Eclipse.

At the time of writing (2013-06), the latest version of GWT is 2.5.0.

A question which comes up from time to time when reading this section of the book is why was so much dedicated to Google's GWT in a book about IBM's BPM? The answer to this is simple but may not be what one wants to hear. The author took the route that he wanted to keep all his thoughts in one document with cross references. Multiple books means multiple administrative challenges when working with content and the simplest solution is to have one book for all activity. The context in which the author studied GWT was that of BPM and much is written about GWT and its potential relationships with BPM.

See Also:

- [GWT Home page](#)

## Making REST calls from GWT

The ability to make REST calls from GWT is core to the BPM story. The REST application will call BPM to obtain useful information. To begin, a GWT application must inherit the HTTP package:

```
<inherits name="com.google.gwt.http.HTTP" />
```

in addition, since a lot of data will involve JSON encoding, you should also include the JSON module:

```
<inherits name="com.google.gwt.json.JSON" />
```

To make the REST call, we need code similar to the following:

```
RequestCallback callback = new RequestCallback()
{
 public void onResponseReceived(Request request, Response response) {
 // ... do something here ...
 }
 public void onError(Request request, Throwable exception) {
 GWT.log("Error");
 }
};
String url = ...;

RequestBuilder builder =
 new RequestBuilder(RequestBuilder.PUT, URL.encode(url));
builder.setUser("admin");
builder.setPassword("admin");
builder.setHeader("Content-Type", "application/json");
builder.setHeader("Accept", "application/json");
try {
 builder.sendRequest(null, callback);
} catch (Exception e) {
 e.printStackTrace();
 GWT.log("err");
}
```

When working with GWT REST requests during development, we may find that we get "sandboxed" when making the requests. One solution to this is to start the GWT application using the IBPM Advanced Proxy capability supplied with Business Space. A URL of the form:

```
http://localhost:9080/mum/proxy/http/127.0.0.1:8888/<PageName>.html?
gwt.codesvr=127.0.0.1:9997
```

entered into a browser will appear (to the browser) as coming from "localhost:9080" while in

reality, we are redirecting through the Proxy to `http://127.0.0.1:8888` which is the debug/development environment for GWT. The HTML web page part should be changed for the GWT project being built. The Proxy configuration of Business Space may need to be changed in order to allow pass through.

Alternatively, we can disable the "Same Origin Policy" security for a Chrome browser by starting an instance of it with the command line flag "`--disable-web-security`".

See also:

- REST Integration

## Working with JSON in GWT

When working with IBM BPM data, it is common to receive that data encoded in JSON. GWT provides JSON processing capabilities through the package called `com.google.gwt.json.*`. The GWT XML file must have a new inherit:

```
<inherits name="com.google.gwt.json.JSON"></inherits>
```

A class called `JSONParser` provides a static method called `parseStrict` which takes a JSON encoded string as input. This method returns a `JsonValue` object which can then be accessed to get the various different JSON types including:

- `JSONArray`
- `JSONBoolean`
- `JSONNumber`
- `JSONObject`
- `JSONString`

For example, on a BPM task query, the return is:

```
{
 "status": "200",
 "data":
 {
 "data":
 [
 {
 "taskAssignedTo": {"type": "User", "who": "admin"},
 "bpdName": "Repeatable events",
 "instanceCreateDate": "2011-11-21T22:04:29Z"
 ...
]
 }
}
```

To get to the actual array of data, the following can be used:

```
JsonValue v =
JSONParser.parseStrict(response.getText()).isObject().get("data").isObject().get("data");
```

## Building GWT Apps in Eclipse

It is very easy to build a GWT app in a suitably configured Eclipse environment. What follows is a suggested recipe for building a GWT development environment. First we will download a copy of Eclipse from <http://www.eclipse.org/downloads>. At the time of writing, this was 3.7.1.



The download size was approximately 175 MBytes. Make sure that you match the 32bit vs 64bit selection of installer against the JVM you plan on using. The file that was downloaded was a ZIP which was extracted into C:\IBM resulting in C:\IBM\eclipse. The eclipse.ini file was edited to increase the size of the memory available to the product. This is optional and a function of how much memory is available on your own machine.

When launched, the splash-screen for Eclipse will show.



Next we will install the Google plug-in for Eclipse. The locations and instructions for this can be found here:

<http://code.google.com/eclipse/docs/download.html>

A typical selection looks like:

Name	Version
▶ <input type="checkbox"/> Google App Engine Tools for Android (requires ADT)	
◀ <input checked="" type="checkbox"/> Google Plugin for Eclipse (required)	
<input checked="" type="checkbox"/> Google Plugin for Eclipse 3.7	2.5.1.v201201120043-rel-r37
◀ <input checked="" type="checkbox"/> GWT Designer for GPE (recommended)	
<input checked="" type="checkbox"/> GWT Designer Core	2.5.0.r37x201112291019
<input checked="" type="checkbox"/> GWT Designer Editor	2.5.0.r37x201112291029
<input checked="" type="checkbox"/> GWT Designer GPE	2.5.0.r37x201201030222
<input checked="" type="checkbox"/> WindowBuilder Core	1.2.0.r37x201112290923
<input checked="" type="checkbox"/> WindowBuilder CSS Support	1.2.0.r37x201112290951
<input checked="" type="checkbox"/> WindowBuilder XML Core (requires Eclipse WTP/WST)	1.2.0.r37x201112291009
◀ <input checked="" type="checkbox"/> SDKs	
<input checked="" type="checkbox"/> Google App Engine Java SDK 1.6.1	1.6.1.v201201120043r37
<input checked="" type="checkbox"/> Google Web Toolkit SDK 2.4.0	2.4.0.v201201120043-rel-r37

The installation time is typically less than 5 minutes

GWT provides a development set of plugins called GWT Designer/WindowsBuilder. Experience shows that the default heap size for the Java VM is probably too small. Editing the eclipse.ini

file located in the same directory as Eclipse to provide additional heap size is recommended.

configuration	6/24/2013 10:59 AM	File folder
dropins	2/4/2013 1:02 PM	File folder
features	6/24/2013 11:02 AM	File folder
p2	6/24/2013 11:00 AM	File folder
plugins	6/24/2013 11:02 AM	File folder
readme	6/24/2013 10:57 AM	File folder
.eclipseproduct	2/4/2013 11:25 AM	ECLIPSEPRODUCT...
artifacts	6/24/2013 11:02 AM	XML Document
eclipse	2/4/2013 12:05 PM	Application
<b>eclipse</b>	<b>2/4/2013 1:02 PM</b>	<b>Configuration sett...</b>
eclipsec	2/4/2013 12:05 PM	Application
epl-v10	2/4/2013 11:28 AM	Chrome HTML Do...
notice	2/4/2013 11:28 AM	Chrome HTML Do...

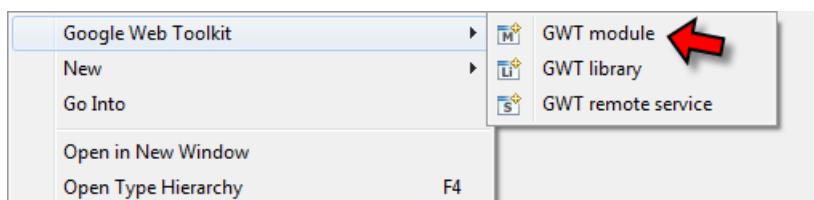
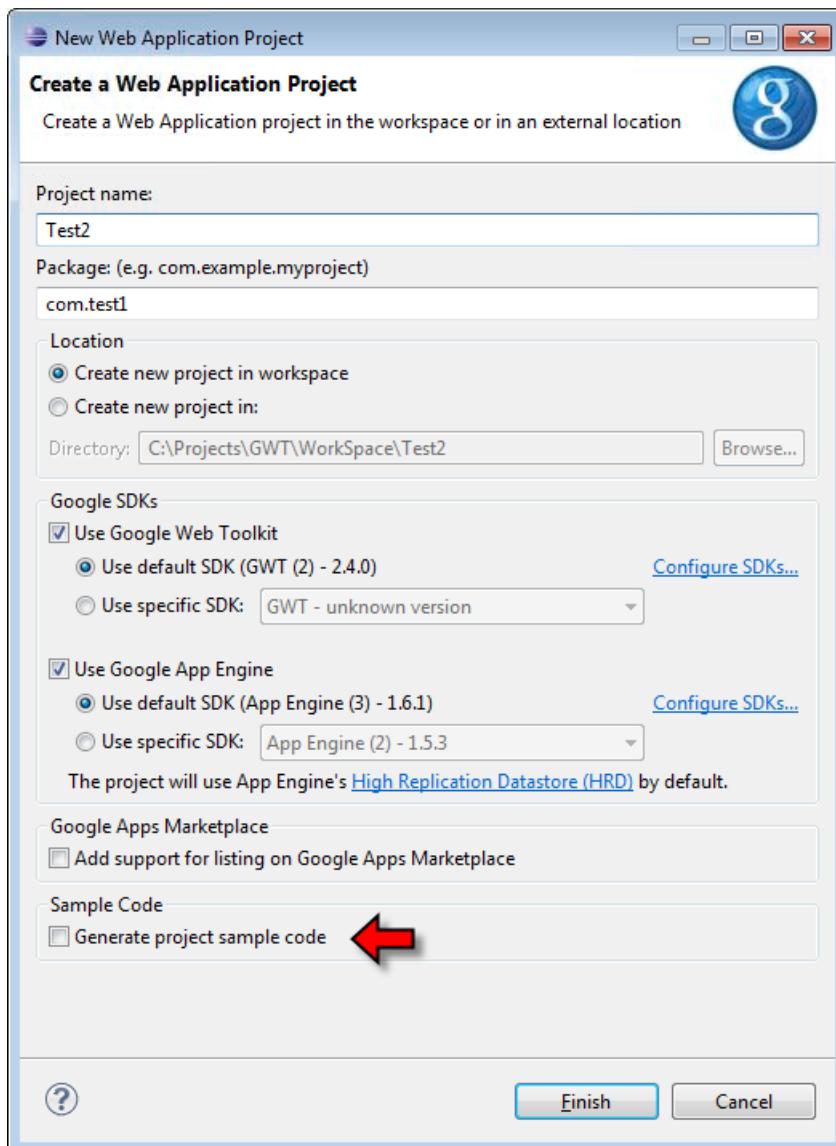
If memory permits, setting the parameters:

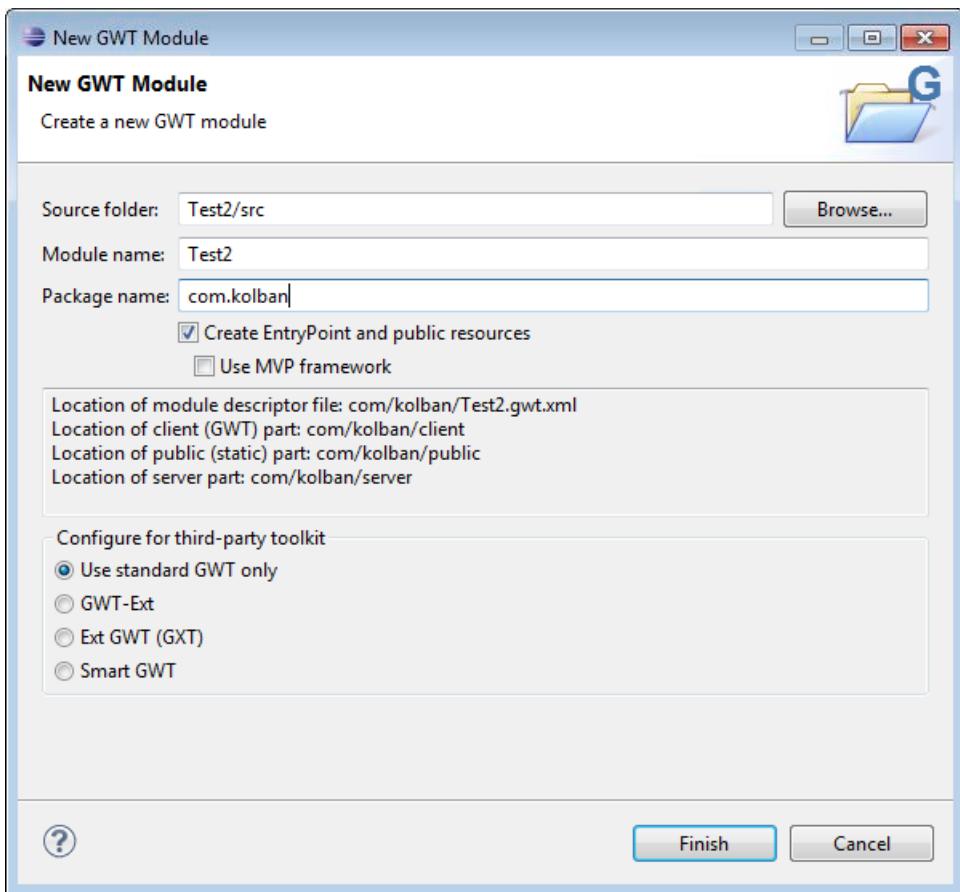
```
-Xms512m
-Xmx1024m
```

provides additional capacity. Even with these additional resources, WindowBuilder still seems to crash/freeze from time to time. Thankfully, it hasn't been seen to corrupt anything and "killing" Eclipse and restarting it seems to return the environment to stability. The mean-time-to-failure is long enough that, even though very disappointing, the occasional crash is still acceptable. The crashes were experienced on very large projects. I personally save all my work and export my projects to a ZIP file approximately every hour. This means that I can always recover from some fatal occurrence. I keep all my exports rather than overlay an old export with a new export just in case something goes *wrong* with the content of the most recent export if I attempted to recover.

To create a sample application that is ready to run, the following recipe can be followed:







In addition to installing the GWT Designer, it is suggested that the Eclipse Web Editor be also installed.

## Notes on GWT Widgets

GWT provides a rich and powerful set of pre-supplied widgets and layouts. The documentation for these can be found in the JavaDoc as well as in the Development Guides. The following sections provide some summary notes on the usage of some of these widgets.

### **RootPanel**

The RootPanel is the top level panel in an application. It has a method called `get()` which will return the RootPanel object. Alternatively, the `get (String id)` method can be used to obtain a RootPanel for a specific HTML DOM element.

**Note:** It seems that in the latest releases, one should **not** use RootPanel but instead use RootLayoutPanel. Failure to do this can result in unexpected problems. I wasted a couple of hours adding a DockLayoutPanel into a RootPanel when I should have added it into a RootLayoutPanel. It *feels* that if one uses any \*\*\*LayoutPanel widgets then the ancestor should be a RootLayoutPanel.

## **DockPanel**

The DockPanel widget is a layout widget that contains areas for North, South, East, West and Center.

## **DialogBox**

The GWT DialogBox class is meant to be subclassed to provide a dialog box. The `hide()` method disposes of it. The method `show()` will show the dialog while `center()` will show it and center it in the browser.

When using a DialogBox with UiBinder, change the new class to be

```
public class MyUI extends DialogBox ...
```

and

```
setWidget(uiBinder.createAndBindUi(this));
```

The method `setGlassEnabled(true)` will add a frosted glass background.

## **DisclosurePanel**

Somewhat similar to the StackedLayoutPanel is the DisclosurePanel. This is a widget with a header and a body. When the header is clicked, the body is shown beneath the header. When the header is clicked again, the body is once more hidden. The default DisclosurePanel has a header with a toggle button and some text. The text of the header can be accessed with the DisclosurePanel's `getHeaderTextAccessor()` method.

It has the following three CSS styles:

```
.gwt-DisclosurePanel
```

```
.gwt-DisclosurePanel-open
```

```
.gwt-DisclosurePanel-closed
```

## **Composite**

The Composite widget is a container for a single other widget. It is primarily used for creating custom widgets. When attaching the single child widget, it is important to use the method `initWidget(Widget)` to add the child widget.

## **ListBox**

The ListBox presents a scrollable list of string items. Items can be added to the ListBox with the `addItem()` method. The currently selected item can be found at the index returned by `getSelectedIndex()`. The value of the item can be retrieved with `getValue(index)`.

## **FlexTable**

The FlexTable widget is similar to Grid.

## **Grid**

The Grid widget lays out its children in rows and columns. When constructed, it defines an explicit number of rows and columns. The children can then be added into these explicit cells using the setWidget(row, column, widget) method.

Each cell in the grid can be customized using the grids's CellFormatter object.

## **HTML**

The HTML widget allows one to insert arbitrary HTML into the page.

## **PushButton**

To set the image on a PushButton we use the getUpFace() to get a CustomButton.Face object which has a setImage() method.

## **FileUpload**

The FileUpload widget will upload a file. In order to use FileUpload it **must** be included in a FormPanel container.

```
final FormPanel myForm = new FormPanel();
form.setAction("url");
form.setEncoding(FormPanel.ENCODING_MULTIPART);
form.setMethod(FormPanel.METHOD_POST);
...
FileUpload fileUpload = new FileUpload();
fileUpload.setName("MyName");
form.add(fileUpload);
form.submit();
```

When defining a form in UiBinder, the form should look like:

```
<g:FormPanel ui:field="myForm" action="/url" encoding="ENCODING_MULTIPART"
method="METHOD_POST">
```

## **GWT Cell processing**

There are a number of widgets that work with GWT Cells. Think of a cell is an extremely light-weight logical widget. To put this in context, imagine a table showing sales orders. Imagine that the table contains a 1000 rows and that each sales order has 10 columns. This means that there would be 10,000 distinct pieces of information on the screen. Each item of information is called a *cell*. If we chose a brute-force approach to solving this kind of a problem, we might consider creating a widget to represent each cell. This would mean that 10,000 widgets would be needed. Even with a lot of machine resources available, this is still an awful lot of data to manage. GWT however provides an elegant solution.

The idea is that we have data and we want to present that data. Rather than create a separate widget for each cell, we create a template widget (a Cell Widget) that knows how to "display" the appropriate data. For each cell to be displayed, rather than creating a whole new widget to display it, the Cell widget is passed the data that should be shown and the Cell widget *generates* as output the HTML needed to show the data. Specifically, this means that the Cell widget is continually reused. It does **not** own the data for the cell. Rather it owns how to visualize data for that type of cell.

- **CellList** – A list using cells as the single column content

- CellTable – A table using cells as the table rows and columns
- CellTree – A tree ...
- CellBrowser – A ????

Each of these Cell widgets implements AbstractHasData<T>

GWT provides a number of pre-supplied cell types but these can be augmented with custom cells.

#### Text Cells

- TextCell
- ClickableTextCell
- EditTextCell
- TextInputCell

#### Buttons, Checkboxes and Menu cells

- ActionCell
- ButtonCell
- CheckboxCell
- SelectionCell

#### Date cells

- DateCell
- DatePickerCell

#### Images

- ImageCell
- ImageResourceCell
- ImageLoadingCell

#### Numbers

- NumberCell

#### Compositions

- CompositionCell

#### Decorators

- IconCellDecorator

## **Cell Selection**

When cells are used to display information, we want to introduce the notion of "selection". Selection is the indication that a user has interacted with the screen and has somehow "selected" one or more cells. The architecture to support this is baked into the solution. An interface called `SelectionModel<T>` is provided which forms the basis for a variety of pre-supplied selection techniques:

- `DefaultSelectionModel`
- `SingleSelectionModel`
- `NoSelectionModel`
- `SelectionModel.AbstractSelectionModel`

The key to understanding how these work is to realize that every `SelectionModel` implements the `SelectionModel` interface which provides:

- `addSelectionChangeHandler()` - A registration function for adding callbacks that notify of selection changes
- `setSelected(T object, boolean selected)` – Flag a particular object as being either selected or not selected
- `isSelected(T object)` – Returns whether or not the specified object is selected

A widget that shows cells will provide a `setSelectionModel()` method to associate a `SelectionModel` with the widget. Once the widget and the `SelectionModel` are associated, the widget will know which cells are selected for display and if the user changes the selection, the `SelectionModel` will be updated to reflect the change causing any event handlers to be fired as needed.

```
final SingleSelectionModel<String> ssm = new SingleSelectionModel<String>();
cellList.setSelectionModel(ssm);
ssm.addSelectionChangeHandler(new SelectionChangeEvent.Handler()
{
 public void onSelectionChange(SelectionChangeEvent event)
 {
 String value = ssm.getSelectedObject();
 GWT.log("Selection changed! : " + value);
 }
});
```

The selected object for a Cell widget is always the whole object. This means that when a CellList is clicked, the clicked object is selected. In a CellTable, it is the row that is selected.

For the `SingleSelectionModel`, the single selected item is returned with `getSelectedObject()`. For the `MultiSelectionModel`, a set of selected objects is returned with `getSelectedSet()`.

## **Adding Data to a Cell widget**

In order to work, a cell widget will need to be provided data that it is to visualize in its cells. This data is provided by two methods:

```
<widget>.setRowCount(int count);
<widget>.setRowData(0, List<T>);
```

A simpler pattern to show all the data is

```
<widget>.setRowData(List<T>);
```

Another alternative model of adding data to a Cell widget is the use of a ListDataProvider. This is an object that wraps a Java List of data. The ListDataProvider also names a Cell Widget to be notified when the list data changes. Think of ListDataProvider as a controller in an MVC paradigm. The model data comes for the list and the view is the Cell Widget.

For example, imagine we have an ArrayList of data, to associate that with a Cell widget we might use:

```
ListDataProvider<T> ldp = new ListDataProvider<T>();
ldp.setList(myArrayList);
ldp.addDataDisplay(myCellWidget);
```

One of the primary advantages of this is that sorting can now be provided.

### **CellList notes**

The primary mechanism for showing a CellList is along the following lines:

```
CellList<MyType> humanServiceList =
 new CellList<MyType>(new AbstractCell<MyType>() {
 public void render(Context context, MyType value, SafeHtmlBuilder sb)
 {
 sb.appendHtmlConstant("<div>" + value.getName() + "</div>");
 }
});
```

What happens is that the render() method is called and is responsible for building an HTML fragment that will be shown in the cell area.

```
TextCell textCell = new TextCell();
CellList<String> cellList = new CellList<String>(textCell);
cellList.setRowData(DAYS);
```

### **CellTable notes**

A CellTable is a rectangular set of Columns. A concrete example of a column is a TextColumn. Generically, a Column provides a getValue() method that is given an object (for the row) and must return a value used by the cell for display.

When a CellTable is constructed, it is supplied with the type of data for a row.

Here is an example of creating a CellTable.

```
CellTable<Data1> cellTable = new CellTable<Data1>();

TextColumn<Data1> textColumnA = new TextColumn<Data1>() {
 @Override
 public String getValue(Data1 object) {
 return object.getA();
 }
};

TextColumn<Data1> textColumnB = new TextColumn<Data1>() {
 @Override
 public String getValue(Data1 object) {
 return object.getB();
 }
};

TextColumn<Data1> textColumnC = new TextColumn<Data1>() {
```

```

@Override
public String getValue(Data1 object) {
 return object.getc();
}
};

cellTable.addColumn(textColumnA, "A");
cellTable.addColumn(textColumnB, "B");
cellTable.addColumn(textColumnC, "C");

cellTable.setRowData(data1Dat);

```

To make a column sortable, we call the column's setSortable(true) method.

See also:

- [GWT Developers Guide – Cell Widgets](#)

### **DataGrid Notes**

An alternative to the CellTable is the DataGrid. DataGrid appears to be identical to the CellTable with one important distinction. The DataGrid's header remains in place during a content scroll. This means that we can scroll and page through the content of the table while the header remains perfectly in place.

### **Cell Tree Notes**

The tree structure shows parent/child oriented data in a tree structure. The core to using Cell Trees is the class known as the NodeInfo which encapsulates everything needed to show an individual node. The nodes are tied together using the TreeViewModel interface.

Imagine that a tree is composed of a tree of MyTreeNode objects. Imagine that the MyTreeNode contains:

- String value – The value of the tree node
- ArrayList<MyTreeNode> children – The children of **this** tree node

Each tree node is then wrapped in a NodeInfo object ...

TreeViewModel.DefaultNodeInfo<MyTreeNode>

This is constructed with:

```
TreeViewModel.DefaultNodeInfo<MyTreeNode>(new ListDataProvider<MyTreeNode>(children), new
TextCell<MyTreeNode>());
```

**Note:** After some study in this area it appears that CellTrees are still somewhat immature. There appears to be great trouble dynamically adding or deleting nodes.

### **GWT Modules**

From a development perspective, parts of a GWT solution can be packaged into "modules" and these modules can become black-box and reusable components. Typically a module defines:

- Other modules needed by this module
- An entry point application class

- Source path entries
- Public path entries
- Deferred binding rules

### **Naming conventions and recommendations**

There are a number of separate artifacts used in the building of a GWT based solution. Each of these have names and file types associated with them. Here are some recommendations and conventions for their use.

- Style Sheets – Style sheets end with a file suffix called ".css". It is recommended that these files be named <package>.<module>.style.css. For example, if a Module had the name com.kolban.MyModule, then the suggested CSS file would be com.kolban.MyModule.style.css. If a style sheet is used by a module, it should be placed into the public folder and referenced in the Module's XML configuration file.

### **The Entry-Point class**

A module can have one or more classes that implement EntryPoint. Any classes contained within a module that implement EntryPoint will be constructed and their onModuleLoad() methods called. For example:

```
public class XYZ implements EntryPoint {
 public void onModuleLoad() {
 GWT.log("Loaded!");
 }
}
```

### **Source Path**

The source path is the package of Java code that will be translated into JavaScript. The default for this is the "client" sub-package.

### **Public Path**

<TBD>

When using images or other static resources in a Module, change the URL for the images to be:  
GWT.getModuleBaseURL() + "imageName.png"

### **Module declaration files**

A module declaration file is an XML file with a file extension of .gwt.xml

```
<module>
 <inherits name="com.google.gwt.user.User"/>
 <inherits name="com.google.gwt.user.theme.standard.Standard"/>
 <inherits name="com.google.gwt.http.HTTP"/></inherits>
 <inherits name="com.google.gwt.json.JSON"/></inherits>
 <entry-point class="com.kolban.project.client.BPMUtils"/>
</module>
```

To include a separate package, use the <inherits> construct.

## **Working with resources**

Resources are collections of data used by a module. These can include images. The resources are based on the concept of a ClientBundle. For example:

```
public interface Resources extends ClientBundle {
 @Source("next_16x16.png")
 ImageResource nextButton();
}
```

will create a resource containing an image. To get the image:

```
Resources resources = GWT.create(Resources.class);
Image image = new Image(resources.nextButton());
```

## **GWT Layouts and resizing**

From GWT 2.0 onwards, a set of new layouts were provided that *modernize* and enhance panel layouts. Take **great** care not to use the old panels when you want to use the new panels; their names are very similar. The new panels are:

- RootLayoutPanel
- CustomScrollPanel
- DeckLayoutPanel
- DockLayoutPanel
- LayoutPanel
- ResizeLayoutPanel
- ScrollPanel
- SimpleLayoutPanel
- SplitLayoutPanel
- StackLayoutPanel
- TabLayoutPanel

When building a custom widget that is going to be involved in these layouts, base the custom widget on ResizeComposite as opposed to Composite.

### **RootLayoutPanel**

RootLayoutPanel is a singleton that contains other layout panels. Typically it is used as a replacement for RootPanel. For example, where one would commonly code:

```
RootPanel.get().add(myWidget);
```

One would now code

```
RootLayoutPanel.get().add(myWidget)
```

### **StackLayoutPanel**

The StackLayoutPanel shows a vertical set of widgets. It replaces the StackPanel widget. When a widget is clicked, it is expanded to show another widget. As such, the panel contains an array of

rows where each row is composed of two widgets. The first is a header widget used to expand an entry while the second widget shows the content.

Example:

```
StackLayoutPanel stackLayoutPanel = new StackLayoutPanel(Unit.EM);
HorizontalPanel horizontalPanel_1 = new HorizontalPanel();
stackLayoutPanel.add(horizontalPanel_1, new HTML("New Widget"), 2.0);
```

The first widget is the body, the second is the header.

```
stackLayoutPanel.add(<body>, <header>);
```

To explicitly show a particular child widget, the `showWidget()` method may be used. This can take either an ordinal value or a widget reference.

Although the `StackLayoutPanel` sounds nice in principle, it seems to have some possible challenges in practice. In a particular project, I wanted to dynamically add sections to the `StackLayoutPanel`. Unfortunately, this didn't seem to work out anywhere near like what I would have liked. Fortunately, the `StackLayout` widget was also available.

## **TabLayoutPanel**

The `TabLayoutPanel` provides a set of tabs that can be used to switch between different widgets. This widget replaces the `TabPanel` widget which is now deprecated.

### **Things that seem to scroll ok**

A `DockLayoutPanel` containing a `ScrollPanel` containing a `Vertical Panel` containing added labels.

Another is a `FlowPanel` > (Dynamic Disclosure Panels)

## **UIBinder – XML descriptions of screens**

So far we have spoken about writing UI code directly in Java. GWT provides an alternative and, arguably, better way of describing UIs. This is through the use of an XML document. The XML declares what the UI should look like and is used as an alternative to the code.

At its heart, a `UIBinder` xml file looks like:

```
<!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
 xmlns:g="urn:import:com.google.gwt.user.client.ui">
 <ui:style>
 .important {
 font-weight: bold;
 }
 </ui:style>
 <g:VerticalPanel>
 <g:Button ui:field="myButton">Hello World</g:Button>
 <g:Label text="New Label"/>
 <g:ListBox visibleItemCount="5"/>
 </g:VerticalPanel>
</ui:UiBinder>
```

Imagine we wish to create a UI component called XYZ. We will create an XML file called `XYZ.ui.xml` and from that a Java class called `XYZ.java` be created. Eclipse Window Builder

understands UI Binder so we do not lose UI drawing capabilities.

In the Java code we will build:

```
public class XYZ extends Composite {
 private static XYZUiBinder uiBinder = GWT.create(XYZUiBinder.class);
 @UiField Button myButton;

 interface XYZUiBinder extends UiBinder<Widget, XYZ> {
 }

 public XYZ() {
 initWidget(uiBinder.createAndBindUi(this));
 }

 @UiHandler("myButton")
 void onMyButtonClick(ClickEvent event) {
 }
}
```

When a Widget is used inside a ui.xml file, any property of that widget can supplied as an attribute in the XML.

UiBinder can also save a lot of time/effort with event handlers using the @UiHandler

See:

- [Declarative Layout with UI Binder](#)

## Using CSS for changing the appearance of UI

CSS is used extensively for changing the appearance of the UI components. A standard CSS can be found in <project>/war/<project>/gwt/standard/standard.css.

A gradient background can be added to a CSS by using this really-cool web site to build the CSS:

<http://www.colorzilla.com/gradient-editor/>

The CSS styling of a Widget is derived from UIObject which is an ancestor of Widget. UIObject contains properties called:

- styleName
- styleDependentName
- stylePrimaryName
- styleElement
- stylePrimaryName

GWT comes with a number of built-in "themes" that provide default styling for the widgets. These are:

- com.google.gwt.user.theme.standard.Standard
- com.google.gwt.user.theme.clean.Clean
- com.google.gwt.user.theme.chrome.Chrome
- com.google.gwt.user.theme.dark.Dark

The "Standard" theme is the default.

## The GWT Editor Framework – Mapping data to widgets

A JavaBean object is one which has properties that adhere to the JavaBean specification. This means that they have getters and setters.

An Editor is an object that supports editing some (or all) of the properties of a JavaBean. An editor can contain sub-editors.

SimpleBeanEditorDriver

See also:

- GWT Wiki – [Editor Framework](#)

## Working with XML in GWT

GWT has built in support for working with XML documents. One needs to declare intent to use the XML package by inserting:

```
<inherits name="com.google.gwt.xml.XML" />
```

In the Java code of the module, one can then call:

```
Document myDocument = XMLParser.parse(xmlString);
```

This will return a DOM document that can then be manipulated. A useful function on XMLParser is called `removeWhitespace()` which will remove empty TEXT nodes from the document.

In the module XML file.

See also:

- [Coding Basics – Working with XML](#)

## Integrating with native JavaScript

Native JavaScript can be utilized within the context of a GWT application. To define a method that implements a call to JavaScript we can code:

```
public class MyClass
{
 public native void sayHello(String name)
 {
 /*-
 * alert("Hi there " + name);
 */
 }
}
```

## Building custom GWT Widgets

It is common to build custom GWT Widgets both for re-use amongst multiple projects or simply for comprehension within a single project. Building coarse grained UI with lots of function vs fine grained UIs with re-usable components is a user's choice. When building custom widgets, some common factors apply and this section describes some of them.

### ***Adding Event Handling***

Let us start with some basic notions. First, when something interesting occurs in a widget, we want

that widget to "generate" an Event. In GWT, the Event is going to be represented by an Object which has a specific format.

Comment: Should the Handlers actually be called Listeners? For example, imagine an event that happens when a "task is selected". This will create a TaskSelected event. What kind of implementation should be notified? There seems to be two possibilities. The first is a "TaskSelectedHandler" and the other is a "TaskSelectedListener" both of which will have a method called "onTaskSelected". The consumer will code:

```
new TaskSelectedXXX()
{
 public void onTaskSelected()
 {
 ...
 }
}
```

Personally, I think I like the idea of a TaskSelectedListener. The word Listener (as opposed to Handler) gives the impression that we are listening/watching for events to be processed. Studying the events supplied with GWT itself, they use the Handler style.

Generically, the new Event will look like:

#### <Custom>EventHandler

```
public interface CustomEventHandler extends EventHandler {
 void onCustom(CustomEvent event);
}
```

Note: I have seen an interesting idea which is to add the XXXHandler interface as a type inside the XXX class.

#### <Custom>Event

```
public class CustomEvent extends GwtEvent<CustomEventHandler> {
 public static Type<CustomEventHandler> TYPE = new Type<CustomEventHandler>();
 private String message;

 public CustomEvent(String message)
 {
 this.message = message;
 }

 public String getMessage()
 {
 return message;
 }

 @Override
 public GwtEvent.Type<CustomEventHandler> getAssociatedType() {
 return TYPE;
 }

 @Override
 protected void dispatch(CustomEventHandler handler) {
 handler.onCustom(this);
 }
}
```

## Custom1

```

public class Custom1 extends Composite implements HasHandlers {
 private HandlerManager handlerManager;
 // maybe
 // private HandlerManager handlerManager = new HandlerManager(this);

 public Custom1() {
 handlerManager = new HandlerManager(this);
 }

 @Override
 public void fireEvent(GwtEvent<?> event) {
 handlerManager.fireEvent(event);
 }

 public HandlerRegistration addCustomEventHandler(CustomEventHandler handler)
 {
 return handlerManager.addHandler(CustomEvent.TYPE, handler);
 }

 public void doSomething()
 {
 /* ... */
 fireEvent(new CustomEvent("hello!"));
 }
}

```

In summary, to handle events:

- CustomEventHandler
- CustomEvent
- Custom1
- Experimental

As a test, I put THIS code in the Class that owns the publication of the event. This may be the easiest way to cookie cut.

```

private HandlerManager handlerManager = new HandlerManager(this);

public static interface SelectionChangedHandler extends EventHandler {
 void onSelectionChanged(SelectionChangedEvent event);
}

public HandlerRegistration addSelectionChangedHandler(SelectionChangedHandler
handler) {
 return handlerManager.addHandler(SelectionChangedEvent.TYPE, handler);
}

public static class SelectionChangedEvent extends
GwtEvent<SelectionChangedHandler> {
 public static Type<SelectionChangedHandler> TYPE = new
Type<SelectionChangedHandler>();

 public GwtEvent.Type<SelectionChangedHandler> getAssociatedType() {
 return TYPE;
 }

 protected void dispatch(SelectionChangedHandler handler) {

```

```

 handler.onSelectionChanged(this);
 }
}

```

With this infrastructure set up, what remains is the use of the event processing system. Imagine now that someone creates an instance of Custom1 and wants to be notified when the CustomEvent (that is fired inside Custom1) happens.

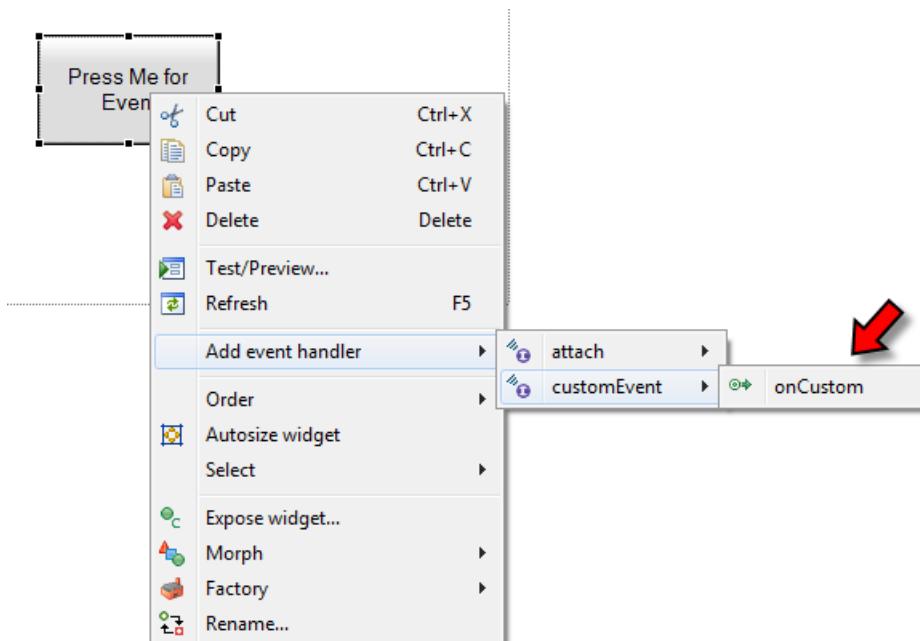
Here is an example of how this might be used:

```

Custom1 custom1 = new Custom1();
custom1.addCustomEventHandler(new CustomEventHandler() {
 public void onCustom(CustomEvent event) {
 GWT.log("Custom event fired!" + event.toDebugString());
 }
});

```

When using Eclipse GWT Designer, this code can be added through the tooling as follows:



There is one final item that needs to be considered and that is *when* does the event actually get fired? Somewhere in the code of the Widget that can *generate* such events there must be code similar to the following:

```
fireEvent(new CustomEvent("hello!"));
```

When executed, it will fire the CustomEvent event.

## Using Google Charts with GWT

Google has provided a rich set of charting functions that are primarily leveragable through JavaScript from ordinary web pages. A binding has been built that allows these charts to be used through GWT. To use these functions, the gwt-visualization.jar file must be downloaded and added to the project's classpath. The .gwt.xml file must include the com.google.gwt.visualization.Visualization package:

```
<inherits name='com.google.gwt.visualization.Visualization' />
public class ChartTests implements EntryPoint {
```

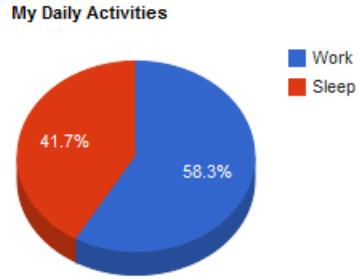
```

public void onModuleLoad() {
 Runnable onLoadCallback = new Runnable()
 {
 @Override
 public void run() {
 RootPanel panel = RootPanel.get();
 PieChart pie = new PieChart(createTable(), createOptions());
 panel.add(pie);
 }
 };
 VisualizationUtils.loadVisualizationApi(onLoadCallback, PieChart.PACKAGE);
}

private AbstractDataTable createTable() {
 DataTable data = DataTable.create();
 data.addColumn(ColumnType.STRING, "Task");
 data.addColumn(ColumnType.NUMBER, "Hours per Day");
 data.addRows(2);
 data.setValue(0, 0, "Work");
 data.setValue(0, 1, 14);
 data.setValue(1, 0, "Sleep");
 data.setValue(1, 1, 10);
 return data;
}

private PieOptions createOptions() {
 PieOptions options = PieOptions.create();
 options.setWidth(400);
 options.setHeight(240);
 options.set3D(true);
 options.setTitle("My Daily Activities");
 return options;
}
}

```



One of the core concepts behind Google Charts is that of the DataTable. The DataTable is the holder of data that is to be visualized in the chart. Logically, the DataTable is a rectangular table of rows and columns. Each column has some properties which include:

- A data type. The types available are:
  - Boolean
  - Date
  - Date/Time
  - Number
  - String

- Time of day
- An optional ID
- An optional label

To use the Google Charts, they have to be dynamically loaded before use. This is achieved via the code snippet that looks as follows:

```
VisualizationUtils.loadVisualizationApi(onLoadCallback, XXX.PACKAGE);
```

The onLoadCallback is an implementation of the Java Runnable interface. Its run method is called when the charts have been loaded. Each chart type has a .PACKAGE in its class and should be specified for loading.

Notes:

- It appears that the usage of Google Charts requires an Internet connection to access Google's Web sites. This implies that non-Internet accessible browsers won't work. This may be the case in enterprises with firewalls in place.

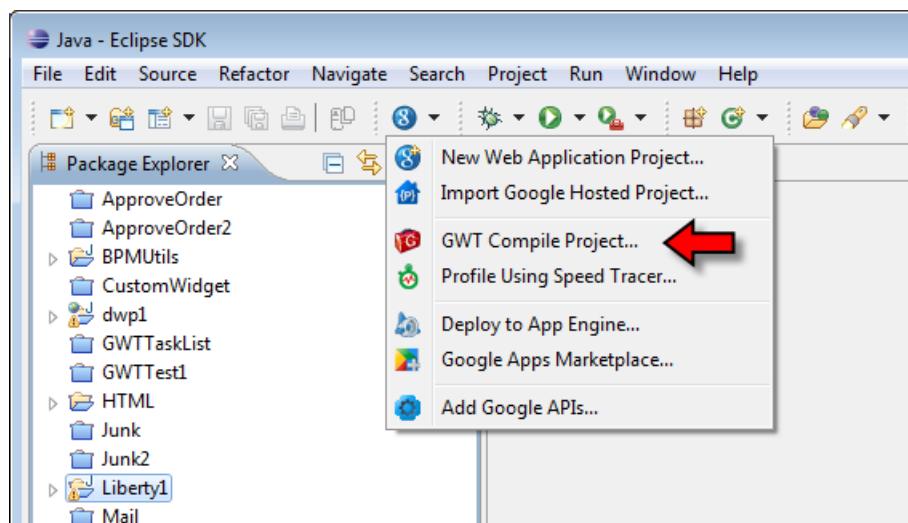
See

- [Google API Libraries for Google Web Toolkit \(GWT\)](#)
- [Google Chart Tools](#)

## Deploying a GWT application

So far we have only examined what it takes to build a GWT based application. Now we need to consider what it takes to actually deploy one. During development, we are coding and testing in a Java code environment. The WindowBuilder IDE built into Eclipse is running the solution as Java. When a solution is built, that solution is "cross-compiled" to JavaScript to run natively within the browser. This is a piece of magic that is brought to us courtesy of the wizards at Google.

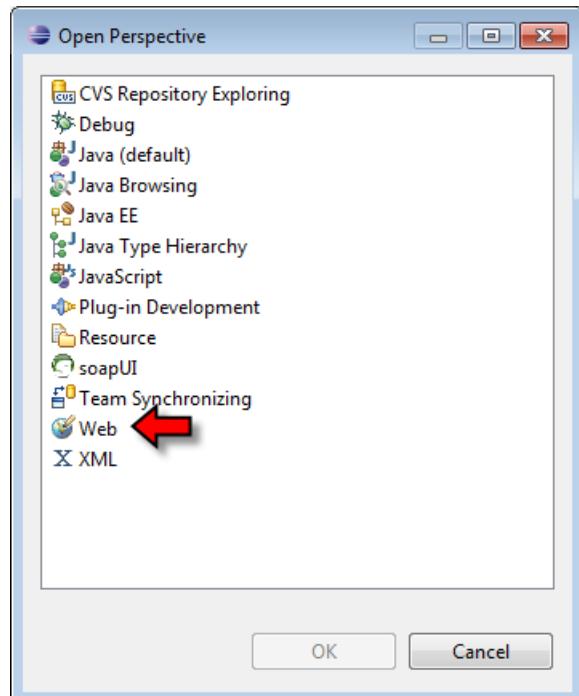
The compilation of the project is best done through Eclipse:



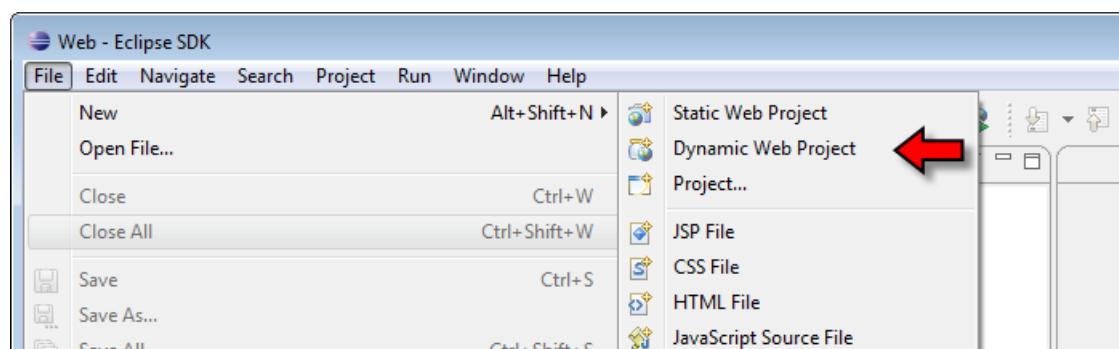
We will start by realizing that the compiled artifacts have to be deployed on a Web page. Because this book is about IBPM, the web server that hosts the web pages will usually be IBPM itself. This will allow access to the the REST requests for the "same host" principal. To achieve this, we want

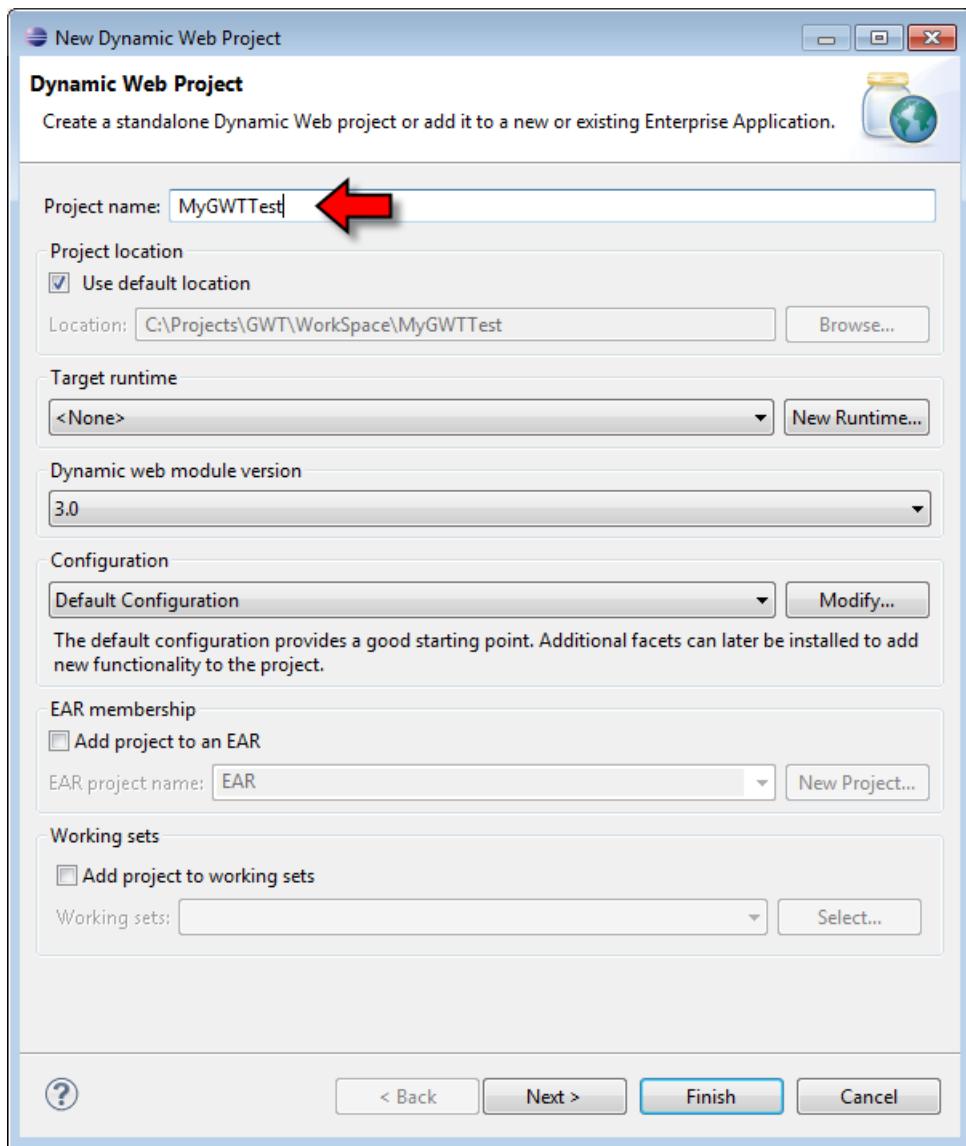
to create a Java EE WAR file that can then be deployed to the WAS server. This WAR file will contain the artifacts that are built in the GWT environment.

After switching to the Eclipse Web perspective:

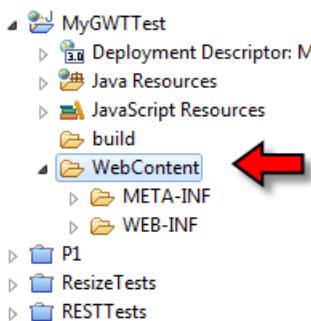


We can now create a Dynamic Web Project. This will eventually result in a WAR.



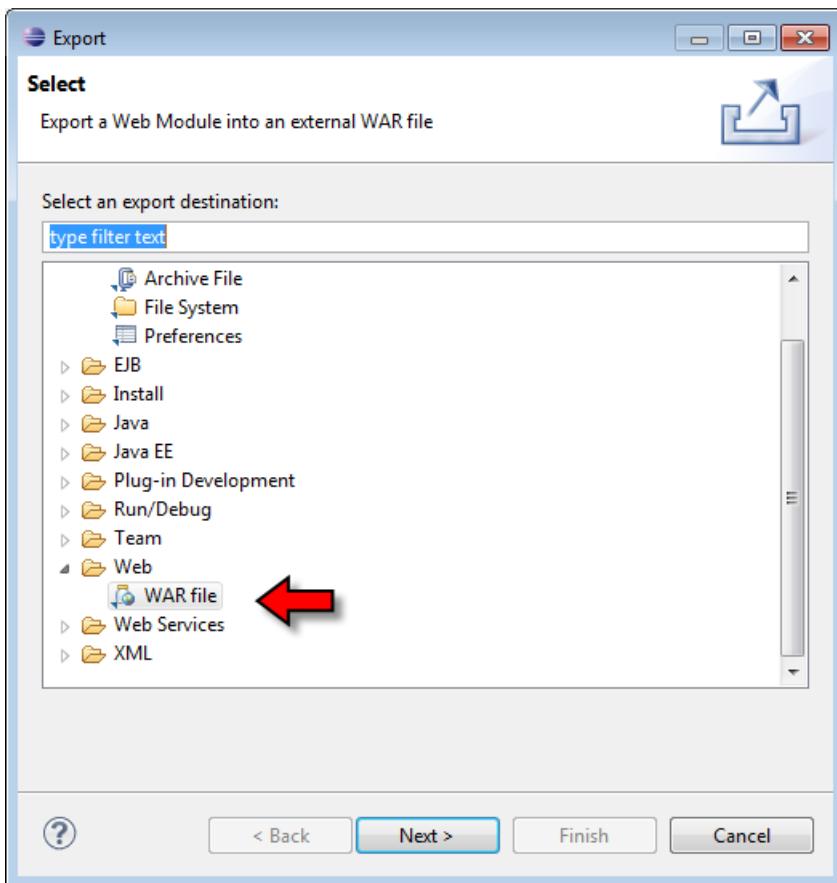


The end result of this will be a new project which contains a folder called WebContent.

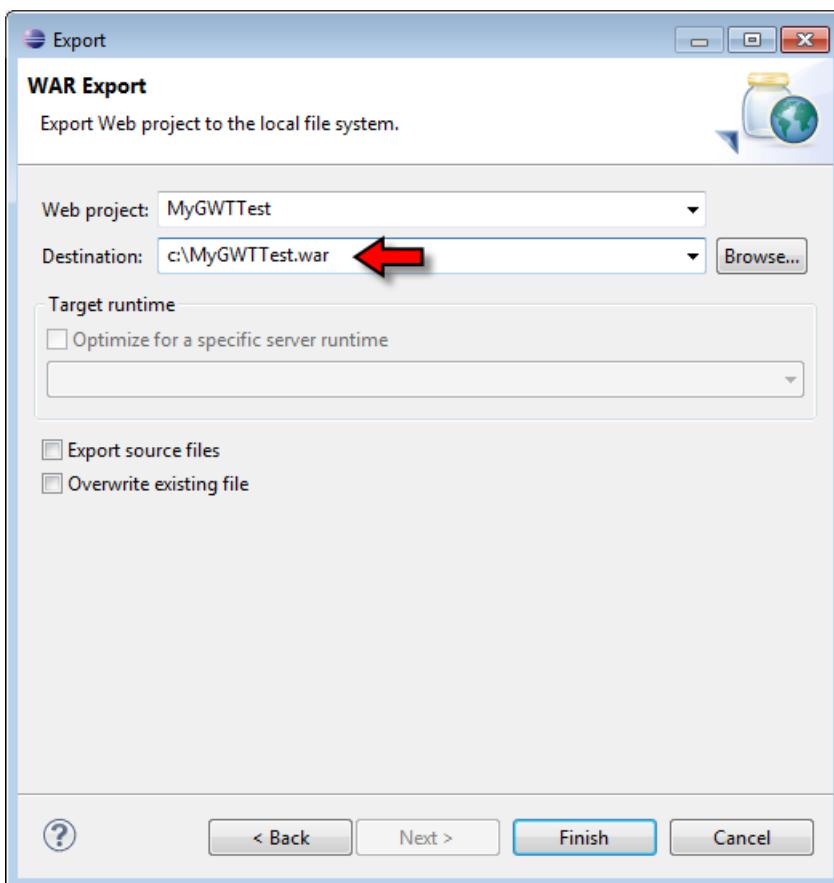


Copy the compiled results of the GWT war folder into the WebContent folder with the **exception** of the folder called WEB-INF.

Once done, we can now generate and export the WAR file:



The name of a file on the file system will be asked for. This is the file into which the resulting WAR file will be written. Be sure to give it the suffix ".war" to mark it as a Java EE WAR file.



Once the WAR file has been exported, our next task is to import it into WAS. This can be done through the IBM admin console (ISC). Under the Applications section, select New Application and select the path to the exported WAR file.

The screenshot shows the ISC interface. On the left, a sidebar lists various sections like Welcome, Guided Activities, Servers, Applications (with 'New Application' highlighted by a red arrow), Application Types, SCA modules, Services, Resources, Security, Environment, Integration Applications, and System administration. The main panel is titled 'Preparing for the application installation' and contains a form for specifying the application path. It includes fields for 'Path to the new application' (radio buttons for 'Local file system' and 'Remote file system'), a 'Full path' input field containing 'C:\MyGWTTest.war' (with a red arrow pointing to the 'Browse...' button), and 'Next' and 'Cancel' buttons at the bottom.

Most of the defaults can be taken but I suggest changing the Context Root. This is where the application root path in the URL directory tree will be set.

The screenshot shows the 'Install New Application' wizard at step 4. The left sidebar lists steps 1-5. The main panel is titled 'Map context roots for Web modules' and displays a table for mapping web modules to context roots. The table has columns for 'Web module', 'URI', and 'Context Root'. One row shows 'MyGWTTest.war' with 'MyGWTTest.war,WEB-INF/web.xml' in the URI column and '/MyGWTTest' in the Context Root column (highlighted by a red arrow). Buttons for 'Previous', 'Next', and 'Cancel' are at the bottom.

Once the installation of the WAR has been complete, we will find it installed but stopped. The default state for a new application installed into WAS is stopped. We want to explicitly start the application.

The screenshot shows the IBM BPM Web API console. It lists deployed applications in a table. One entry is 'MyGWTTest.war' with a red 'X' icon next to it, indicating it is not running. Below the table, it says 'Page: 1 of 2' and 'Total 31'.

Once started it will show "green" in the console.

The screenshot shows the IBM BPM Web API console again. The 'MyGWTTest.war' application now has a green icon next to it, indicating it is running. Below the table, it says 'Page: 1 of 2' and 'Total 31'.

We are now able to start the GWT app by connecting to the WAS server at the content root/page.

<http://localhost:9080/MyGWTTest/MyTest.html>

The compilation of a large project can take a little time.

## Debugging and Logging GWT applications

A package called gwt-log is available which provides the most comprehensive debugging for GWT applications.

To get running, download the gwt-log-x.x.x.jar file. Add it to the build path of the Eclipse GWT project. In the gwt.xml file add an entry that looks like:

```
<inherits name="com.allen_sauer.gwt.log.gwt-log-DEBUG" />
```

Finally, in the code of the application, add:

```
Log.info("Hello World!");
```

## GWT Libraries – Smart GWT

GWT has a number of extension libraries available for it. These are 3<sup>rd</sup> party code built on top of the base GWT that adds significant function on top of the base GWT environment.

Smart GWT appears to be one of the most prevalent and feature rich additional library sets available for GWT. It can be downloaded from:

<http://code.google.com/p/smartgwt/>

A quick-start guide (PDF) can be found here:

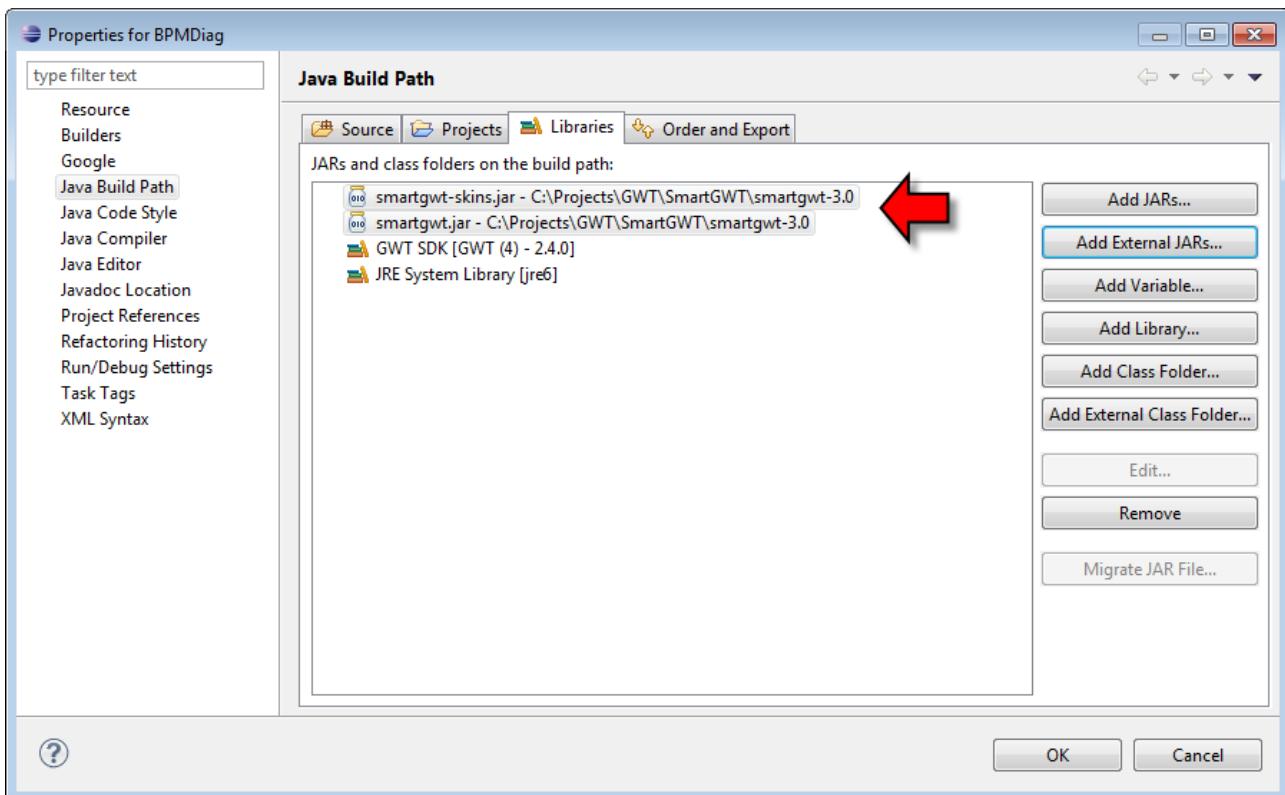
[http://www.smartclient.com/releases/SmartGWT\\_Quick\\_Start\\_Guide.pdf](http://www.smartclient.com/releases/SmartGWT_Quick_Start_Guide.pdf)

The version of SmartGWT available now (2012-01) is v3.0. This version appears to have done something rather odd. It has removed support for GWT Designer. What this means is that visual composition in the Eclipse tool can not be performed. Previous versions provided this support. This seems a very strange state of affairs. The developers claim that hand coding the Smart GWT widgets is an easy proposition.

To prepare an application for using SmartGWT, two JARs must be added to a project's class path. These are:

- smartgwt.jar
- smartgwt-skins.jar

These can be added from the project build path:



The .gwt.xml file should be updated to include:

```
<inherits name="com.smartgwt.SmartGwt"/>
```

### **Smart GWT – ListGrid**

The ListGrid is the Smart GWT version of a visual table and what a table it is. At a high level, here is the recipe for use:

- Create an instance of a ListGrid
- For each column that you want, add a new ListGridField using the setFields () method
- For each row of data to be show, create a ListGridRecord and add it using the setData () method

Here is an example:

```
public void onModuleLoad() {
 Canvas canvas = new Canvas();
 ListGrid listGrid = new ListGrid();
 ListGridField c1ListGridField = new ListGridField("c1", "Column C1");
 ListGridField c2ListGridField = new ListGridField("c2", "Column C2");
 listGrid.setFields(new ListGridField[] { c1ListGridField, c2ListGridField });
 canvas.addChild(listGrid);
 listGrid.setRect(6, 6, 374, 228);
 ArrayList<ListGridRecord> al = new ArrayList<ListGridRecord>();
 ListGridRecord lgr = new ListGridRecord();
 lgr.setAttribute("c1", "c1val");
 lgr.setAttribute("c2", "c2val");
```

```

al.add(lgr);
lgr = new ListGridRecord();
lgr.setAttribute("c1", "c3val");
lgr.setAttribute("c2", "c4val");
al.add(lgr);
listGrid.setData(al.toArray(new ListGridRecord[0]));

canvas.draw();
}

```

This produces the following:

Column C1	Column C2
c1val	c2val
c3val	c4val

A ListGrid can show arbitrary components in each of the cells. This is achieved by over-riding the `createRecordComponent` method of the ListGrid. Here is a quick example showing buttons:

```

ListGrid listGrid = new ListGrid()
{
 @Override
 protected Canvas createRecordComponent(ListGridRecord record, Integer colNum)
 {
 String fieldName = this.getFieldName(colNum);
 System.out.println("FieldName: " + fieldName);
 if (fieldName.equals("c1"))
 {
 IButton button = new IButton();
 button.setTitle("Press Me");
 return button;
 }
 return super.createRecordComponent(record, colNum);
 }
};

listGrid.setShowRecordComponents(true);
listGrid.setShowRecordComponentsByCell(true);

```

Column C1	Column C2
Press Me	c2val
Press Me	c4val

## Smart GWT – DataSource

One of the more interesting capabilities of Smart GWT is the concept of a DataSource. This is a container for structure descriptions **and** data that can be passed in to visualization widgets that can then show the data.

A DataSource description consists of a set of DataSourceFields.

A DataSource field contains:

Property	Value
name	Identifier field
type	One of: <ul style="list-style-type: none"><li>• text</li><li>• integer</li><li>• float</li><li>• boolean</li><li>• date</li><li>• sequence</li></ul>
title	Label for the field
length	
hidden	
detail	
required	
valueMap	
editorType	
primaryKey	
foreignkey	
rootValue	

To create a DataSource in client side Java, we perform the following:

- Create an instance of a DataSource object
- Set the ID property
- Create instances of DataSourceField for each field in the DataSource
- call the setFields() method of the DataSource to add each of the DataSourceFields

Here is some working code that will create a ListGrid and set a DataSource against it:

```
public void onModuleLoad() {
 dataSource.setClientOnly(true);
 DataSourceTextField c1DSField = new DataSourceTextField("c1", "c1");
 dataSource.addField(c1DSField);
 DataSourceTextField c2DSField = new DataSourceTextField("c2", "c2");
 dataSource.addField(c2DSField);

 ListGridRecord record = new ListGridRecord();
 record.setAttribute("c1", "c1Val");
 record.setAttribute("c2", "c2Val");
 dataSource.addData(record);

 Canvas canvas = new Canvas();

 ListGrid listGrid = new ListGrid();

 ListGridField c1ListGridField = new ListGridField("c1", "c1");
 ListGridField c2ListGridField = new ListGridField("c2", "c2");

 listGrid.setFields(new ListGridField[] { c1ListGridField, c2ListGridField });
```

```

listGrid.setDataSource(dataSource);
listGrid.setAutoFetchData(true);
canvas.addChild(listGrid);
canvas.draw();
}

```

The end result of this is:

c1	c2
c1Val	c2Val

See also:

- [Showcase](#)
- [code.google.com – SmartGWT](#)

### **Smart GWT – Drawing**

Drawing is the notion of drawing items (usually geometry) on a canvas area. SmartGWT provides just that capability using the DrawPane technology. In order to use this, an additional entry must be added to the GWT XML file:

```

<inherits name="com.smartgwt.Drawing"/>

final DrawPane pane = new DrawPane();
pane.setWidth(500);
pane.setHeight(500);
pane.setShowEdges(true);
pane.draw();

```

### **Integrating GWT with IBPM Coaches**

One interesting area of potential GWT usage is integrating GWT with IBM Coach technology. This would mean having a Coach appear which includes GWT capabilities. There are a number of ways that this could work and this section describes some of the tests and experiences.

The first question is how do we get GWT included in a Coach page?

The answer to this is that we insert a Custom HTML script that contains the following:

```

<script type="text/javascript" language="javascript"
src="http://localhost:9080/mum/proxy/http/localhost:8888/coachinclude/coachinclude.nocache.js">
</script>


```

This is going to take a little explaining. Here I am assuming that the GWT code is still in the Eclipse development environment. That means that the JavaScript for GWT can be found at:

<http://localhost:8888/<name>/<name>.nocache.js>

This is the default runtime for GWT testing. However, notice that the hostname/port isn't the same as that of the BPM server. As a result of that, things are likely to get wrong from a networking sandbox perspective. To solve that problem, we use the Web Proxy that is shipped as part of Business Space.

The last line of the Custom HTML is an HTML <span> tag with an "id". This is the anchor where the GWT will insert its content. For example,

```

public void onModuleLoad() {
 RootPanel rootPanel = RootPanel.get("GWTHERE");
 if (rootPanel == null)
 {
 GWT.log("Unable to find panel anchor point");
 return;
 }
 Label label = new Label("Hi World");
 rootPanel.add(label);
}

```

In a simple page, this results in:



### ***Using tables with GWT***

GWT provides table viewers for displaying tabular data. If we want to leverage a GWT table within a Coach, how might we go about that? The first puzzle is to determine how to get data from the Human Service into the Coach so that it can be rendered by GWT? Native Coach technology seems to build an HTML table in-line (i.e. no Ajax) using HTML <TD> tags. This doesn't appear to be of any use to us. What we really want is the data that is to be shown in the table to be supplied to us for display.

## Kolban's GWT BPM Utils

GWT is a powerful and flexible toolkit that allows a user interface developer to build just about anything. When it comes to building custom UIs for IBPM, experience is showing that the same patterns occur again and again. Instead of starting from scratch, a set of GWT libraries has been built that attempt to address some of the common occurrences. These are still a work in progress but the author ([kolban@us.ibm.com](mailto:kolban@us.ibm.com)) is very keen to work on these and hear back from users who have tried to use them or are looking for enhancements. The code is provided "as-is" in source format but any changes to the code should be fed back for inclusion for all.

The GWT BPM Utils provide the following functions:

- Visual Task Lists
- REST wrappers and helpers

### Visual Task Lists

A common need is to present a list of tasks that a user is eligible to work upon. A GUI widget called the `StackedTaskList` presents such a list. This widget provides three visual containers. One for tasks that are `Owned`, one for tasks that are `Available` and one for Tasks that are `Completed`.



For each task found in the IBPM run-time, an entry is added to the corresponding container. The format of an entry is composed of two parts. The first is a header which is always shown that provides a quick summary of the task. The second entry is the body which is shown when the user expands the header. The thinking behind this is to provide a summary of tasks and, for tasks that a user wishes to know more about, the details can be displayed.

A screenshot of the StackedTaskList interface. The header shows '(22) Owned', '(0) Available', and '(5) Closed'. Below the header, there are two expanded sections. The 'Owned' section shows two rows of task details: 'taskId: 8 InstanceId: 9 taskSubject: Step: Untitled' and 'taskId: 160 InstanceId: 160 taskSubject: Step: Step 1'. The 'Available' section is collapsed. At the bottom, there are icons for search, refresh, and other actions, along with a list of task details: 'taskId=160.0' and 'taskDueDate=2013-06-24T23:03:16Z'.

- taskId=160.0
- taskDueDate=2013-06-24T23:03:16Z

The implementation of this widget exposes a method called `searchTasks()`. When this method is called, a search is executed against the server and the results displayed in the widget. To refine the search, a set of conditions can be supplied (`setSearchConditions()`) which will be used to choose which results are returned. To choose which individual columns to return, a list of column names can also be supplied (`setSearchColumns()`).

The entries for each task are made up of two further Widgets which are also customizable. These widgets should inherit from `SearchResultViewer`. One widget will be used for the header and the other for the body. Users of the `StackedTaskList` can implement their own viewer widgets. The contract with the `SearchResultViewer` is simple. First it inherits from GWT

Component so any content visualized there will be displayed in the appropriate area. An implementation of such a class must implement the `newInstance()` method. This returns a new instance of the `SearchResultViewer`. This is needed to create new instances, one for each task entry. To display content associated with the task, the `setSearchResult()` method is called which has a `SearchResult` object passed to it. The `SearchResult` contains the details of the task as returned from the search. The implementer of the `SearchResultViewer` is expected to visualize chosen data in the implementation.

In addition to the body for the viewer, each task entry has a button in it to allow the task to be "selected". A callback can be registered to be invoked when the button is pressed.

## REST wrappers and helpers

The REST wrappers and helpers provide high level interfaces to the IBPM functions exposed through REST. These include:

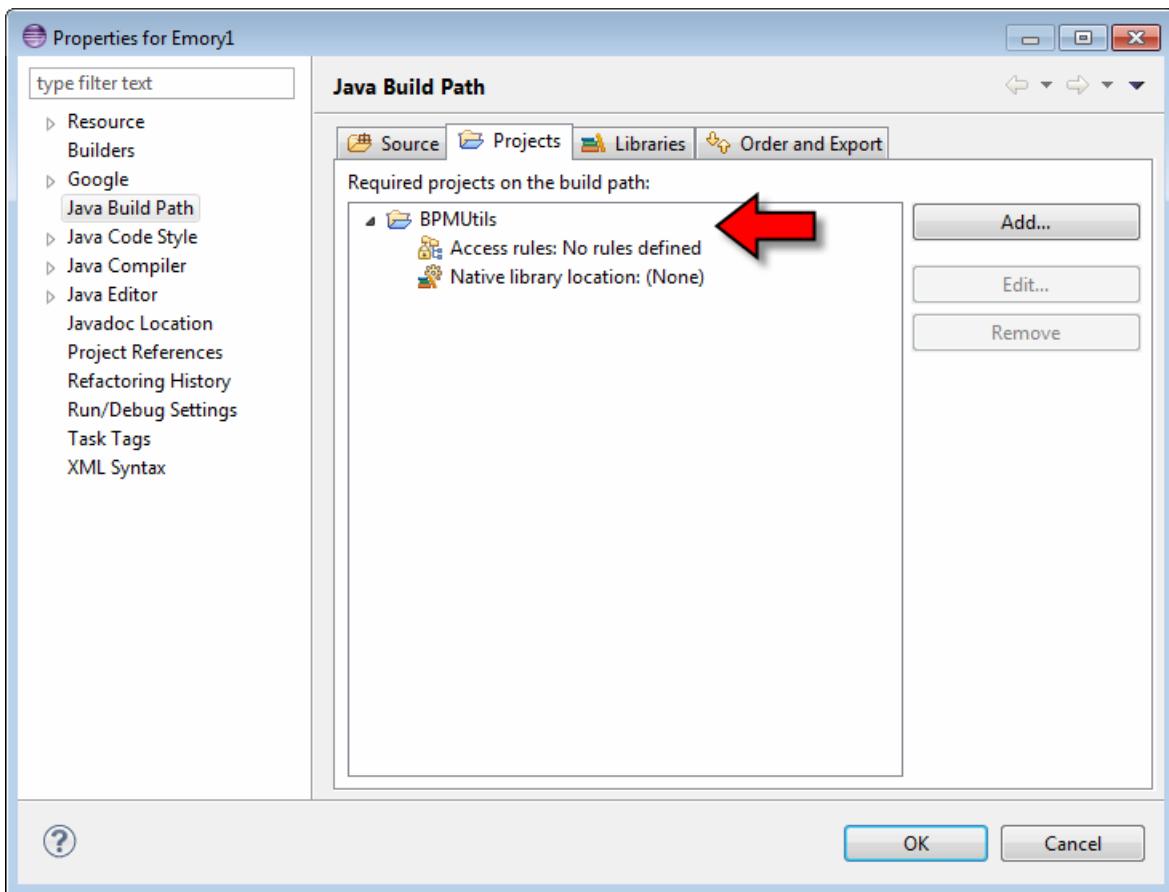
assignTask	
claimTask	
completeTask	Complete the task with returned data
executeSearch	Execute a search for tasks and instances
exposed	
getTask	Retrieve details of a specific task
porocessApps	
startService	Run an instance of a specific service
userList	

The way one appears to use this toolkit is:

1. Add an inherits of the BPMUtils module into the XML file for your current project  
eg. Add:

```
<inherits name="com.kolban.project.BPMUtils"/>
```

2. Add the BPMUtils project as a dependency at the Eclipse level when building



3. Use the REST APIs
- 4.
- 5.

## Examples

Using the GWT BPM Utils is pretty straightforward. This section provides some examples of some of the more common usage patterns:

### ***Showing a visual data table***

Consider a data type within a Process Application that looks as follows:

## Business Object

The screenshot shows a configuration interface for a business object named "OrderItem". The "Common" section displays system details: System ID (guid:56d35d2f221c8d0e-5a405e), Modified (admin Jan 20, 2012 2:04:45 PM), and Documentation (empty). The "Parameters" section lists three variables: name (String), color (String), and quantity (Integer).

It represents an item in an order. Now imagine that we have a variable in a process which is a list of such items. In our UI, we wish to present a table of these items. The end result should look like:

Name	Color	Quantity
Widget X	Green	10
Widget Y	Blue	20

One way to achieve this is to use the `getTask()` method which returns a `TaskDetails` object. This object contains a property called "variables" (retrieved with the `getVariables()` method). The data type of this property is a `JSONObject` which contains all the variables of a task instance. For a list object owned by the process, an example of data returned would be:

```
{
 "order": {
 "selected": [],
 "items": [
 {
 "name": "Widget X",
 "color": "Green",
 "quantity": 10
 },
 {
 "name": "Widget Y",
 "color": "Blue",
 "quantity": 20
 }
]
 }
}
```

Take a few minutes to review the structure. See how a list variable contains a child called `items` that is an array of objects. This is important as access to the data is *not* achieved by directly accessing the variable but rather its `items` child. A list can also indicate which (if any) of the items within it are flagged as “selected”. A list variable has a property called “selected” which is a list of integers. Think of the list as having an ordinal set of entries. If the ordinal value appears in the selected set,

then the list entry is considered selected.

Now let us look at how we can map this to a visual table.

First we create a Java Class that will act as a holder for each item. This class looks as follows:

```
public class OrderItem
{
 public String name;
 public String color;
 public int quantity;
}
```

Now we can look at the code that utilizes the GWT BPM library to retrieve the content of a task:

```
REST rest = new REST();
int taskId = // The taskId to be retrieved
rest.getTask(taskId, new TaskDetailsResultsCB() {
 public void onTaskDetailsResults(TaskDetails taskDetail) {
 JSONObject variablesObject = taskDetail.getVariables();
 JSONObject orderJO = variablesObject.get("order").isObject();
 JSONArray itemsJA = orderJO.get("items").isArray();
 ArrayList<OrderItem> al = new ArrayList<OrderItem>();
 for (int i=0; i<itemsJA.size(); i++)
 {
 JSONObject itemJO = itemsJA.get(i).isObject();
 OrderItem item = new OrderItem();
 item.name = itemJO.get("name").isString().stringValue();
 item.color = itemJO.get("color").isString().stringValue();
 item.quantity = new
Double(itemJO.get("quantity").isNumber().doubleValue()).intValue();
 al.add(item);
 }
 cellTable.setRowData(al);
 }
});
```

There looks like a lot going on here but in reality, not that much. First a REST call is made to get the task details. The callback from the REST call provides a TaskDetails object. From this we retrieve the JSONObject that represents the variables. We then look for a variable called *order*. We know that *order* is a list variable so we get its content through its *items* property. This is an array of JSONObjects. For each element in the array, we build a Java *OrderItem*. When this has all been done, we set the data in the GWT CellTable.

See also:

- Getting Task and Instance details through REST

### **Starting a process from GWT**

To start a process using the toolkit, we can make a call to `rest.startProcess()`. This takes some parameters as follows:

- `processTemplateId` – The id of the BPD template to be started
- `processAppId` – The id of the process app containing the template
- `snapShotId` – The snapshot id of the process app

- parameters (optional) – Optional parameters
- callback (optional) - Optional callback

Imagine we want to invoke an instance of "MyBPD" but that is all we know about it. First we would have to get a list of all exposed processes. From there, we could search for the exposed process with a display name of "MyBPD". Given that, we have all we need to make the startProcess() call.

For example:

```
rest.exposedProcesses(new ExposedDetailsCallback() {
 @Override
 public void onExposedDetails(ExposedDetails[] exposedDetails) {
 // TODO Auto-generated method stub
 for (ExposedDetails ed : exposedDetails) {
 if (ed.getDisplay().equals("MyBPD")) {
 String processTemplateId = ed.getItemId();
 String processAppId = ed.getProcessAppId();
 String snapShotId = ed.getSnapshotId();
 rest.startProcess(processTemplateId, processAppId, snapShotId, null, null);
 }
 }
 }
});
```

### ***Showing a Coach within GWT***

If we have a BPM task and we wish to show the BPM Coach associated with that task within a GWT environment, we can leverage the GWT Frame component. This injects an HTML <IFRAME> into the page and allows us to set the URL that should be display.

For example,

```
rest.getClientSettings(taskId, new ClientSettingsDetailsCallback() {
 public void onClientSettingsDetails(ClientSettingsDetails clientSettingsDetails) {
 frame.setUrl(clientSettingsDetails.getURL());
 }
});
```

---

# IBM Business Monitor

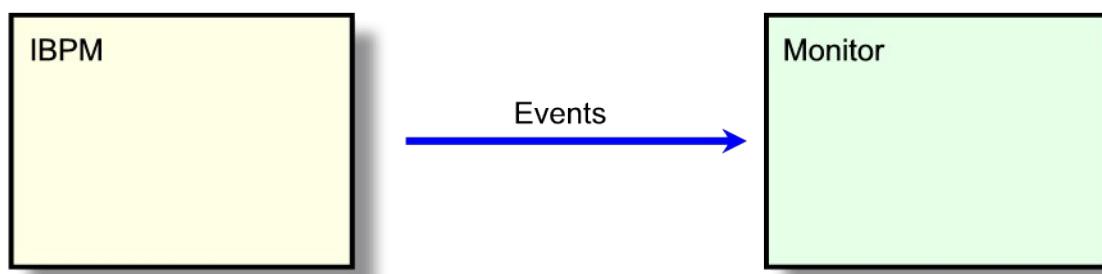
IBM BPM solutions are intended to run the process oriented aspects of your business. An example process might be that of handling sales orders. When an order arrives, the order is built, the customer is billed and then the product is shipped. All of this may be governed by the execution of the process. Great!! But now consider that hundreds of such orders may be processed a day. After a few months of operation, you may find that you have the following questions:

- What was the average value of an order?
- Which states or provinces ordered the most blue widgets?
- Which sales rep brought in the highest return?
- Which orders take the longest to be processed?
- Is there a pattern in orders received in a week?

In order to be able to begin to answer these kinds of questions, we must have some mechanism of capturing the raw information that we are later going to analyze. IBM BPM provides an out-of-the-box solution to capturing such information and presenting it to the user through the Performance Data Warehouse story. An alternative to the IBM BPM supplied monitoring is to employ and integrate the separate IBM Business Monitor product.

IBM Business Monitor is its own rich and sophisticated environment and, once again, could trivially be the topic of its own book. Here we will try and discuss Business Monitor in context of IBM BPM.

The high level architecture of Business Monitor (Monitor) is that it acts as the target of events sent from external applications. In our story, IBM BPM can be considered such an application. Events are generated from IBM BPM and consumed by Monitor.



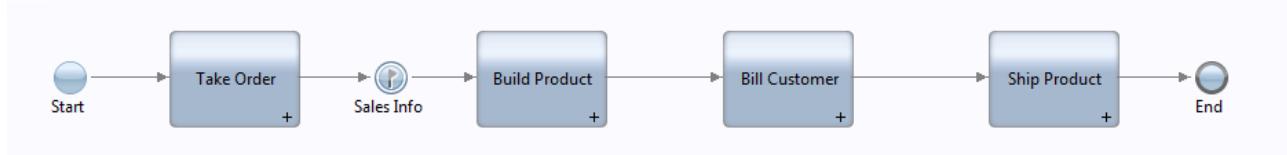
So ... what might be considered an event? The answer is primarily:

- BPD Process information
  - Start of a process
  - End of a process
- BPD Activity information
  - Start of an activity
  - End of an activity
- Tracking events
  - Message Intermediate events originating from tracking points

These low level events seem quite technical in nature and not immediately that useful to business ...

for example, BPD activity XYZ started or Tracking Point ABC was invoked. The nature of the event by itself isn't that interesting but if we place it in context it suddenly has meaning.

Let us look at this simple process as an example:



A customer submits an order, we build the product to specification, we charge the customer and finally ship the product. If we now say that the start and end of a BPD step generates events, we immediately can start examining the operation of our processes:

- How many orders arrive over a period of time?
- How long on average does it take from receiving the order until till we bill the customer?
- What is the average order price?
- To which US states are we shipping the majority of our products?

Notice that these are *Buisness Level* questions which can be answered by overlaying what our process *means* with the data that is produced at different steps in our process.

To be able to answer these kinds of question, we have to model the notion of a single instance of a process and then when multiple processes have completed, we can aggregate such individual instances together. The information that we wish to collect on a single instance of a process are generically called *metrics*. Metrics are the meaningful data items associated with a process instance. In our sample process, these are potential metrics we could define as wanting to collect:

- `orderValue` – The value of a single order.
- `shipState` – Which state is the product being shipped to.
- `orderToBillingDuration` – The amount of time that order took from order taking to billing.
- `orderDate` – The date when the order was received.

When IBM BPM generates events that are sent to Monitor, these events are *raw* events. There is nothing in them that provides these nice and clean metrics. The raw events are almost purely technical in nature. An example of a raw event might be (not real data)

```

Event:
{
 processId: 1234,
 step: TakeOrder,
 state: completed,
 dateTime: 2011-07-07T17:34Z
}

```

which might signify that for a specific process instance identified as 1234, the step in the process that the developer called TakeOrder has completed at a given time. Trying to build sense out of raw events like this would be impossible (or at least extremely difficult). It is here that Business Monitor comes into play.

Business Monitor has the notion of a *Model*. A model is a description of the logical metrics we wish to collect (eg. the value of an order) and the *rules* used to populate or construct that metric

from the raw events arriving from the external systems. The Monitor Model is the bridge between raw events and the business friendly metrics. Monitor Models can be constructed in two ways. They can either be generated by the tooling or they can be built using the Monitor Model editors supplied with IID. A hybrid approach is also possible where generated models can then be modified by the Monitor Model editors. Personally I feel that to be effective, one should understand how to build a Monitor Model from scratch. Not only does this grow confidence and understanding in the product but if and when generated models need to be modified, there is significantly less mystery involved.

## Installation of Business Monitor

At the time of writing (2013-12), the latest version of Business Monitor is 8.0.1.

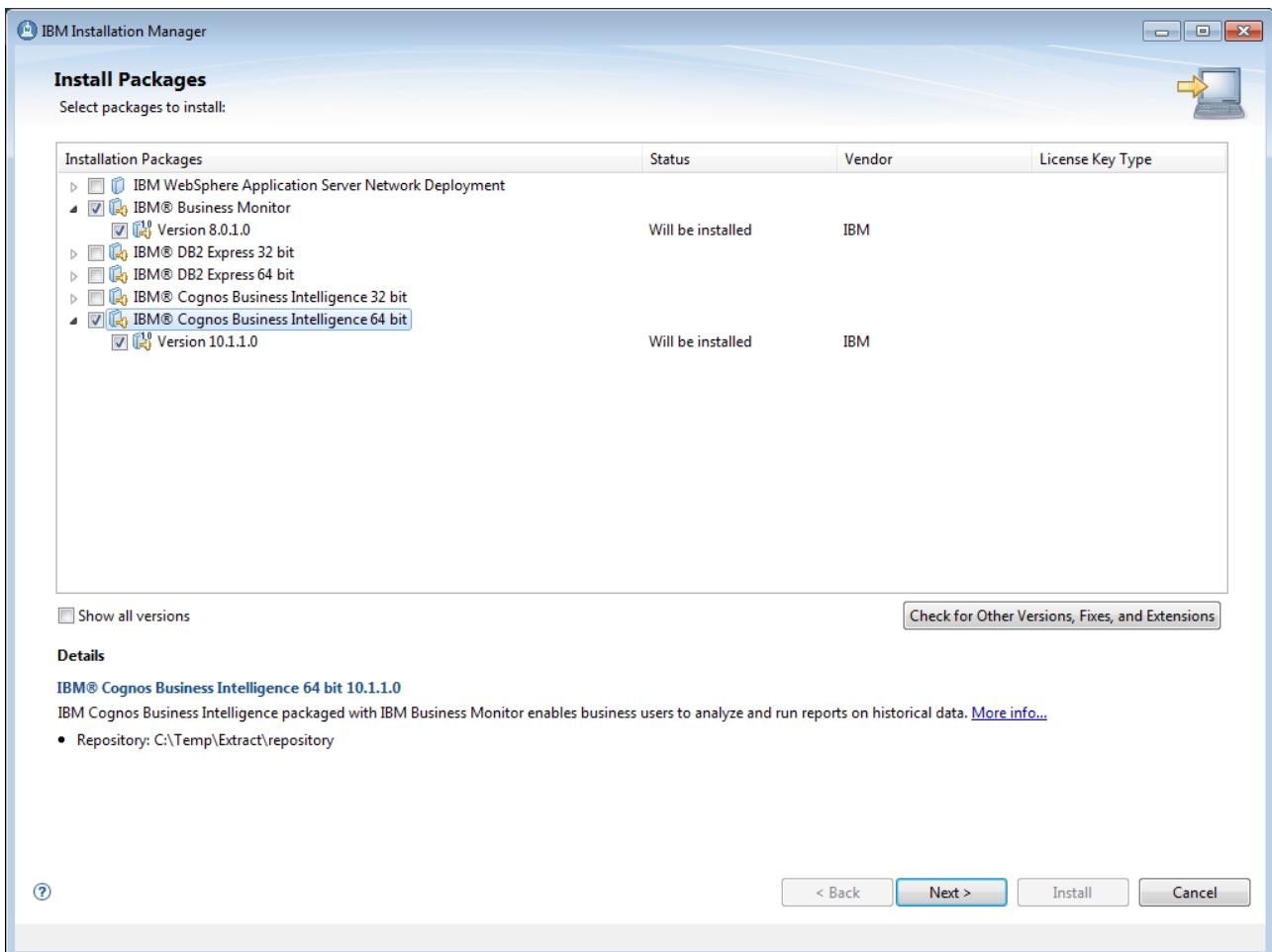
The part numbers are:

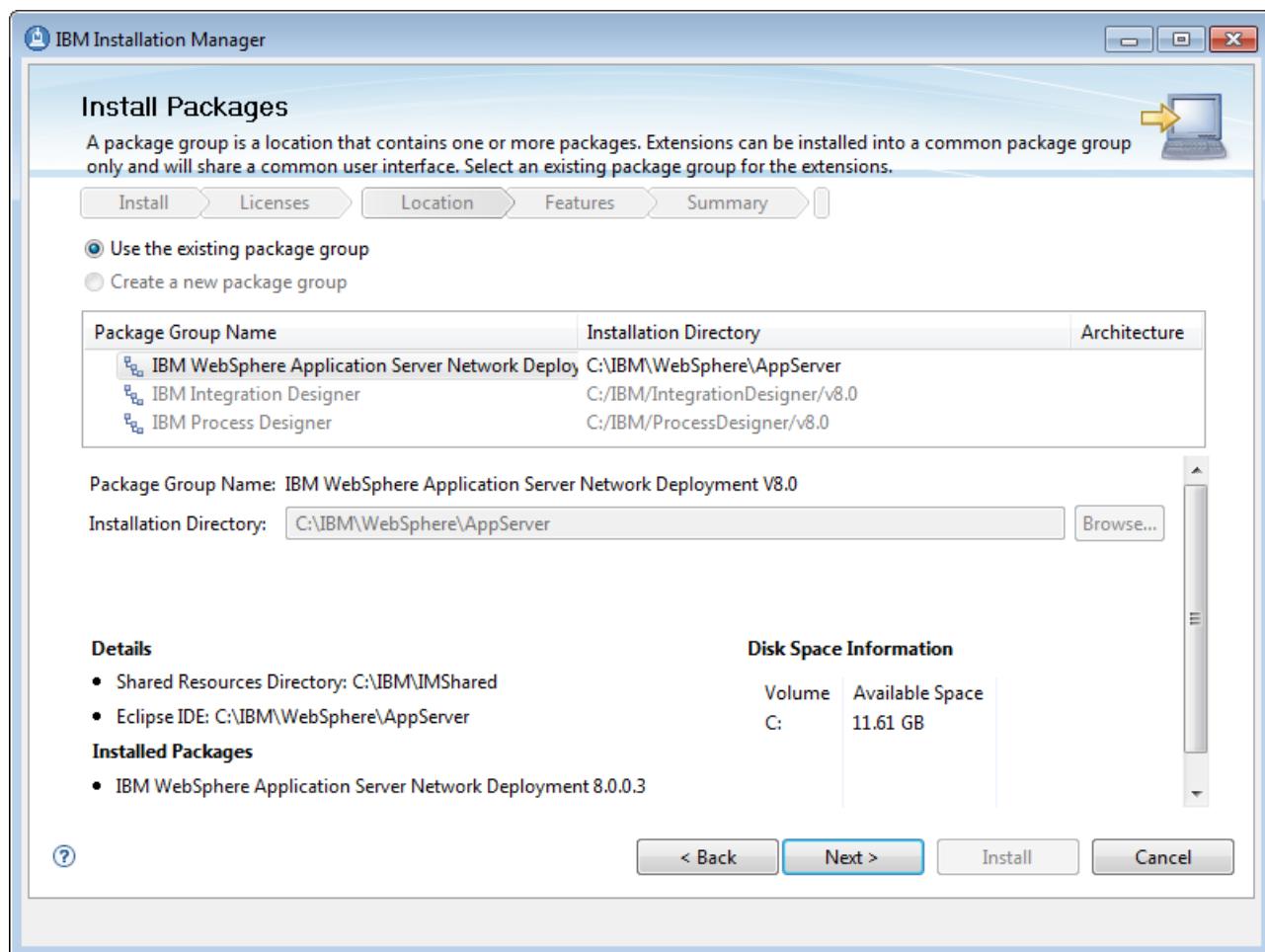
Name	Part Number
IBM Business Monitor v8.0.1 for Windows 1 of 4	CIC9YML
IBM Business Monitor v8.0.1 for Windows 2 of 4	CIC9ZML
IBM Business Monitor v8.0.1 for Windows 3 of 4	CICA0ML
IBM Business Monitor v8.0.1 for Windows 4 of 4	CICA1ML

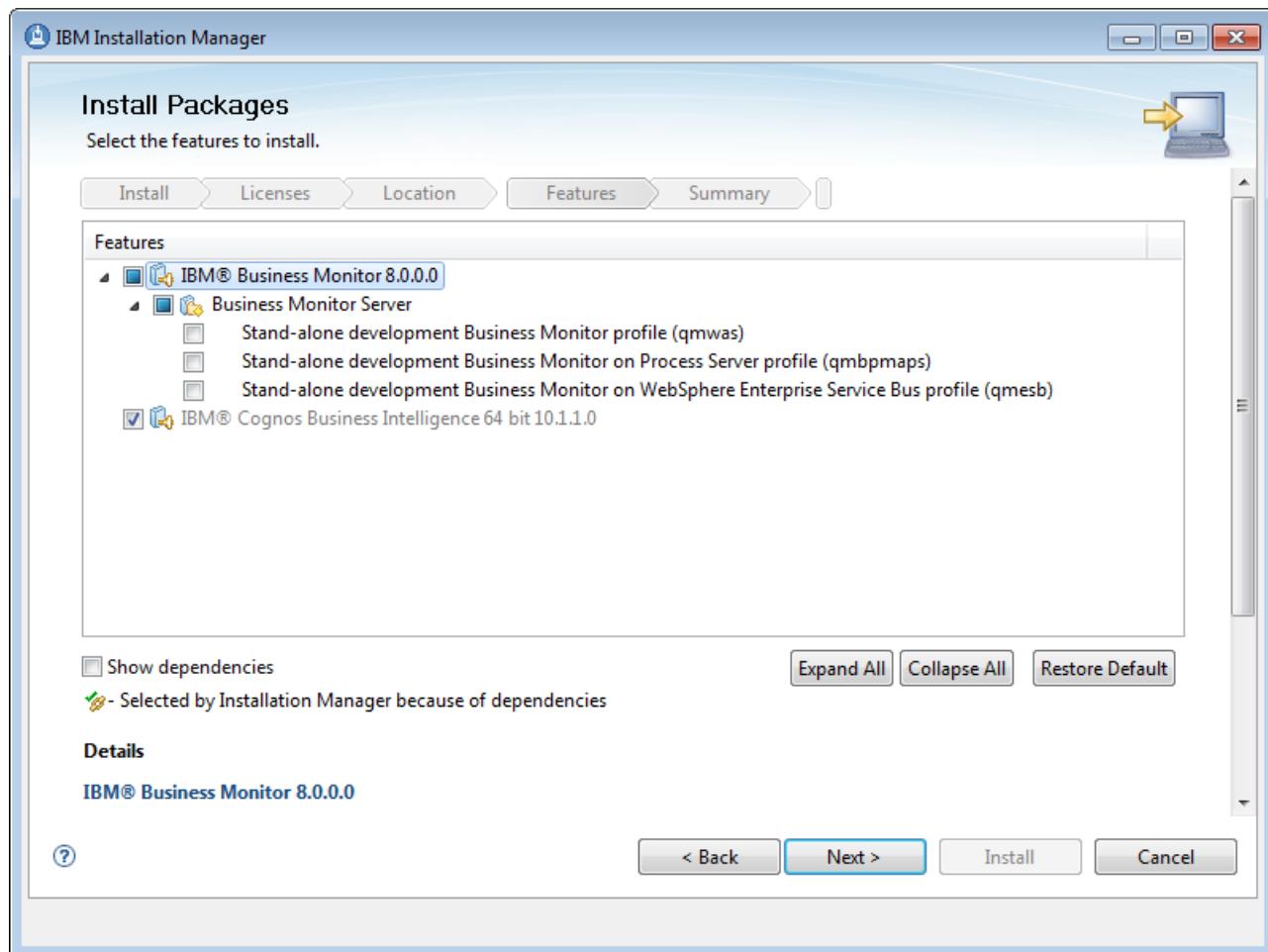
The prerequisites for IBM Business Monitor can be found here:

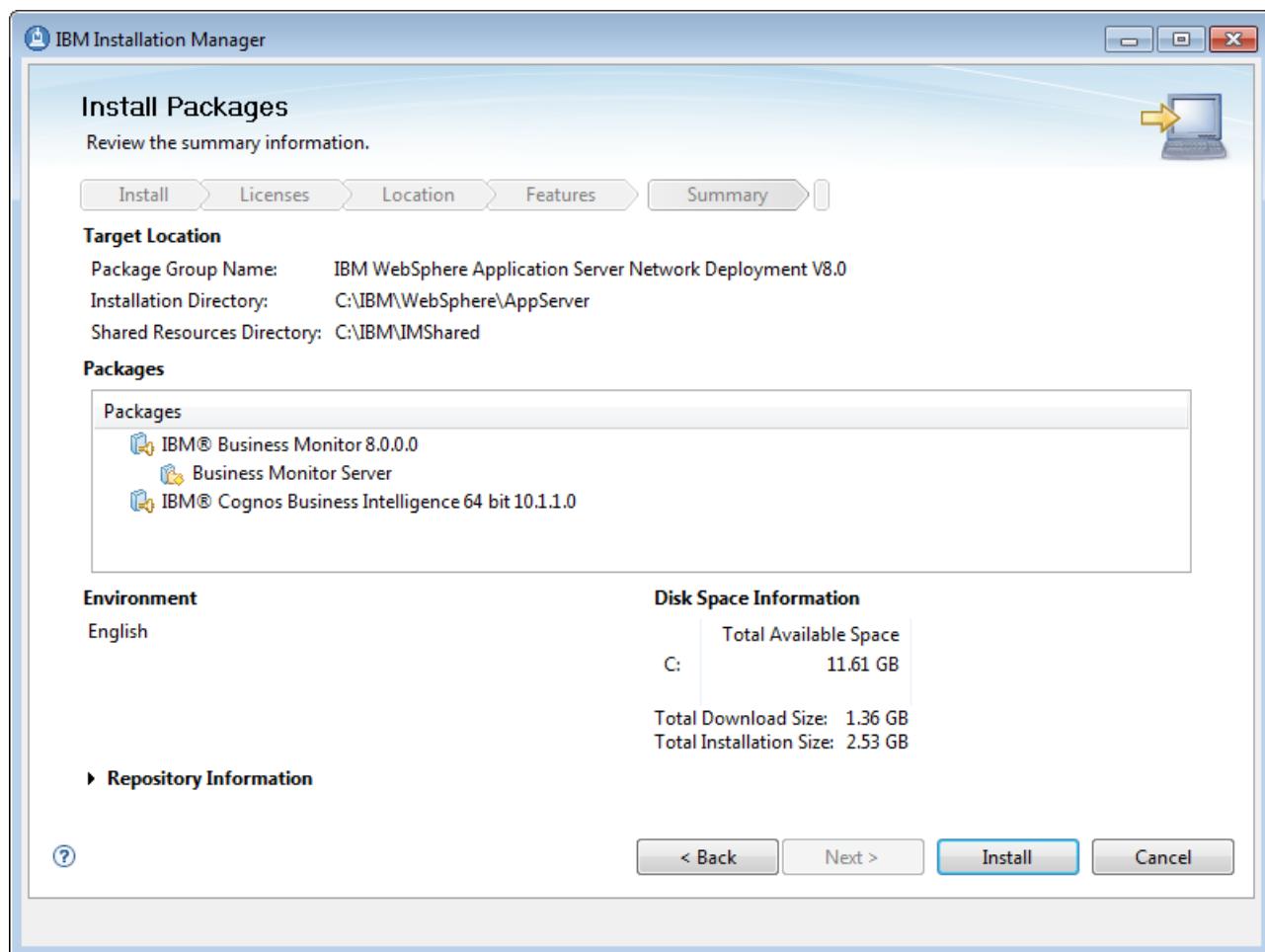
<http://www-01.ibm.com/support/docview.wss?uid=swg27008414>

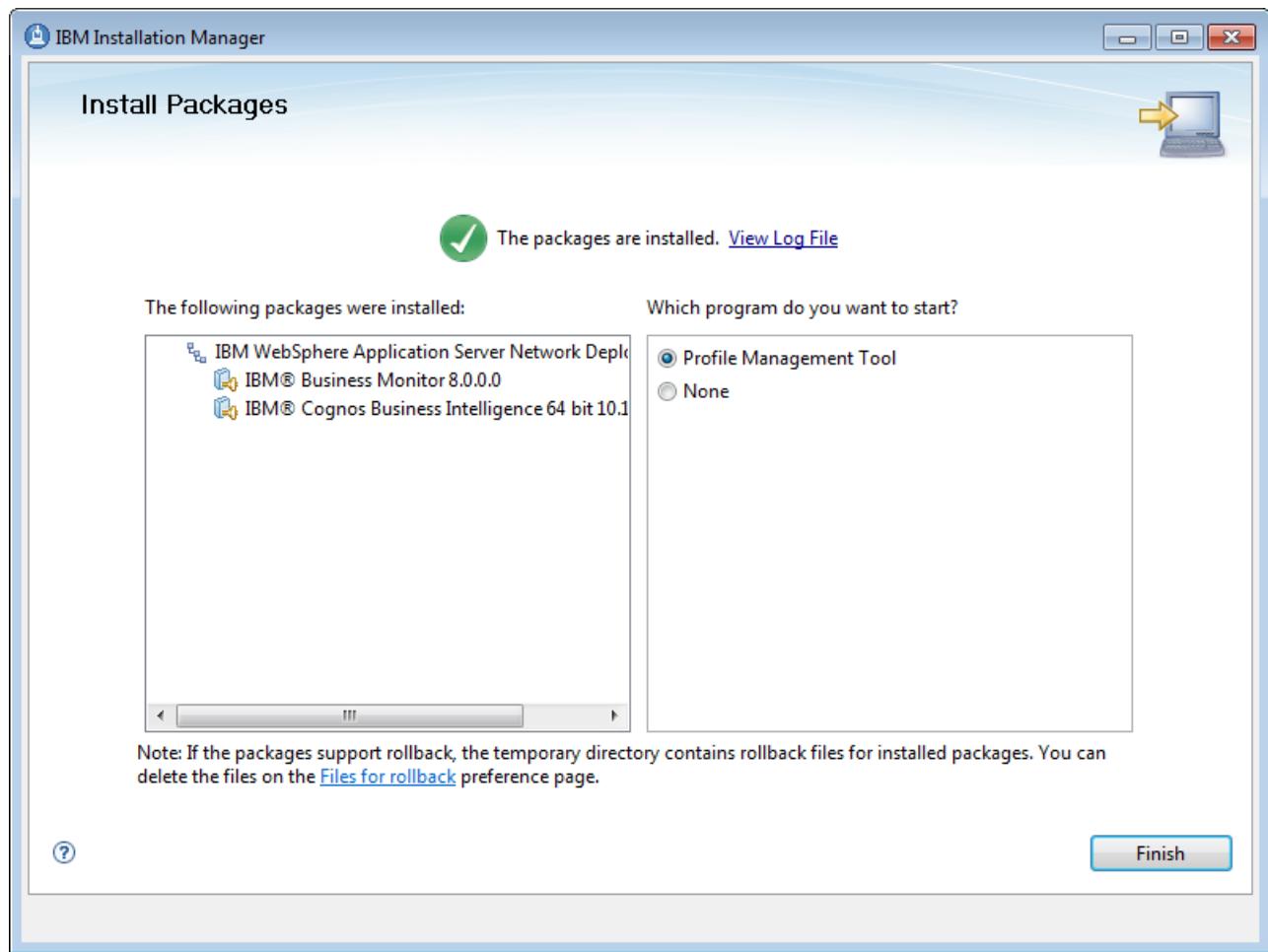
The IBM IID product also includes a copy of IBM Business Monitor.





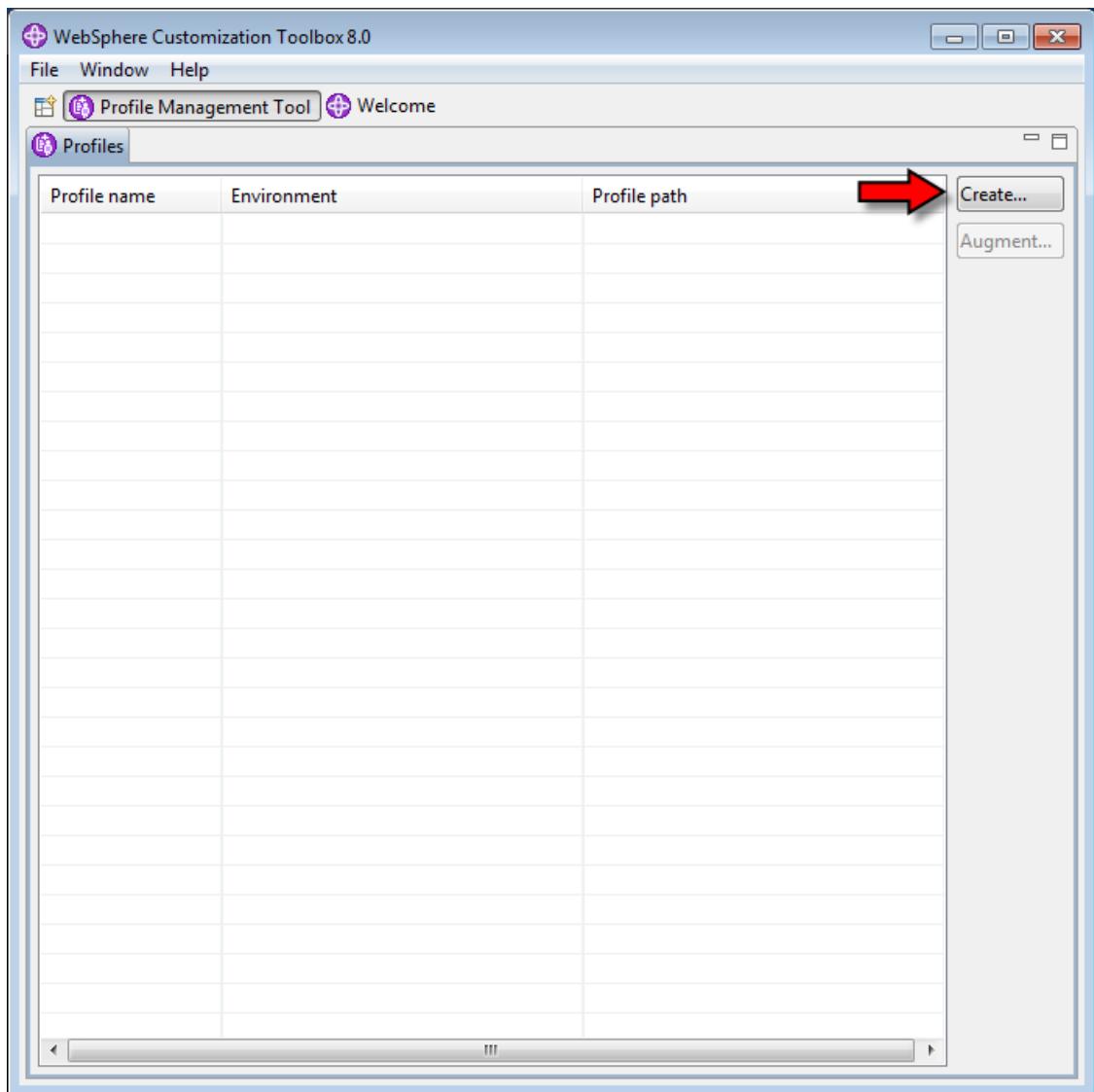


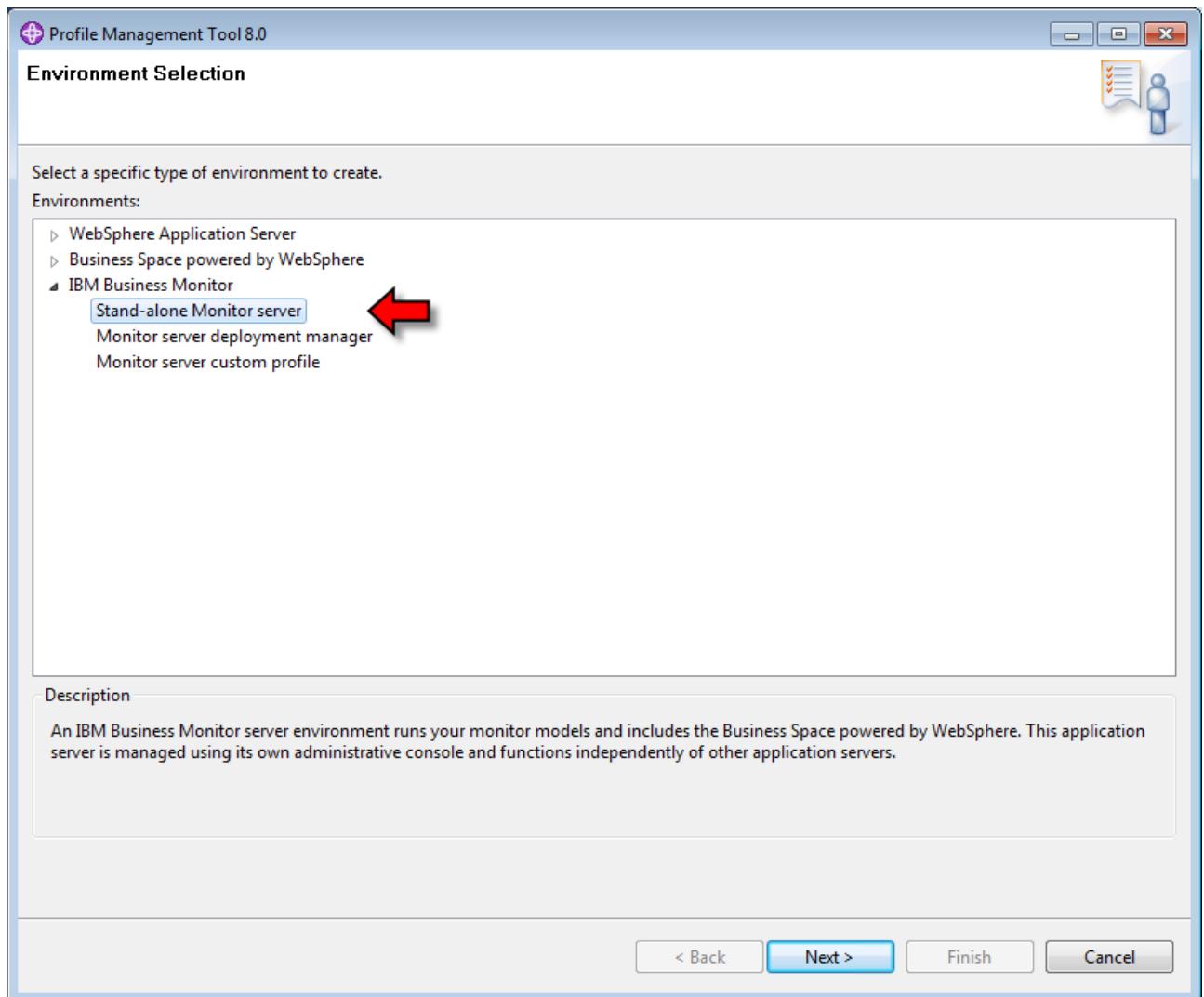


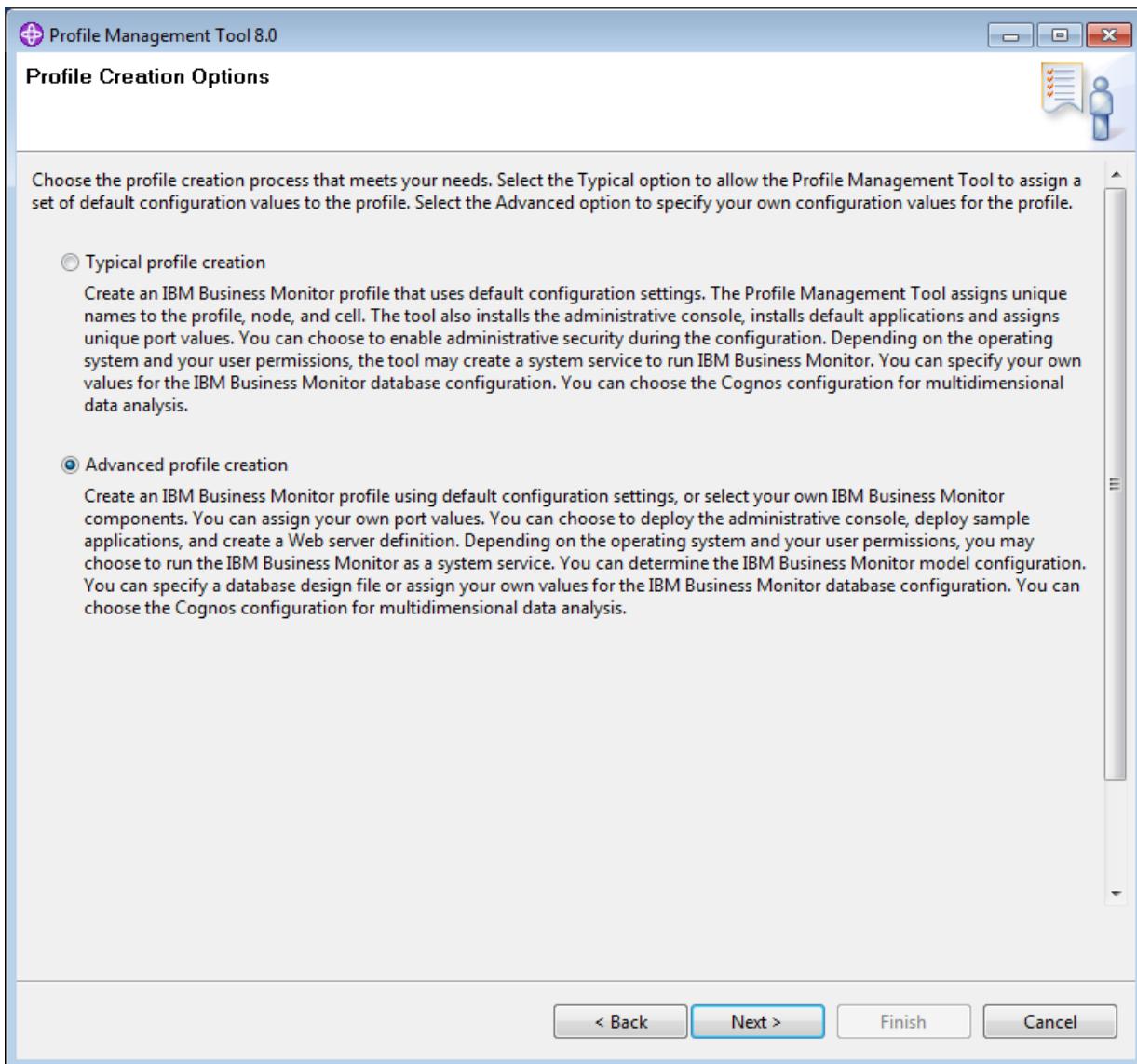


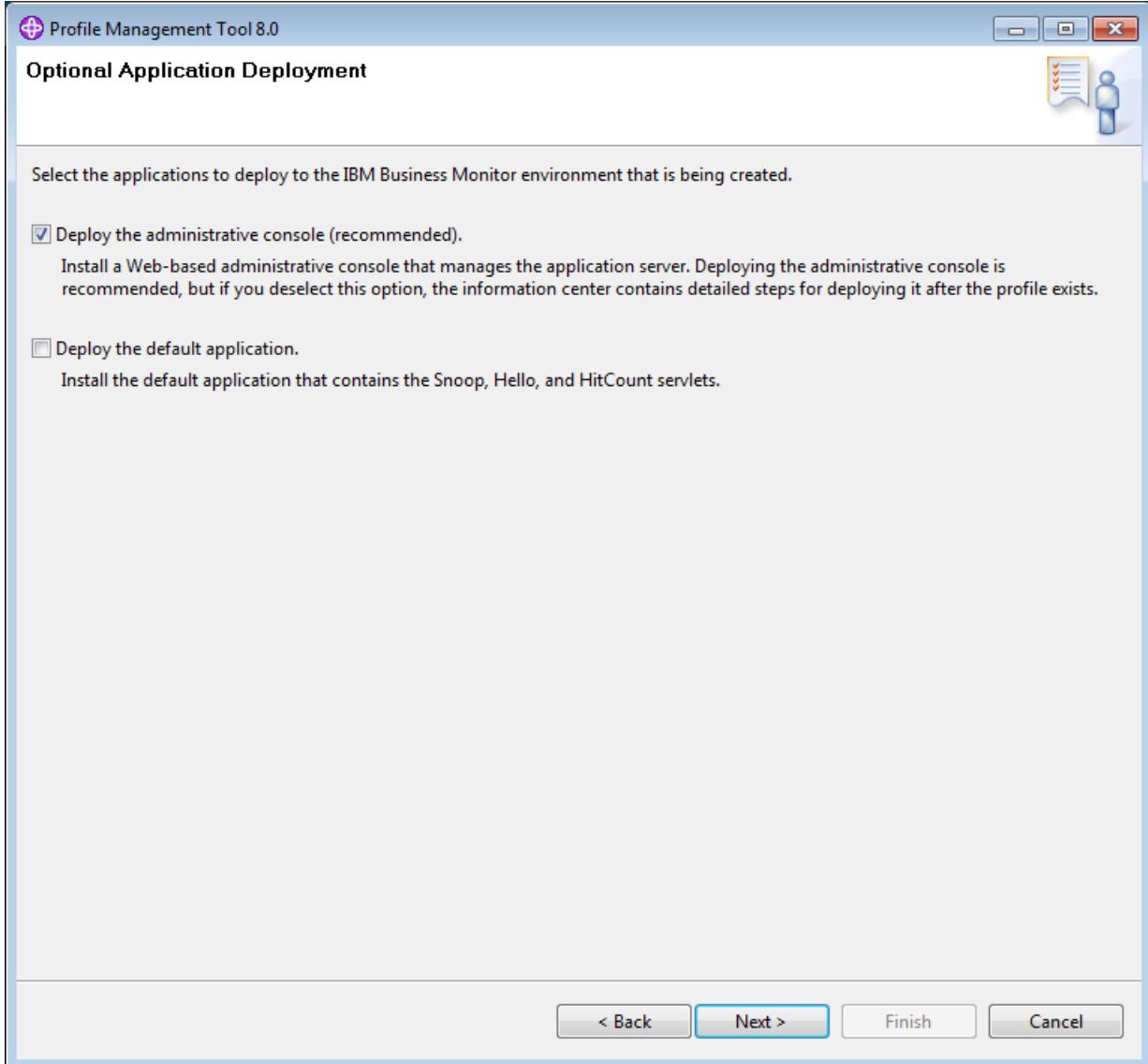
## *Creating a profile*

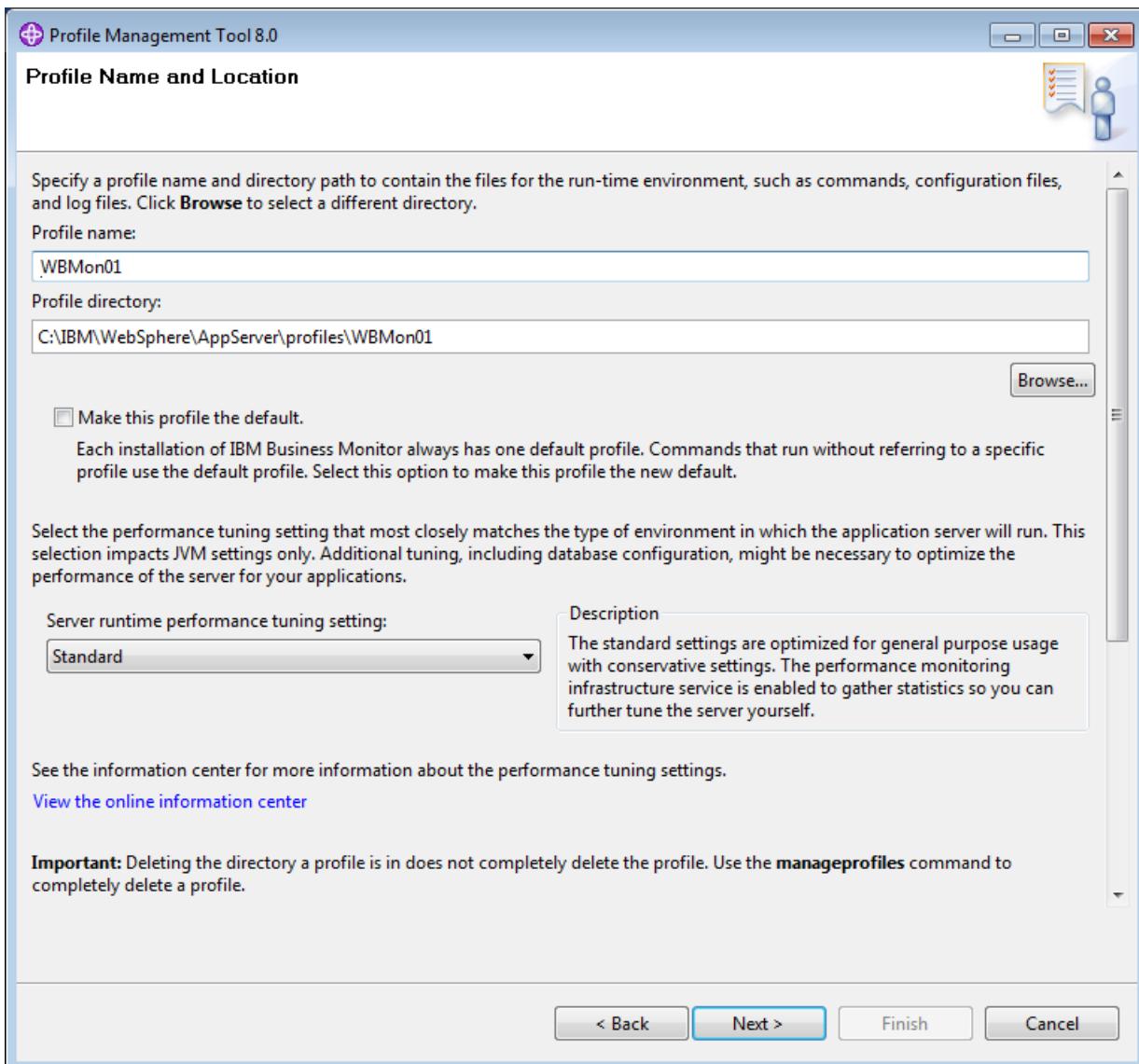
A new profile for a Business Monitor server can be created with the WebSphere PMT tool. This can be found in the Start Menu under IBM > Business Monitor 8.0 > Profile Management Tool.

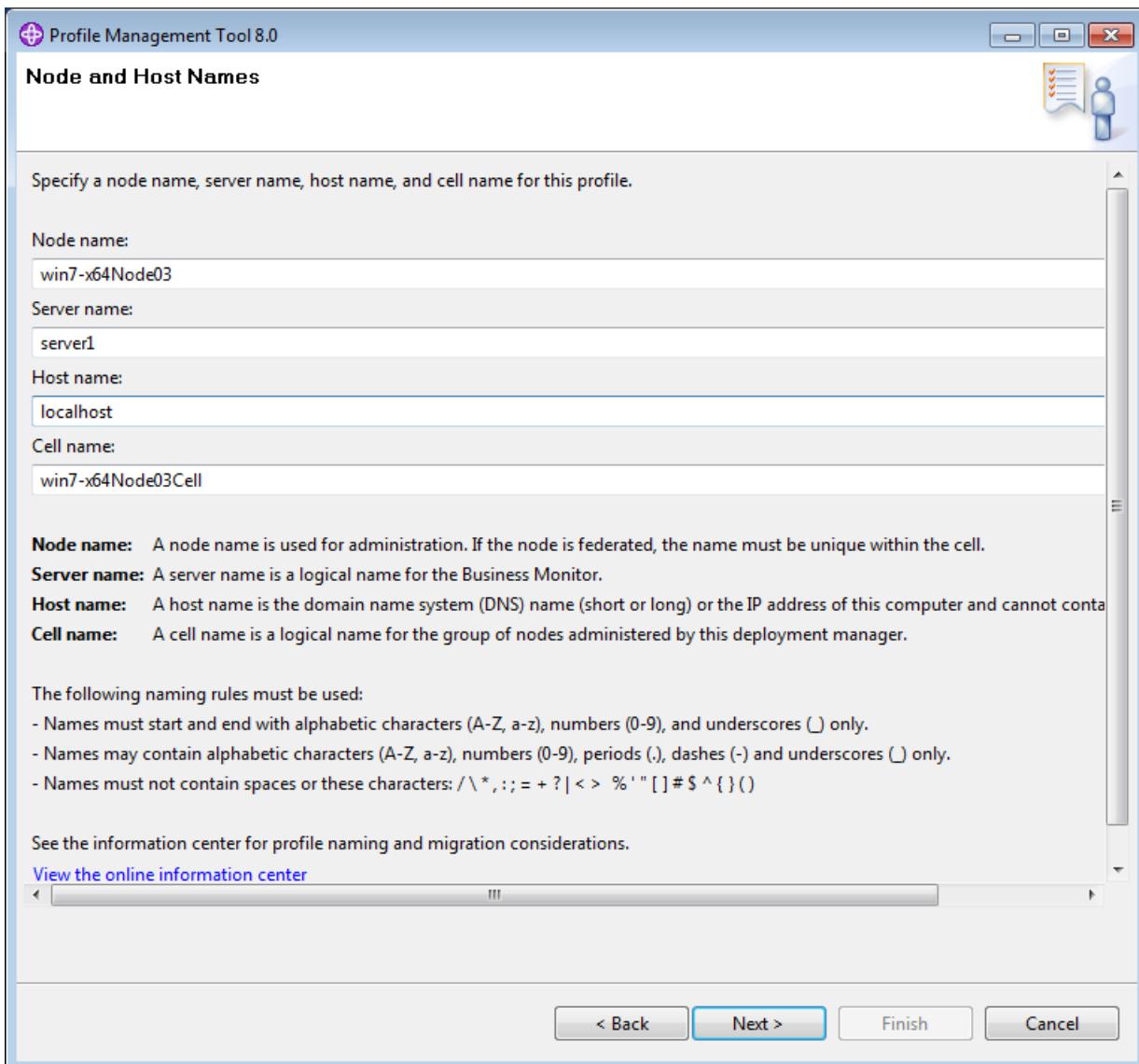


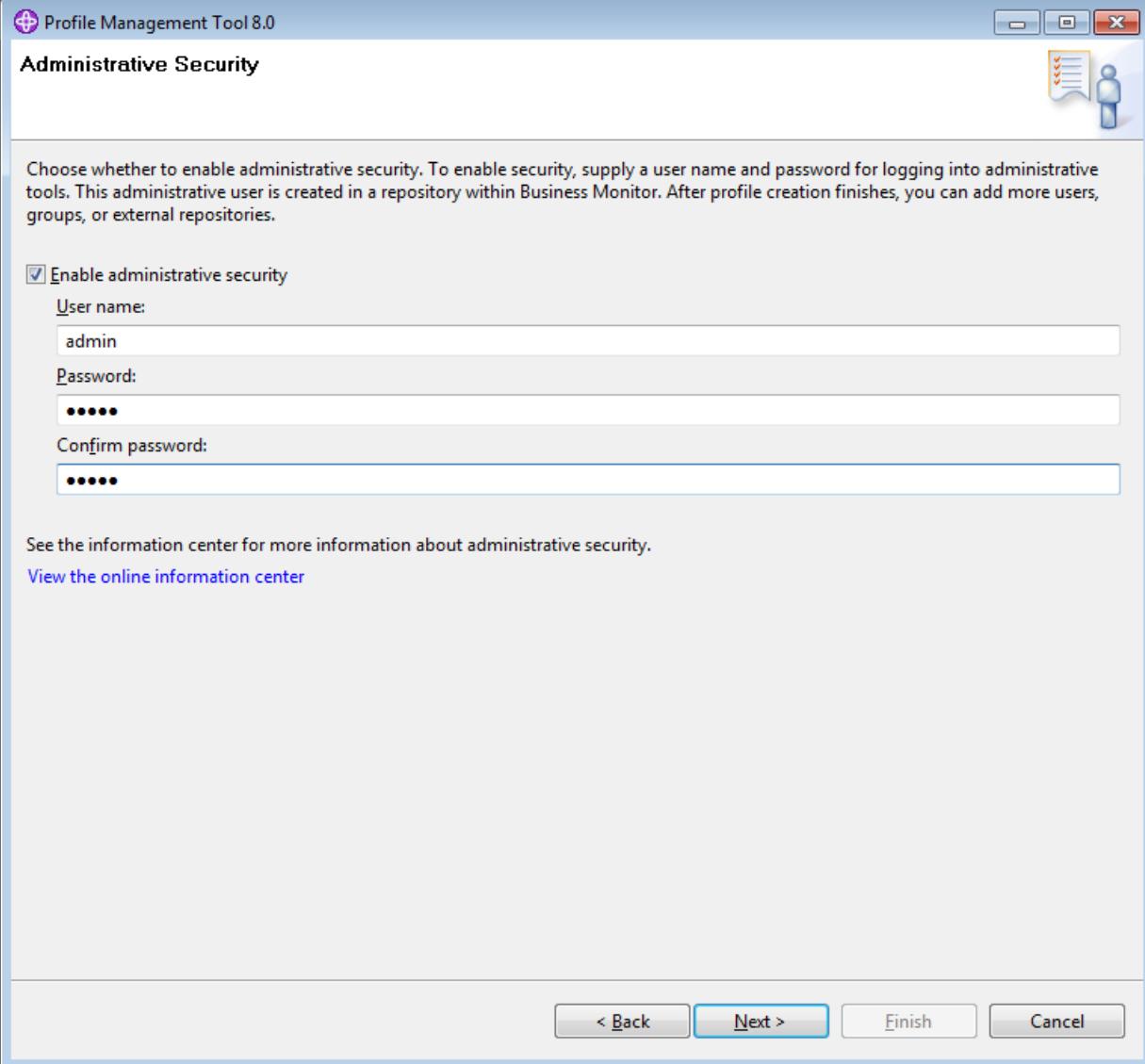


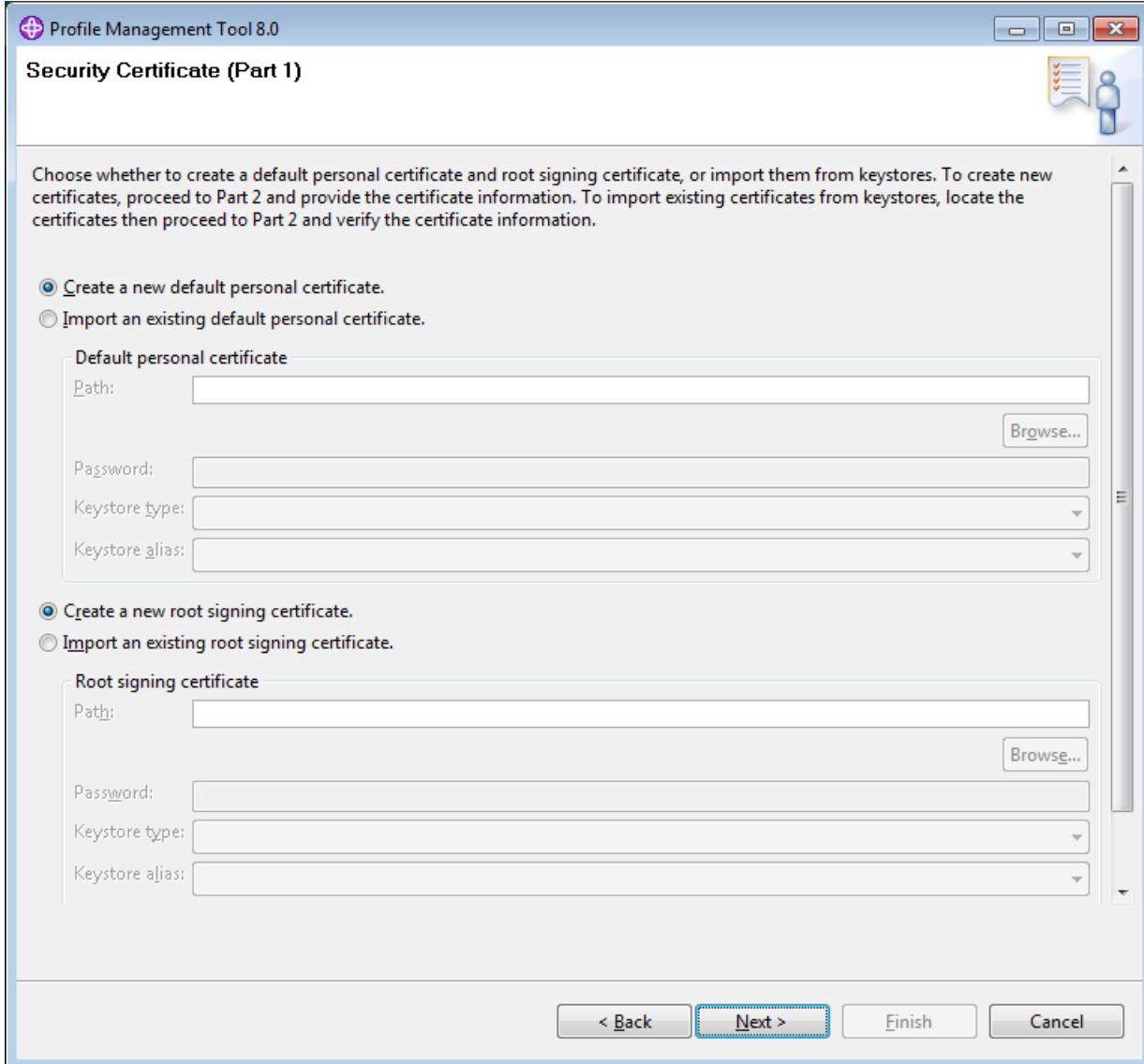


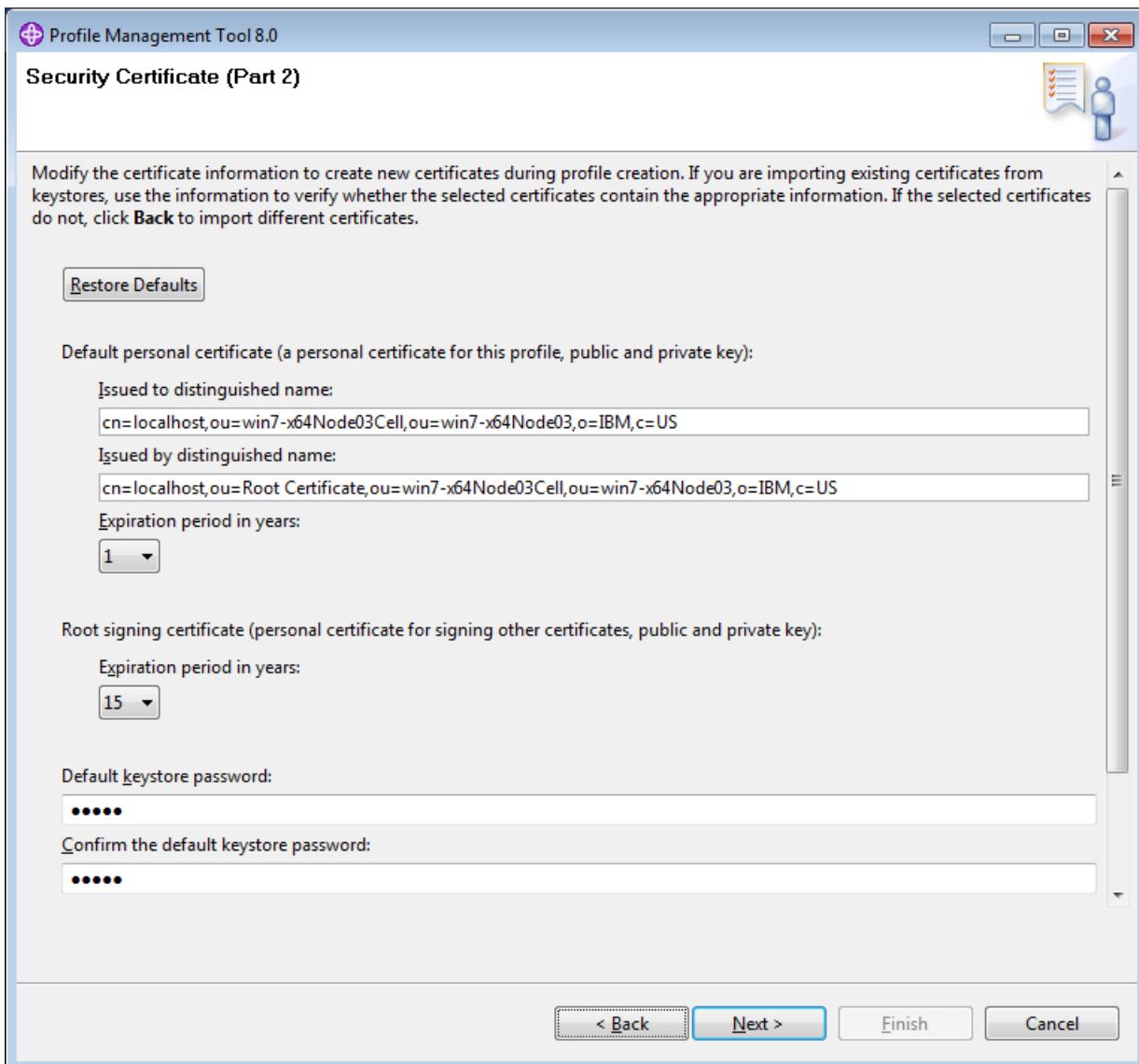


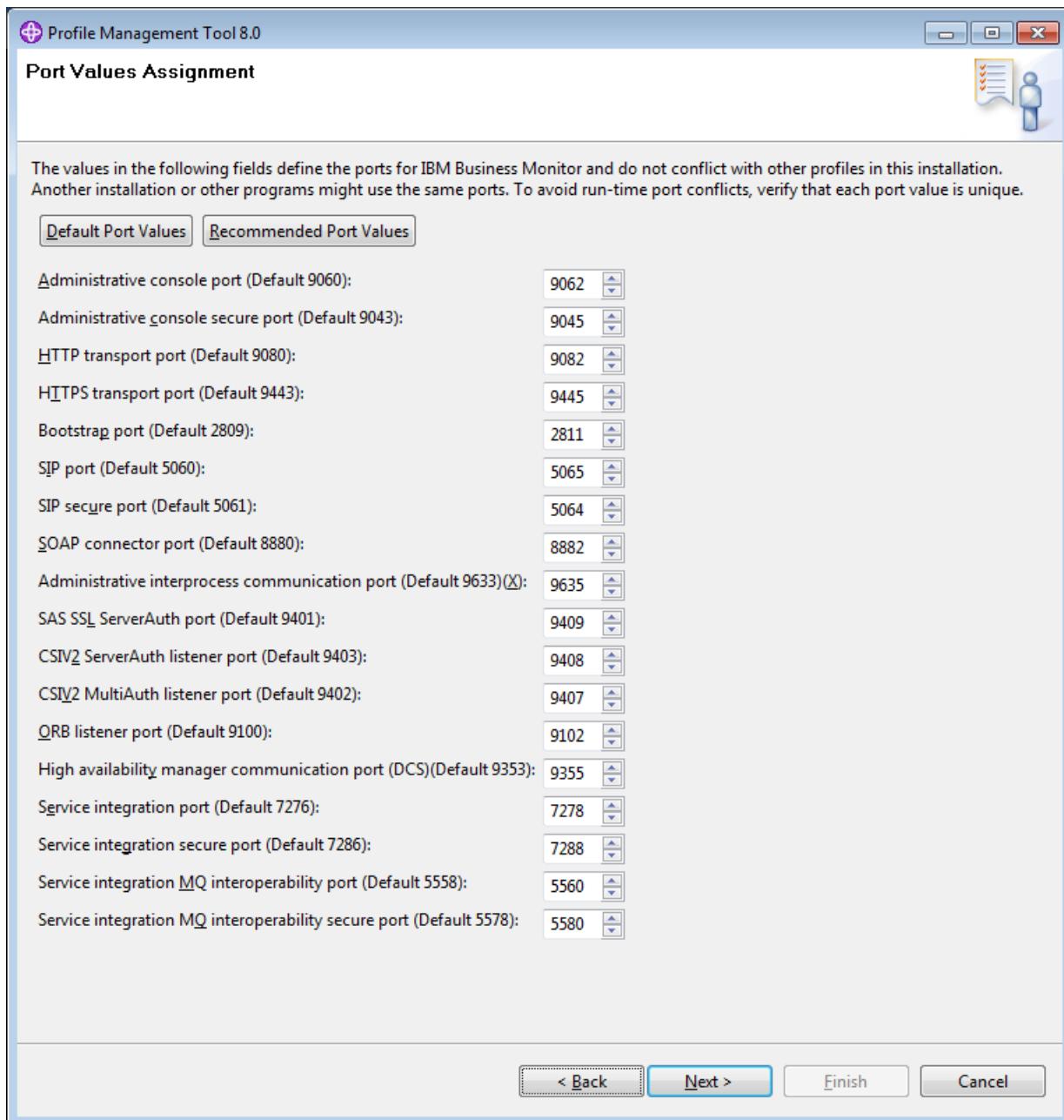












When running on the same machine as IBM BPM, make sure that you choose distinct port numbers. Adding "5" to the basics might be a good choice:

Description	Base port + 5
Administrative console port	9065
Administrative Console Secure Port	9048
HTTP Transport port	9085
HTTPS Transport port	9448
Bootstrap Port	2814
SIP Port	5070
SIP Secure Port	5071
SOAP Connector Port	8885
Admin IPC Port	9638

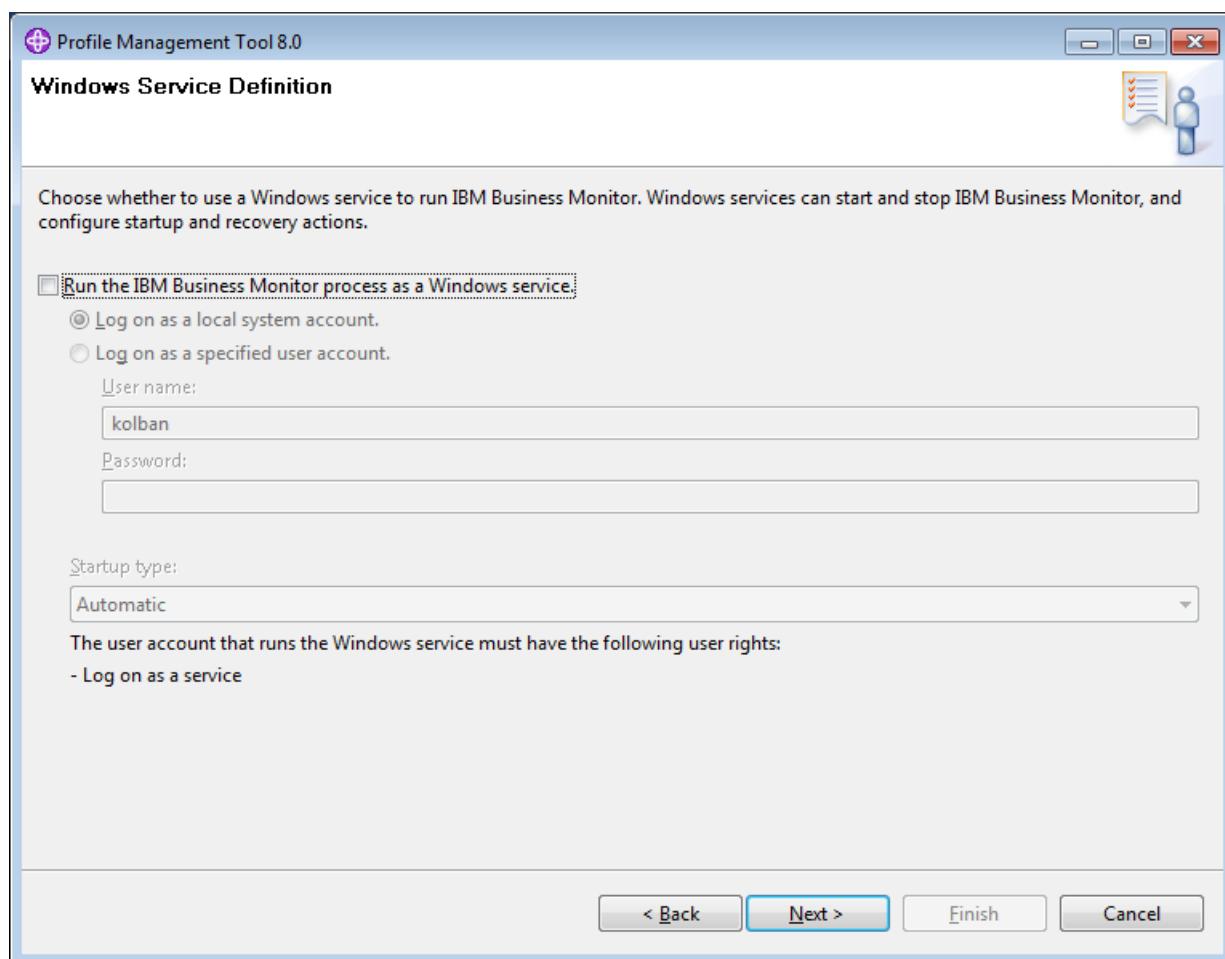
SAS SSL Server Auth Port	9416
CSIV2 ServerAuth Listener Port	9418
CSIV2 MultiAuth Listener Port	9417
ORB Listener Port	9105
HA Communication Port	9358
Service Integration Port	7281
Service Integration Port Secure	7291
Service Integration MQ interoperability port	5563
Service Integration MQ interoperability secure port	5583

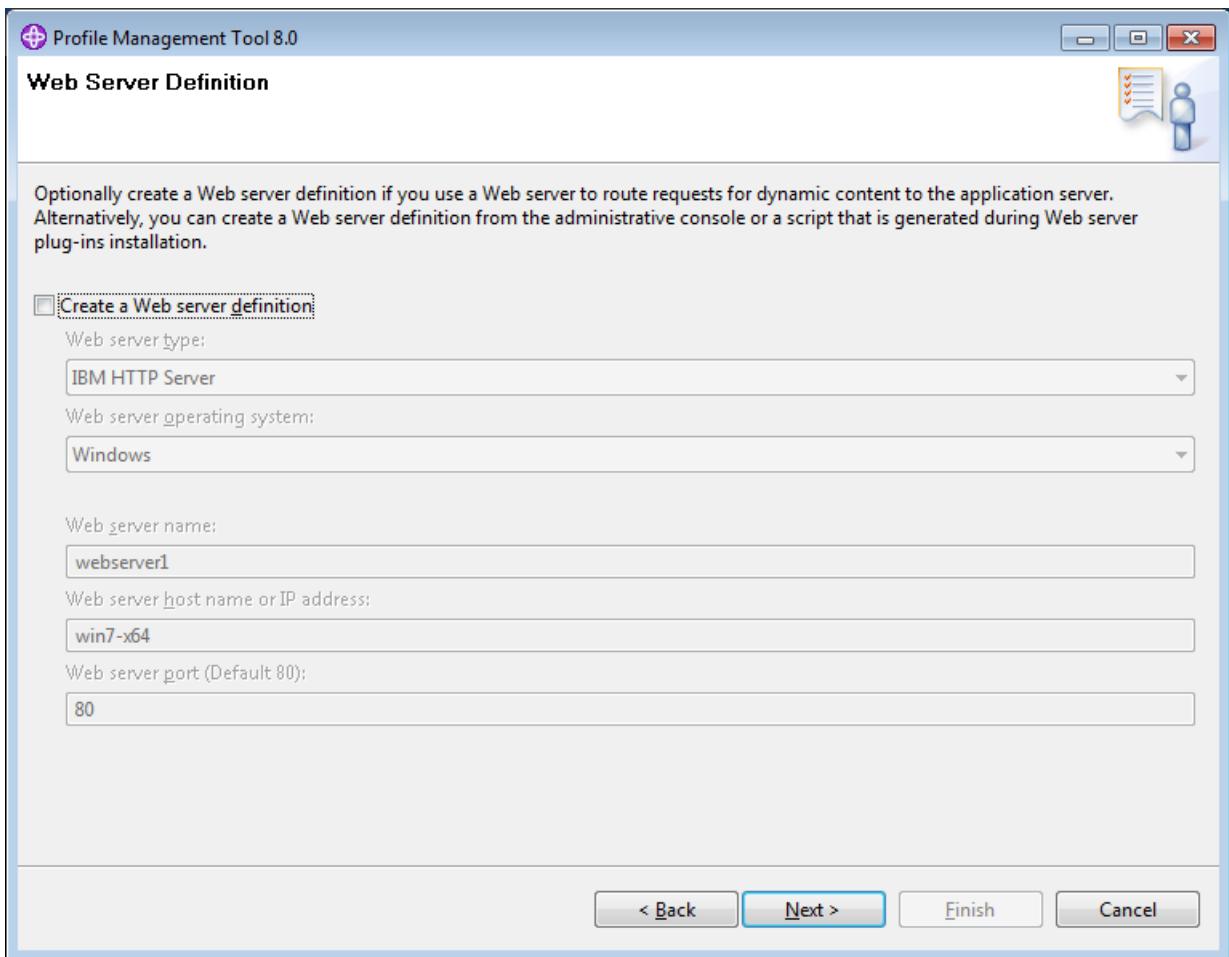
These settings will result in Business Space being available at:

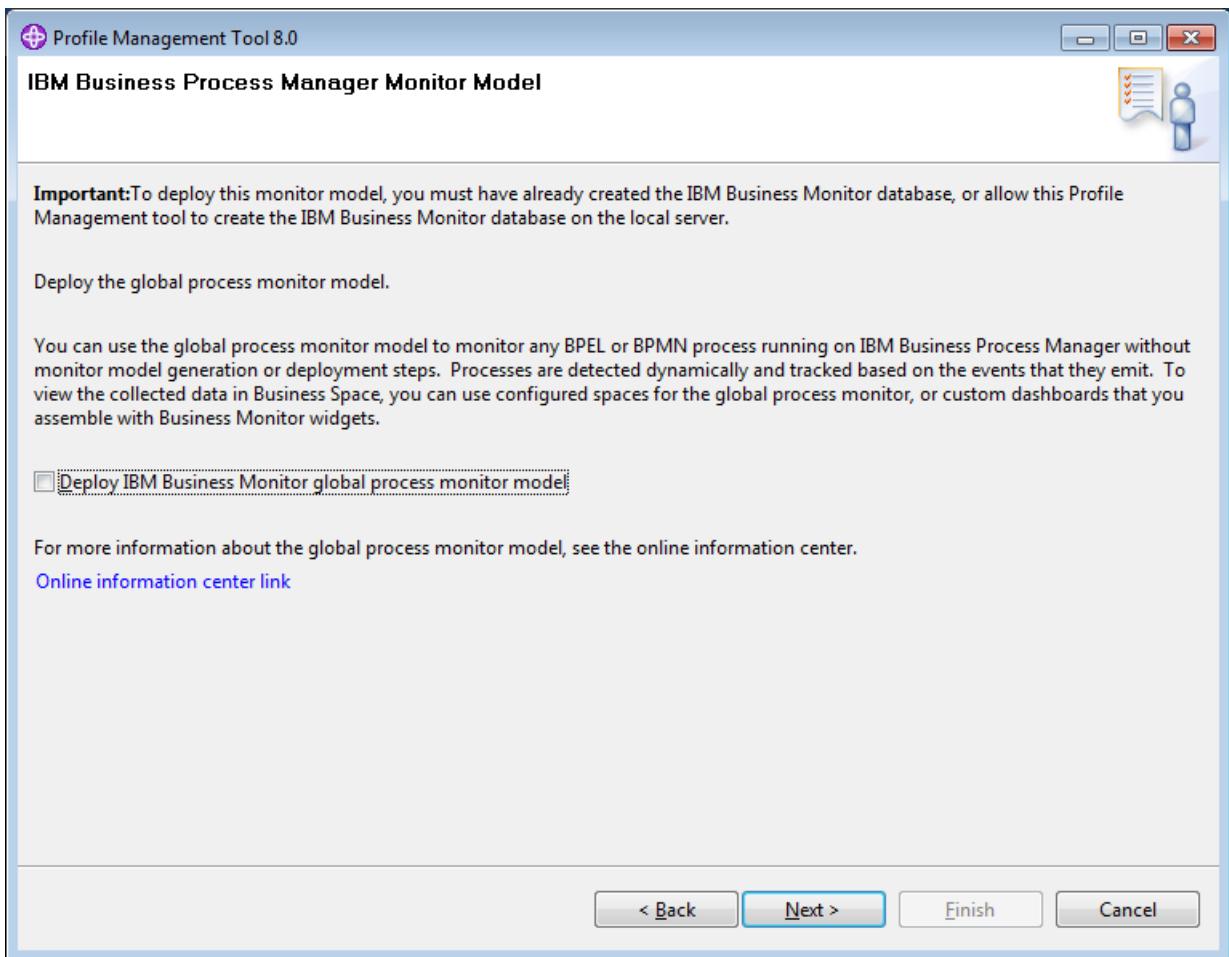
<http://localhost:9085/BusinessSpace>

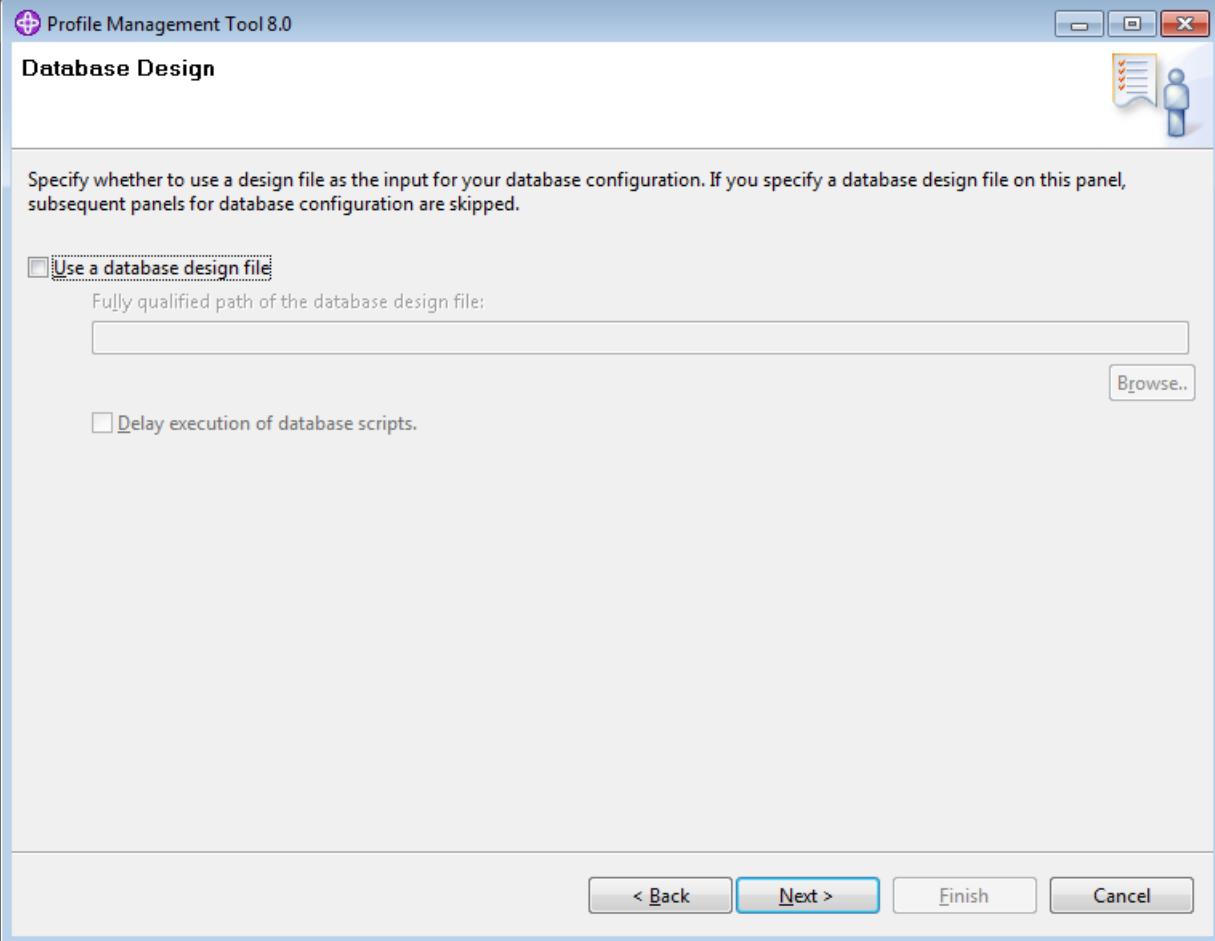
and WAS Admin Console at:

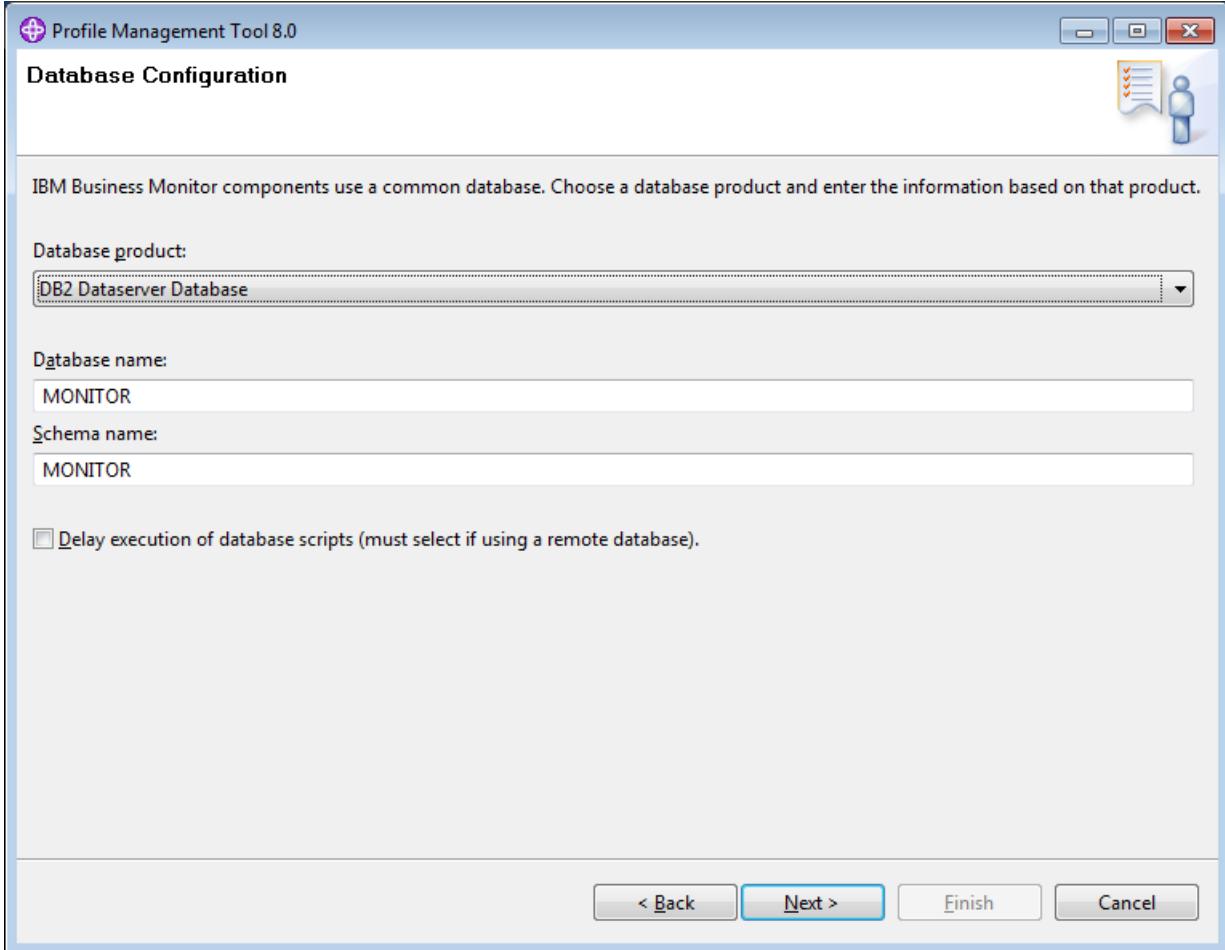
<https://localhost:9048/ibm/console/logon.jsp>











Profile Management Tool 8.0

## Database Configuration (Part 2)

Additional information about the database server you are using is required to complete configuration for the DB2 Dataserver Database database. For database authentication, you must type the user name and password that will be used to connect to the database. The database user must have read and write access on the database.

User name:

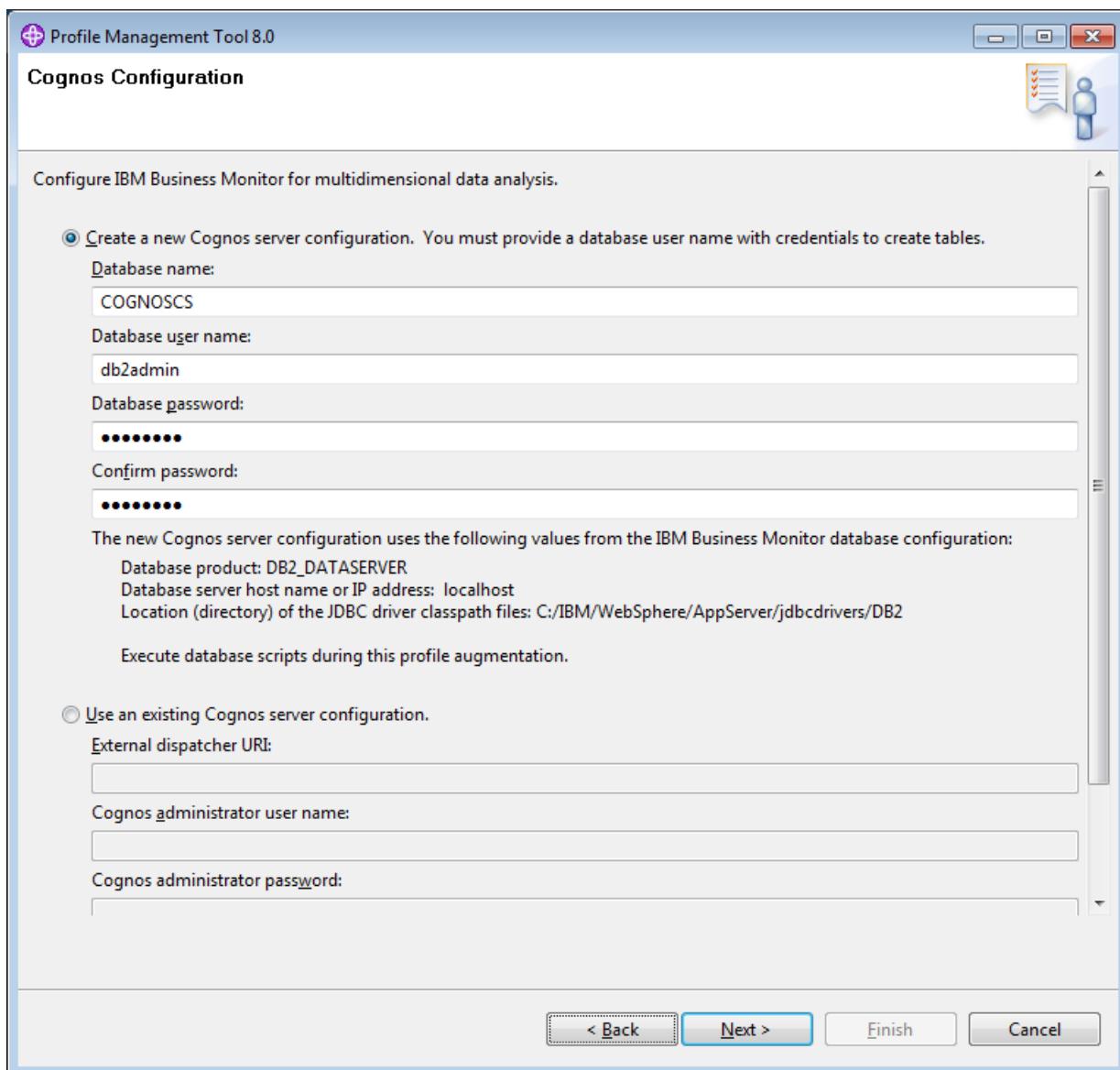
Password:

Confirm password:

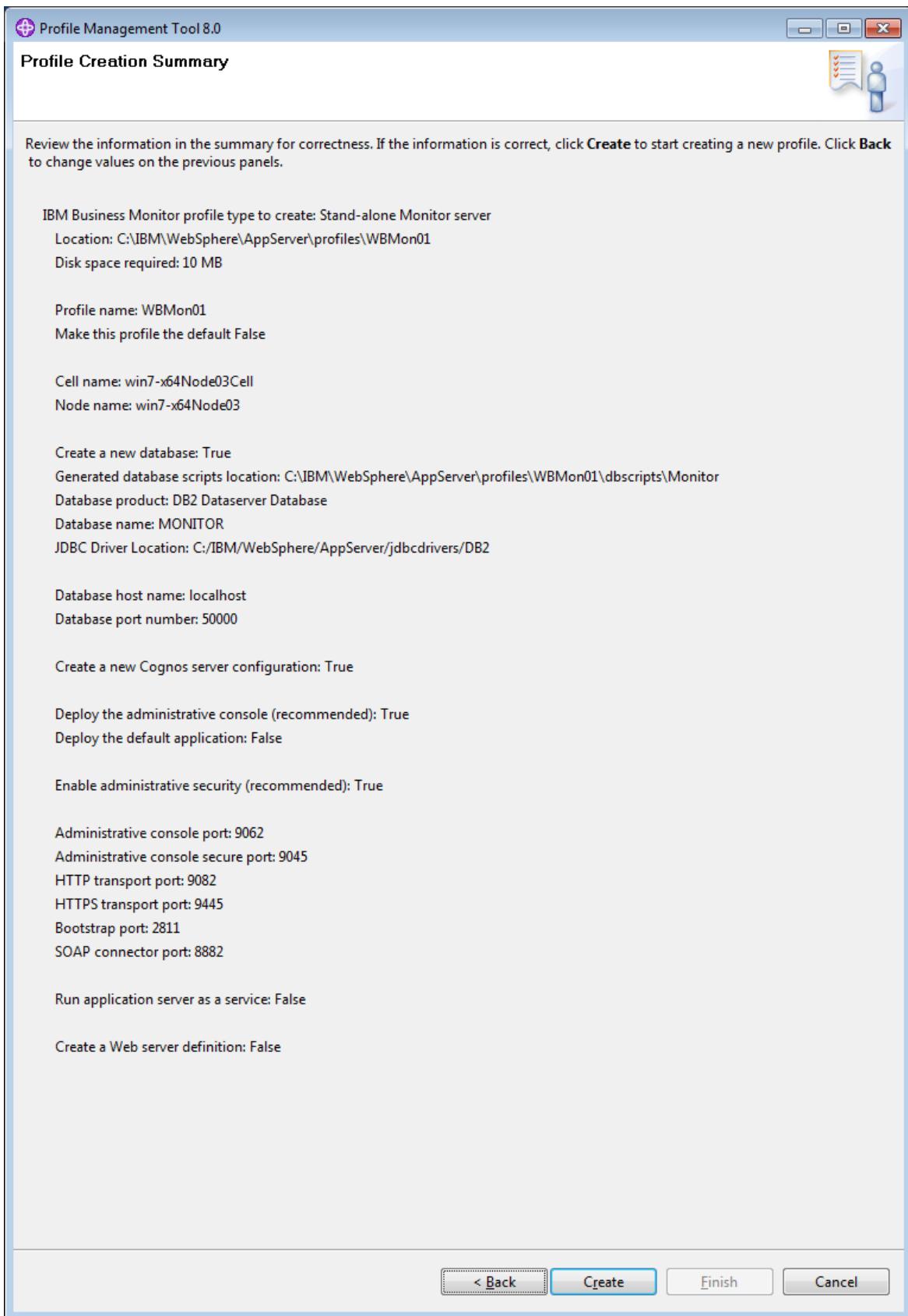
Location (directory) of JDBC driver classpath files:

Database server host name or IP address:

Database TCP/IP service port or listener port:



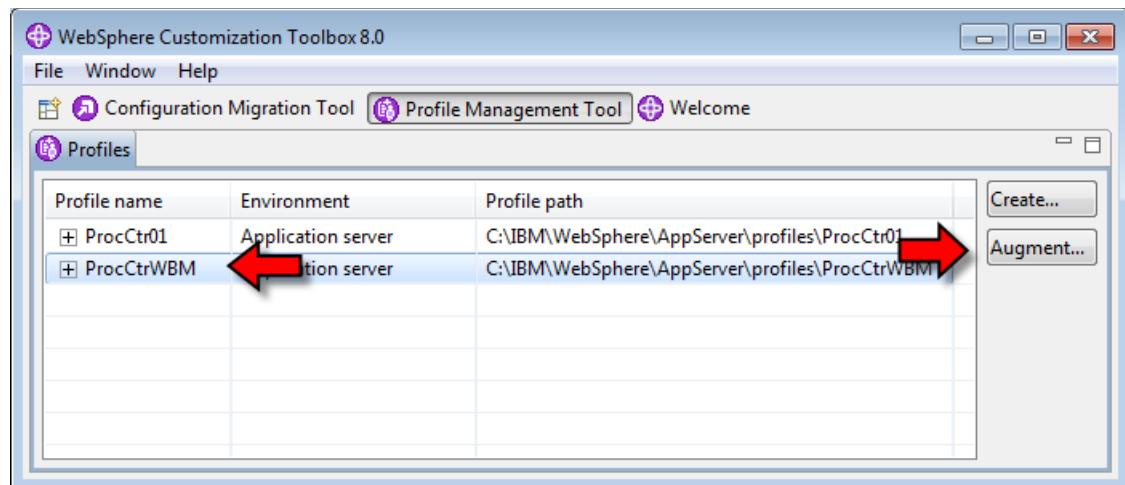
Finally we get to run the configuration command.



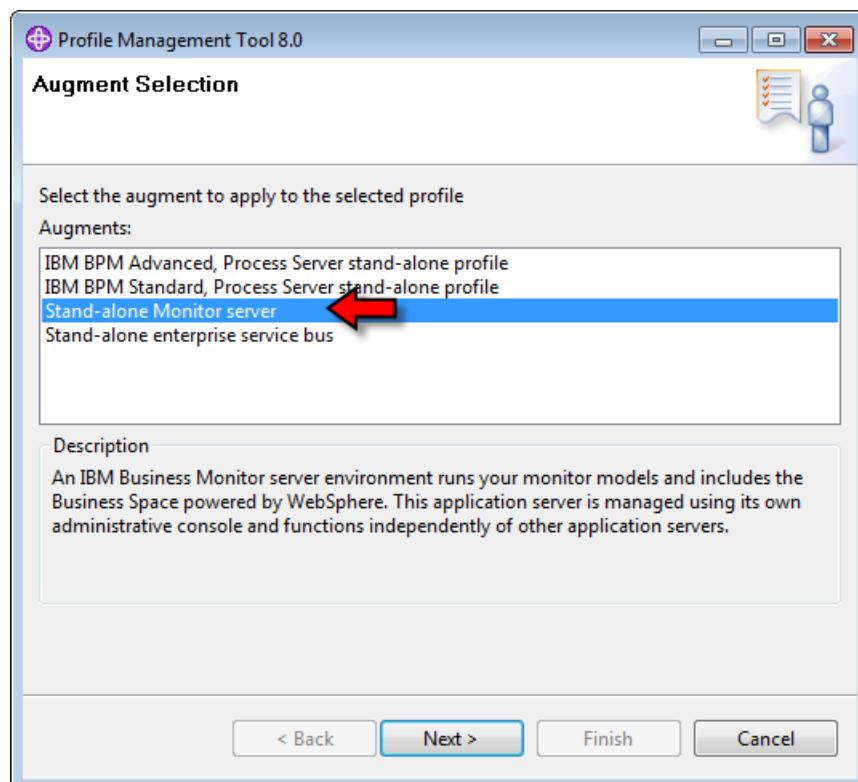
## ***Augmenting an existing profile with Business Monitor***

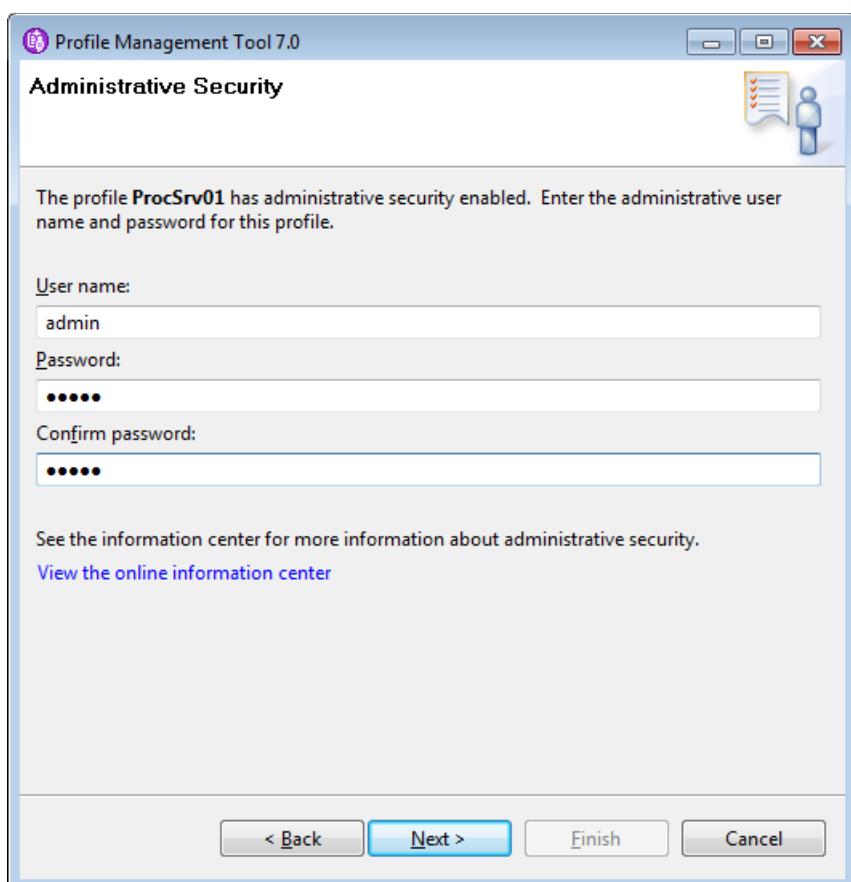
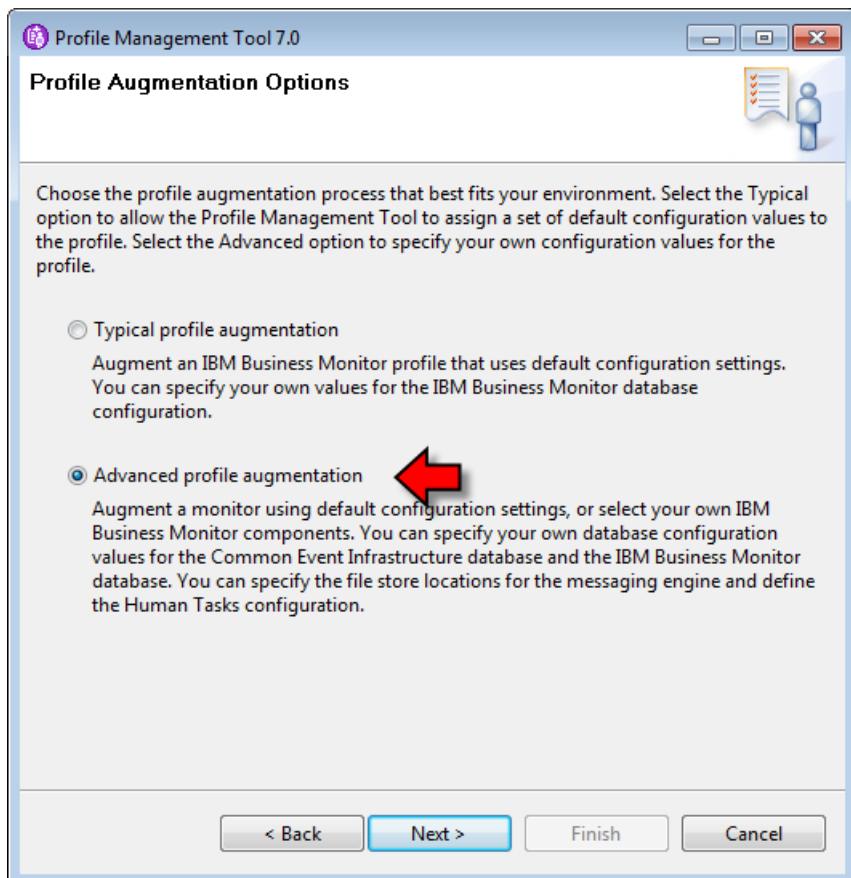
Augmenting a profile is the task of updating a profile definition with additional functions. Monitor can be added to an existing profile including a profile for Process Center and/or Process Server. To

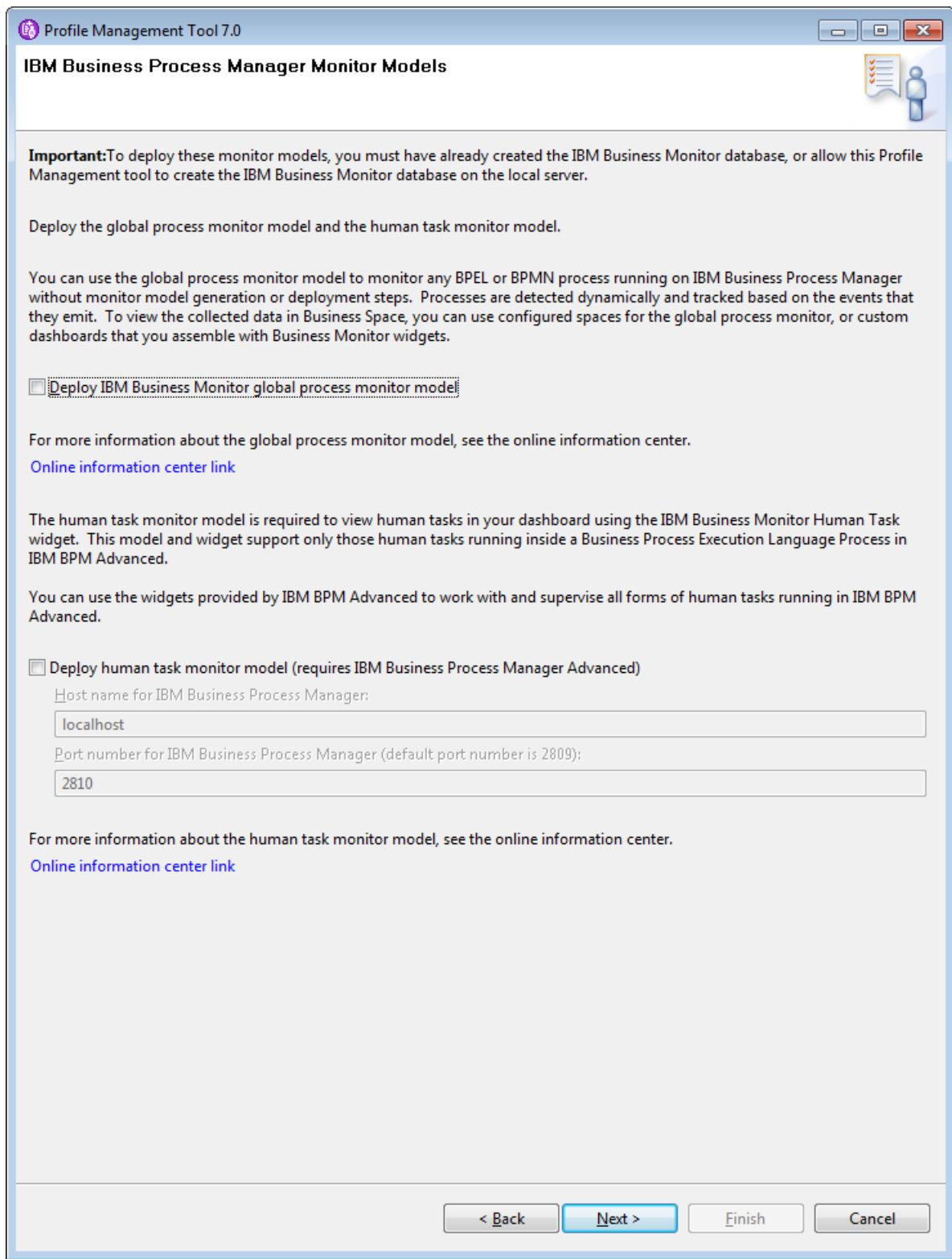
perform this task, launch the PMT tool and select the server to be augmented. Once selected, click the Augment button to begin the steps.

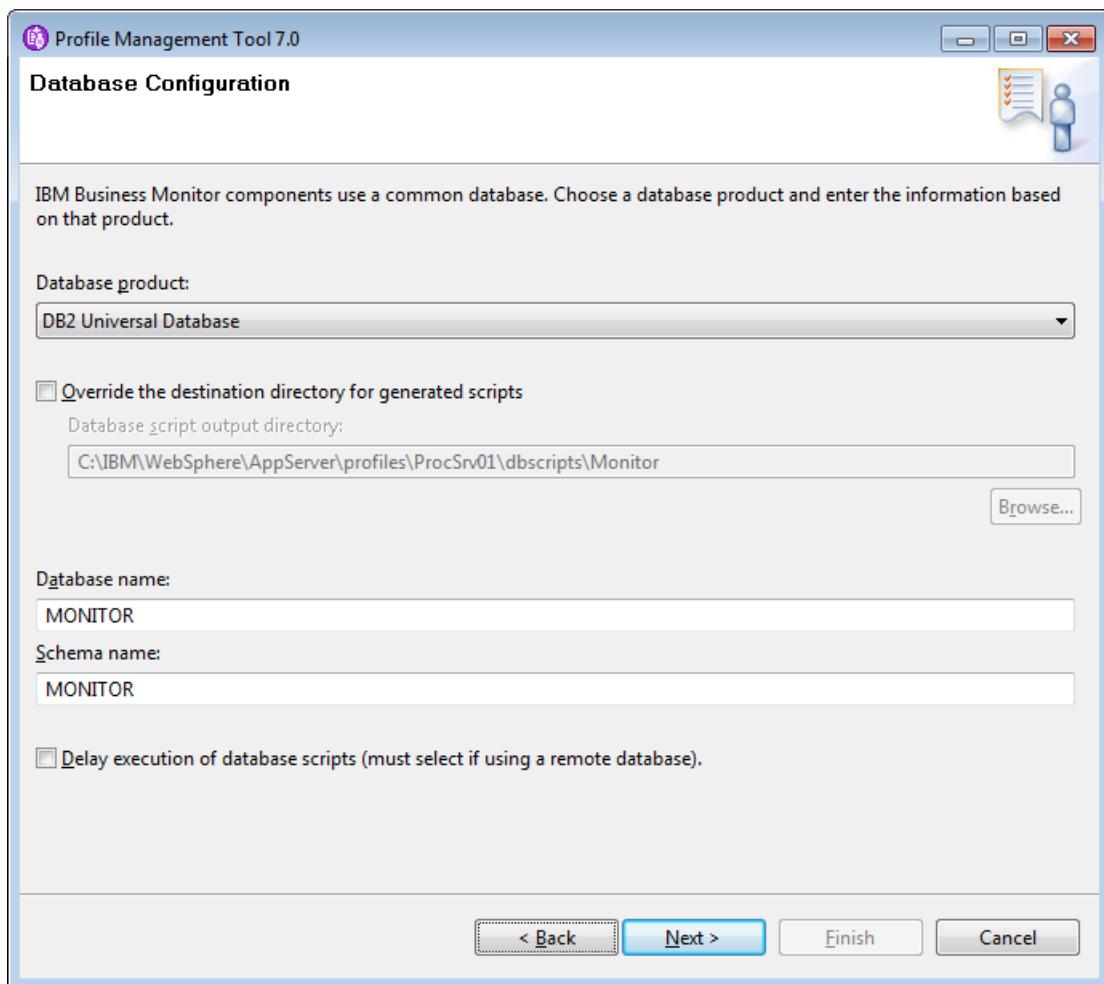
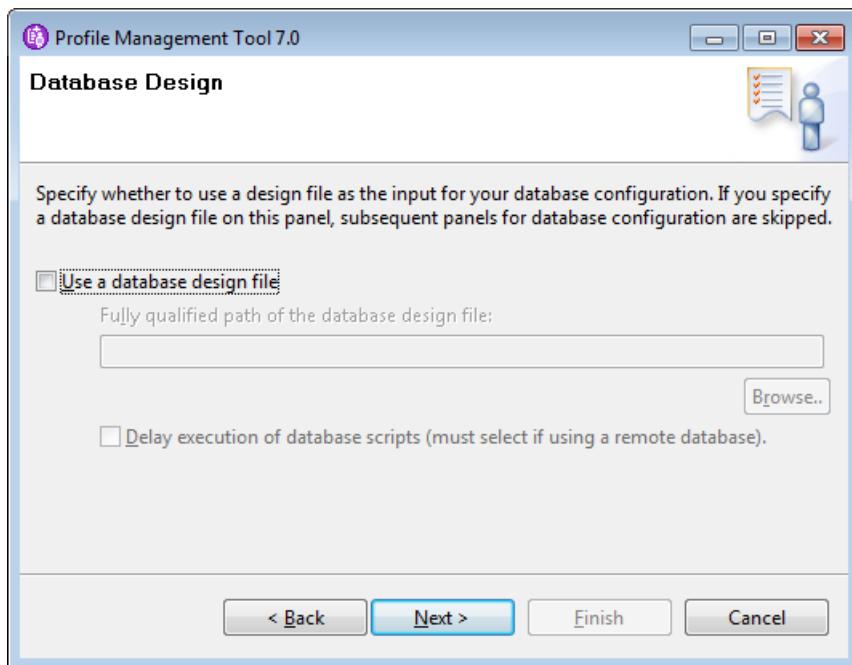


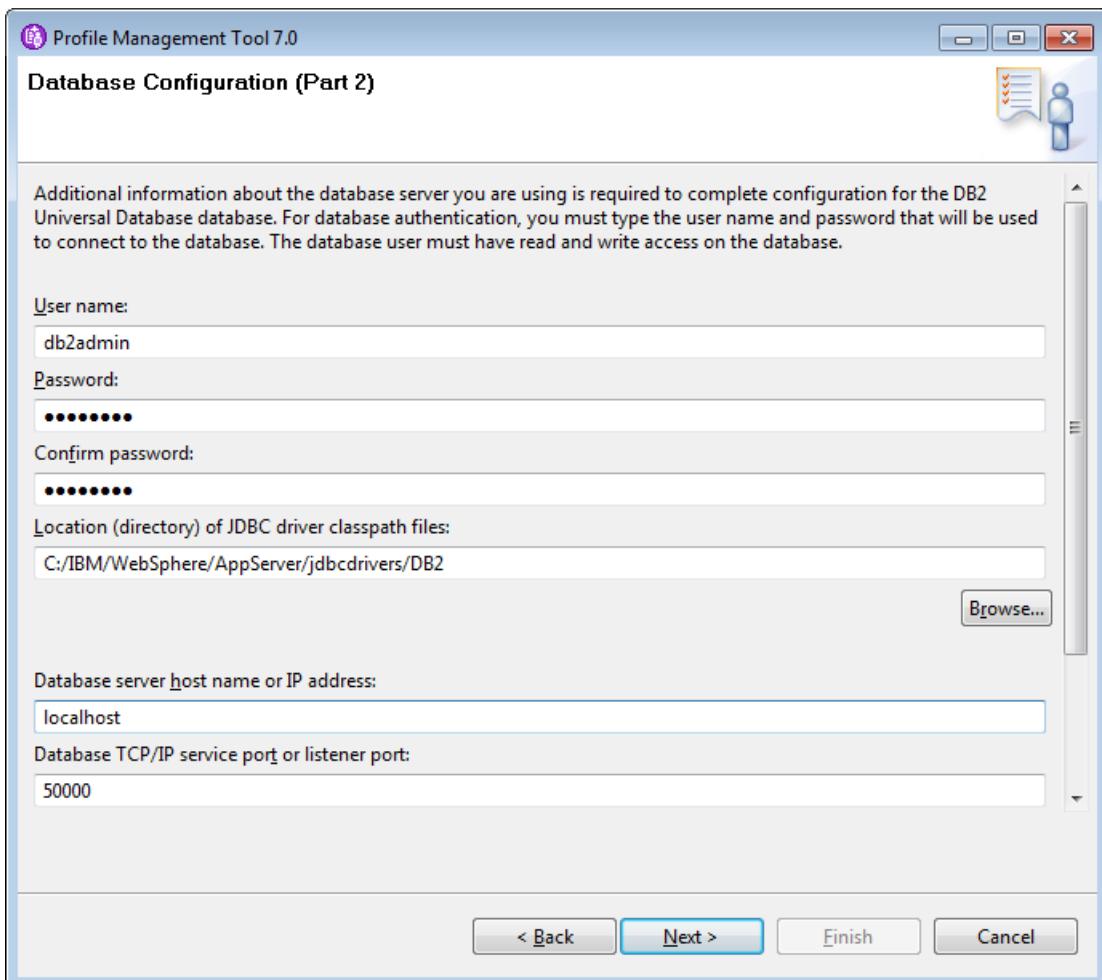
From the augmentation types available, select the Monitor server.











Profile Management Tool 7.0

## Cognos Configuration

Configure IBM Business Monitor for multidimensional data analysis.

Create a new Cognos server configuration. You must provide a database user name with credentials to create tables.

Database name:  
COGNOSCS

Database user name:  
db2admin

Database password:  
\*\*\*\*\*

Confirm password:  
\*\*\*\*\*

The new Cognos server configuration uses the following values from the IBM Business Monitor database configuration:

Database product: DB2\_UNIVERSAL  
Database server host name or IP address: localhost  
Location (directory) of the JDBC driver classpath files: C:/IBM/WebSphere/AppServer/jdbcdrivers/DB2

Execute database scripts during this profile augmentation.

Use an existing Cognos server configuration.

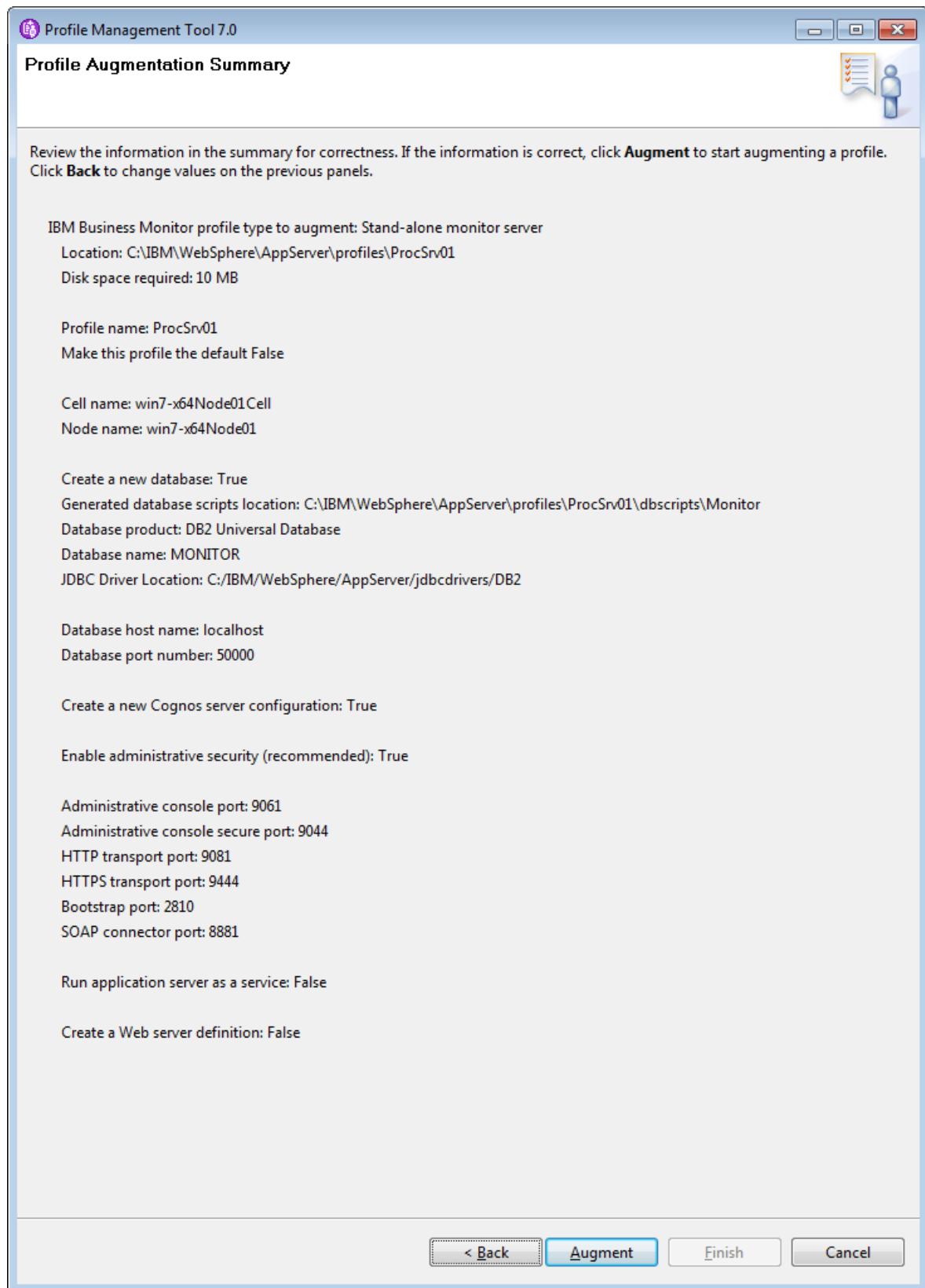
External dispatcher URI:

Cognos administrator user name:

Cognos administrator password:

Confirm Cognos administrator password:

< Back    Next >    Finish    Cancel



## Information Sources for Business Monitor

### *InfoCenter for Business Monitor*

The InfoCenter for IBM Business Monitor can be found at:

<http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r0m1/index.jsp>

The screenshot shows the IBM InfoCenter interface. At the top, there's a navigation bar with links for Home, Solutions, Services, Products, Support & downloads, and My IBM. A search bar is present, along with a 'Scope: All topics' dropdown and a 'Go' button. Below the navigation is a 'Contents' tree. The 'IBM Business Monitor, V8.0.1' node is expanded, showing various subtopics like 'Getting started with IBM Business Monitor', 'Scenarios', and 'Glossary'. A large red arrow points to the 'IBM Business Monitor, V8.0.1' node.

A downloadable version can be found here:

<http://www-01.ibm.com/software/integration/business-monitor/library/>

## ***Redbooks***

- [Business Activity Monitoring with WebSphere Business Monitor V6.1](#) – SG24-7638-00 - 2008-07-30

## ***DeveloperWorks for Business Monitor***

DeveloperWorks hosts many articles on IBM Business Monitor.

- [Monitoring business processes: How your topology influences your monitoring approach](#) – 2013-08-07
- [Some tips for IBM Business Monitor modeling and versioning](#) - 2013-01-30
- [What's new in IBM Business Monitor V8](#) - 2013-09-26
- [Troubleshooting common CEI-related issues with IBM Business Monitor](#) - 2012-09-20
- [Monitoring business processes with IBM Business Process Manager and IBM Business Monitor](#) - 2011-12-14
- [Monitoring very long-running business processes with WebSphere Business Monitor](#) - 2011-12-14
- [Enabling event flow in BPM solutions and best practices for monitor model deployment, Part 1: Configuring and verifying event flow between a clustered Process Server and IBM Business Monitor server](#) - 2011-11-16
- [Monitoring business processes by using tracking groups in IBM BPM V7.5](#) - 2011-08-10
- [Business activity monitoring with IBM Business Monitor V7.5, Part 1: What's new in IBM Business Monitor V7.5](#) – 2011-06-29
- [Business activity monitoring with IBM Business Monitor V7.5, Part 2: A quick tour of the dashboards](#) – 2011-06-29

- [Business activity monitoring with IBM Business Monitor V7.5, Part 3: Monitoring IBM Business Process Manager](#) – 2011-06-29
- [Business activity monitoring with IBM Business Monitor V7.5, Part 4: Customizing the dashboard experience](#) - 2011-06-29
- [Monitoring your business applications, Part 2: Products offering basic integration with WebSphere Business Monitor](#)[Monitoring your business applications, Part 2: Pro](#)
- [Smart interactive process design for modeling and monitoring](#) – 2011-01-19
- [Multi-module monitoring with the V7 WebSphere BPM suite, Part 1: Use two new plug-ins for WebSphere Business Modeler V7 to achieve end-to-end business process monitoring](#) - 2010-10-06
- [Monitoring and administering human tasks with WebSphere Business Monitor, Part 2: Using and customizing the global human task model](#) - 2010-12-14
- [BPM voices: Joachim Frank: Behind the scenes of WebSphere Business Monitor event processing](#) – 2010-10-06
- [Multi-module monitoring with the V6.2 WebSphere BPM suite](#) - 2010-10-06
- [Building smarter event-driven business processes with the WebSphere BPM suite](#) – 2010-05-05
- [Comment lines: Modeling for execution, revisited](#) – 2010-04-14
- [Configure database high availability for WebSphere Business Monitor V6.2](#) – 2010-04-12
- [Business Event Processing with WebSphere Business Events, Part 5: Integrating Business Events with WebSphere Business Monitor](#) - 2010-03-15
- [Dimensional analysis in business activity monitoring \(BAM\), Part 2: Improve performance in WebSphere Business Monitor V6.2 with materialized query tables](#) – 2010-02-24
- [A technical deep dive into the global process monitor model](#) – 2010-02-03
- [Developing custom widgets for Business Space using Dojo, Part 1: Generating Dojo markup using a generic markup handler](#) - 2009-12-09
- [Developing custom widgets for Business Space using Dojo, Part 2: Creating a Dojo DataGrid component using a generic grid handler](#) – 2010-01-20
- [Develop custom widgets for Business Space with Rational Application Developer or WebSphere Integration Developer](#) – 2009-12-20
- [Business process monitoring in WebSphere Process Server V6.2 using WebSphere Business Monitor V6.2, Part 1: Configuring the integration](#) – 2009-12-16
- [WebSphere Business Modeler and WebSphere Business Monitor feedback loop](#) – 2009-12-15
- [How to use event processing in CICS: Part 3, Consuming CICS events with WebSphere Business Monitor](#) – 2009-11-25
- [Generate events for monitoring in WebSphere Business Monitor V6.2 from an Oracle database](#) – 2009-10-28
- [Configure WebSphere Business Monitor for high throughput and performance](#) – 2009-11-19
- [Improve performance and usability by implementing TopCount functionality for WebSphere Business Monitor dimensions with DB2 Alphablox](#) – 2009-11-02
- [Dimensional analysis in business activity monitoring \(BAM\), Part 1: Leverage WebSphere Business Monitor V6.2 to perform dimensional analysis](#) – 2009-08-25
- [Monitoring your SAP system using WebSphere SAP Adapter and WebSphere Business Monitor](#) – 2009-07-22
- [What's new in WebSphere Business Monitor V6.2](#) – 2009-01-28
- [Use Common Event Infrastructure for business-level logging to improve business processes](#) - 2005-12-07

## Introducing the Monitor Model

To provide business level monitoring, IT systems must produce data that is recorded. We can imagine a system producing an event that a sale has occurred. The event data may be described as XML, character delimited, COBOL copybook or some other format of data. This is the concrete representation of the data. It may be stored in files or database tables. However, this is too low-level for our purposes. What we really want to do is record the conceptual information. We want to have a *model* of the data under consideration and have the real world physical data be used to populate the notion of our model.

Let us try and make this real. Imagine we wish to answer the question "What was the average value of an order?" To be able to answer this, we need to record the value of each order as it is

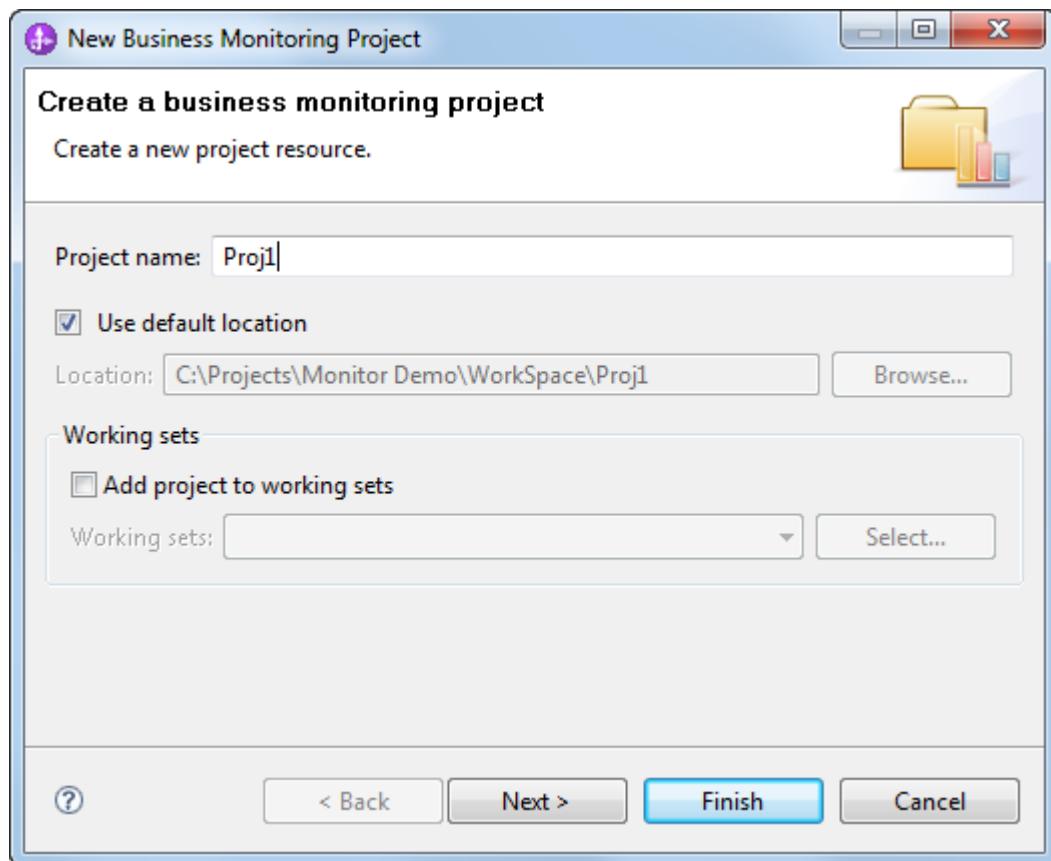
encountered. An order may be taken by a variety of sources. For example, a web site may result in an order or a phone call to a CSR. Each of these may result in an event, possibly in a different format, indicating that an order has been made. However the order is received, all of them indicate *an order*. This leads us to the notion of a "an order" as a logical entity separate from the physical data. The set of logical entities in aggregate are described as a *model*. The model describes:

- What are the logical entities, which are called *metrics*, that are to be recorded?
- How do we map incoming data to populate these metrics
- More to be described

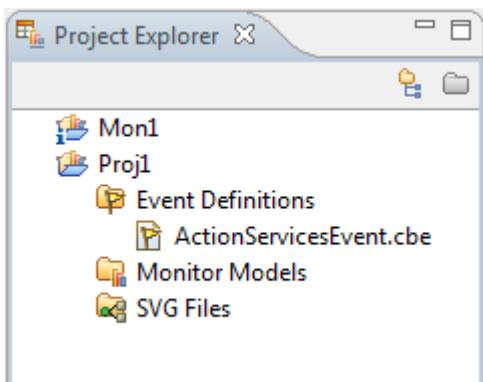
The overview then of how Business Monitor is used is that a developer builds a Monitor Model (MM) describing the metrics that they would like to work with. Next the developer will map the real incoming data to those metrics describing when and how they are populated.

Monitor models are created and edited in an Eclipse Perspective that can be found in IID.

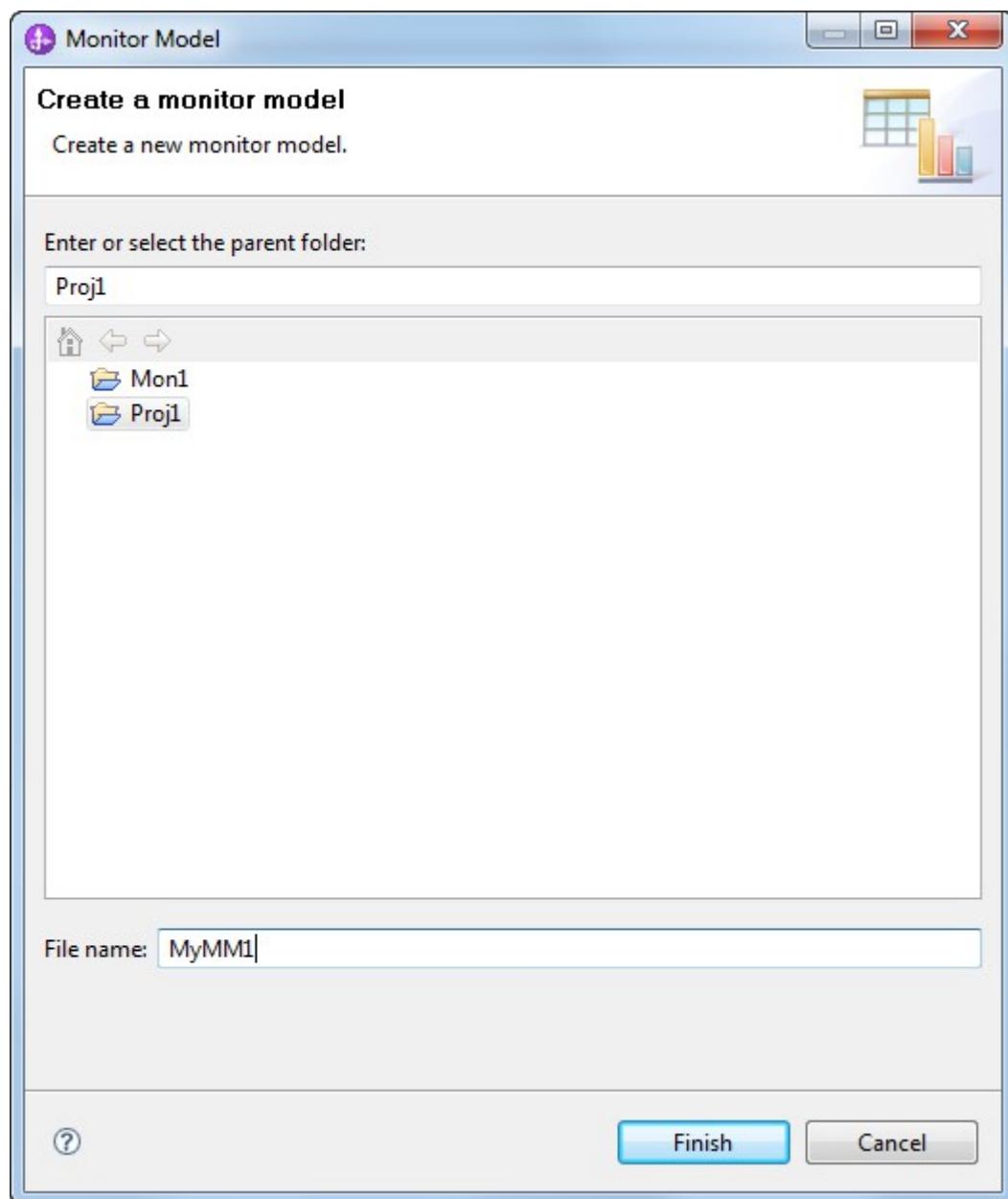
Before building a model, a monitor project must be constructed. A simple wizard is presented into which the name of the new monitor project can be supplied.



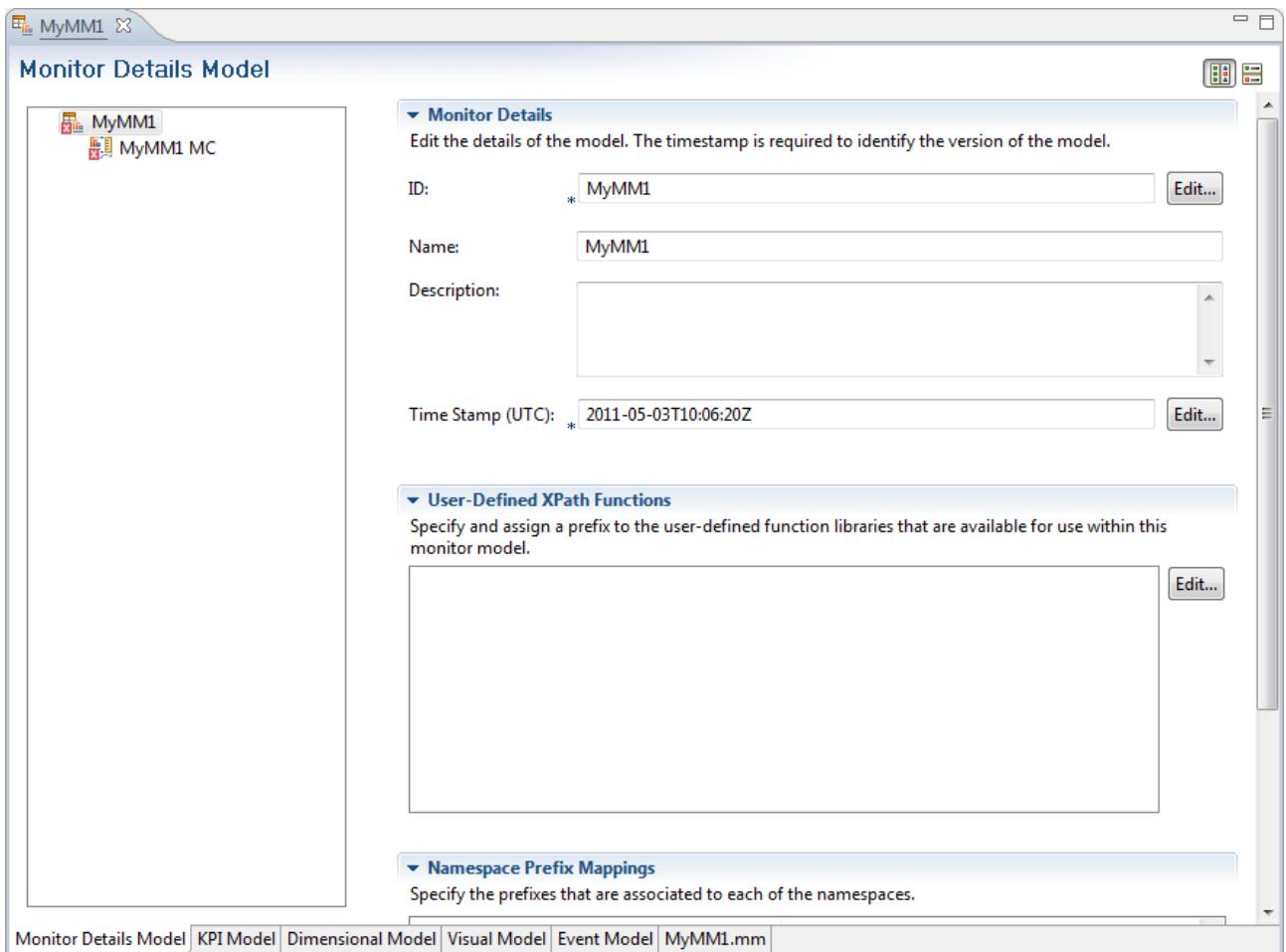
Once created, the project holds a number of folders. The one of interest to us right now is Monitor Models.



The wizard for creating the Monitor Model is also very simple.



Once created, the Monitor Model editor is opened. It is within this editor that the vast majority of monitor model definitions are performed.



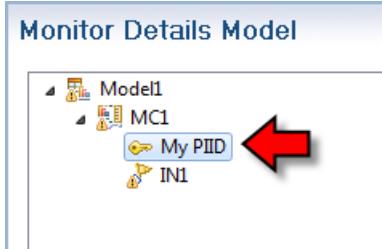
## Monitoring Context

A monitoring context is part of the Monitor Model. A context defines a set of metrics and other attributes that are related together through a common thread. For example, if a sales order has a description of what is being ordered, a price and an actual shipped date, it may be that this data is **not** generated at one time. For example, when the order is first received, we know what is being ordered and how much it is selling for but it is some time later that we know the date of actual shipment. If the data that describes this sale arrives in pieces over time then we need to ensure that the distinct pieces are contextually related to each other. It is the Monitoring Context that allows us to group together these properties. As subsequent events arrive which belong to the same monitoring context, the model for that instance is updated with the new information.

## Key Fields

Within a Monitoring Context one or more fields must be flagged as key fields. It is the key fields that provide the identity of the context.

Key fields show up in the monitor model with a key icon beside them



The properties of the key details look as follows:

**Key Details**  
Edit the details of the key. Each monitoring context requires at least one key.

ID:	<input type="text" value="piid"/> Edit...
Name:	<input type="text" value="My PIID"/>
Description:	<input style="height: 100px; width: 100%;" type="text"/>
Type:	<input type="text" value="String"/> *
Maximum String Length: <input type="text" value="256"/>	
<input type="checkbox"/> Allocate additional space in database to accommodate Unicode string for globalization	
<input type="checkbox"/> A value is required for this key	
Default Value:	<input type="text"/> Edit...
<input type="checkbox"/> This key can be used for sorting	

**Key Value Expressions**  
Specify the expressions that set the value of the key.

Expression IN1/My_Event_Part/mon:eventPointData/mon:correlation/wle:starting-process-instance
<input type="button" value="Add"/> <input type="button" value="Remove"/>

A core aspect of the key is the XPath expression used to set its value.

## Inbound Event

The Inbound event monitor model definition describes an external event that is to be recognized by this monitor model. An inbound event is external data that physically arrives at Business Monitor. The data has to be in XML format. The inbound event conforms to an XML Schema Definition (XSD) that is used to map the incoming data.

The event data descriptions are described through either CEI or through XSDs created in another section.

#### ▼ Event Type Details

Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both.

Extension name:

[Browse...](#)

[Clear](#)

Event parts:

ID	Name	Type	Path

[Add](#)

[Remove](#)

When an event is received, all monitoring context are given the opportunity to work with the event. The mandatory Filter Condition executes an XPath expression against the received data. If the expression evaluates to true, then the event will be *handled* by this model, otherwise the model will ignore the event as though it had never happened.

#### ▼ Filter Condition

Define a condition based on the event attributes to identify whether to accept an event of this type.

Once an event is determined to be appropriate for this model, the next question becomes one of which instance of the model should the event be delivered? Here an expression is supplied that is used to match the data in the event against a specific instance of the monitor model context. Rules then govern what to do in each of the possible circumstances where zero, one or many instances are found.

#### ▼ Correlation Expression

Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

If no instances are found

If one instance is found

If multiple instances are found

## Metric

The Metric definition defines a specific data item that can be used by dashboard designers for populating those dashboards. The Metrics are the logical entities of the model that we eventually wish to populate with data received contained within externally published events that are received by the Inbound Event.

Each Metric definition has an ID and a Name. The ID is the unique value maintained for the

metric. The Name is a human readable annotation for the metric.

Each of the metrics has an associated data type. The allowable data types are:

- Boolean
- Date
- DateTime
- Decimal
- Duration
- Integer
- String
- Time

#### ▼ Metric Details

Edit the details of the metric, which is a holding spot for information used in other calculations.

ID:	<input type="text" value="My_Metric"/> *	<input type="button" value="Edit..."/>
Name:	<input type="text" value="My Metric"/>	
Description:	<input type="text"/>	
Type:	<input type="text" value="String"/> *	
Maximum String Length: <input type="text" value="256"/>		
<input type="checkbox"/> Allocate additional space in database to accommodate Unicode string for globalization		
<input type="checkbox"/> A value is required for this metric		
Default Value:		<input type="button" value="Edit..."/>
<input type="checkbox"/> This metric can be used for sorting		

#### ▼ Metric Value Expressions

Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.

Trigger	Expression

Within the metric definition is the ability to supply an expression that will be used to populate the

metric from an incoming piece of data.

## Key

## Counter

## Stopwatch

The Stop watch is a timer that can be started, stopped or reset. Initially, it is not counting but when an event or a trigger arrives, it can be told to start. Other events and triggers can be used to stop or reset it.

## Trigger

A trigger is an indication that something has happened. Specifically, a trigger can occur when a metric's value changes, another trigger fires, an inbound event arrives or a clock timer fires. Let us call this the trigger's source event.

In addition to the source event, there is an optional expression which can be evaluated. The outcome of this expression is either true or false. If no expression is supplied, the trigger will fire every time the source event occurs. If an expression is supplied, then when the source event happens, the expression will be evaluated. Only if the expression results in a true value, will the trigger actually fire.

A boolean attribute on the trigger is available called "Trigger is repeatable". This indicates whether or not the trigger may fire more than once for this monitoring context.

If is set to true (checked), then the trigger will fire every time its source event happens. If a condition is also supplied, then obviously the trigger will only fire if the condition is also true.

If the repeatable flag is set to false, then if there is no condition supplied, the trigger will fire once and once only. If a condition is supplied, it will fire the first time a source event happens and the condition is true and will not fire again until the condition becomes false and then true once more.

Another attribute of the trigger is called "Terminate monitoring context". If this flag is set to true then when the trigger fires, this is the indication that the monitoring context is no longer needed and the monitoring context will be flagged as "closed".

As mentioned previously, the source event that can cause a trigger to fire can be a time base. The trigger definition for time events has three possible options.

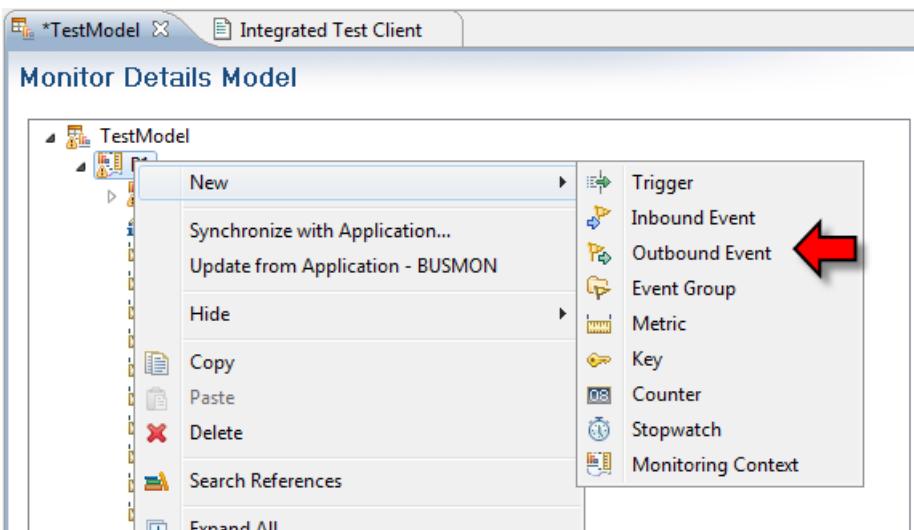
The first is a recurring evaluation. This is an interval of time after which the trigger condition will be evaluated (if one is supplied) and the trigger will fire. If the trigger is flagged as repeatable, the interval will be repeated otherwise it will be a one-off fire.

The second option is a specific time evaluation. This is a one-time trigger that will happen on the date/time specified.

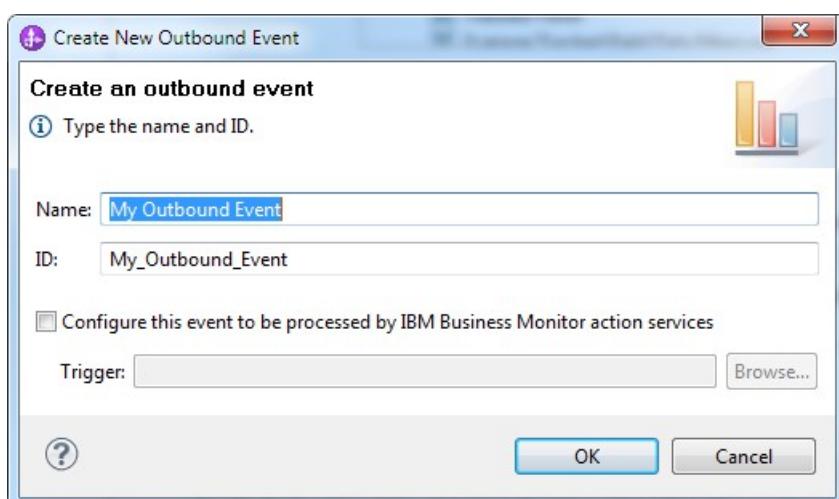
The third option is a daily evaluation which will fire at a specific time of the day each day that the monitoring context is open.

## Outbound Event

Monitor not only has the ability to receive events for processing, it has the ability to publish events for others to consume. One of the core consumers of this is the notification mechanism. Within a monitor model we can define "Outbound Event" definitions which declare when an outbound event should be fired and what it may contain.



When created, we get to name the event and also define



In the creation wizard, we can define that this is an event that will be destined for the Monitor action services and what trigger will be used to cause the event to be emitted.

Commonly we define a value for the "BusinessSituationName" which will be used as the key for the type of action service to be performed:

Outbound Event Content			
Specify the triggers that cause the event to be sent. Use the Expression column to specify the value for each event attribute when the event is sent.			
Name	Type	Expression	
AmountChanged			
Property Data			
Extended Data			
BusinessSituationName	A string	X <sup>SY</sup> ? 'HighValueSale'	← Red arrow points here
OrderID	xs:string	X <sup>SY</sup> ? "ABC"	
Amount	xs:decimal	X <sup>SY</sup> ? bmon_f_amount	

# Key Performance Indicators

So far we have just discussed the population of modeled metrics based on the arrival of events. An example of such a metric might be the duration it takes to process an individual order. Now we look at aggregating these metrics together, the result of which we term a "Key Performance Indicator". For example, if we aggregate a set of sales order durations using an average function, we now find the average duration of sales orders. That becomes a very meaningful concept.

In a KPI definition, we get to define a metric that will be aggregated across its instances. The aggregation functions supplied by IBM include:

- Average – The average of metrics from the time set
- Minimum – The minimum value of a metric from the time set
- Maximum – The maximum value of a metric from the time set
- Sum – The sum of metrics from the time set
- Count – The number of metrics in the time set
- Standard Deviation - ??

When we define a KPI we give it a Name and an ID. The Name is used to display the KPI in a human readable form when it is displayed to the end user, while the ID is used to track the KPI internally and through APIs.

For numeric values, we can supply a "Decimal Precision" property which is the number of decimal places to show if the value of the KPI is shown. The value of the KPI is not altered by this property ... it is available to any display technology to format the output.

We can also request that the KPI be displayed as a "percentage". This means (for example) that if the value of a KPI were 0.3, it would be displayed as "30%".

**KPI Details**  
Edit the details of the KPI, which is a performance measurement used to track business objectives.

ID: \* My\_KPI

Name: My KPI

Description:

Type: \* Decimal

Currency  Decimal Precision 0   Show as a percentage

Hide from dashboards  
 Keep track of historical values for this KPI

The Time Filter section provides the range of time over which the set of metrics are calculated.

I like to think of a KPI like a dial on my car dashboard. At a quick glance I can tell if I am going too fast/slow, what my fuel level is and other key aspects of the operation of my car. A KPI is not the same as a chart ... instead, the KPI should be used to show that you are operating within an acceptable range.

Let us use an example to illustrate KPIs. Imagine that we are processing sales orders and we have a target of \$1000 worth of sales per day. Each sale that is made creates a metric called "amount" that represents the amount of a sale. To calculate our KPI, we want to calculate the sale value over the last day.

We create a new

The time filter allows us to specify which instances of metrics should be used to calculate the KPI. For example, if we have been running our process for many months and have been collecting information, the daily sales should be calculated from what happened today and should not include sales metrics from previous days. When we define our KPI, we have the option of specifying a "Time Filter" which defines whether to include or exclude metrics. The Time Filter expression has four options available to us:

- None – Include all available historic metrics. Effectively no filtering is performed and the KPI is based on all data available.

- Repeating – This defines an interval of time that is a window into the data. It may have one of four options:
  - Daily – Data is calculated on a per day basis
  - Monthly – Data is calculated on a per month basis
  - Quarterly – Data is calculated on a per quarter basis
  - Yearly – Data is calculated on a per year basis

We can also calculate the KPI as a function of a full "period" or the "in progress" period. To illustrate this, if we say that the period is "daily" and we select "Last full period", then if today is Friday at 11:30am, then the KPI will be built from the data from Midnight to Midnight of Thursday (yesterday). Any data from today (Friday) will not be included. However, if select "Period in progress", then the data will be built from Midnight of today (Friday) up to and including now.

**Time Filter**  
Select a time period over which the KPI should be calculated.

Metric: Start Time

Time period:

None  Repeating  Rolling  Fixed

Period type: \*  Time zone: \*  Location (daylight saving):

Base period on  Last full period  Period in progress

- Rolling – This setting builds KPI data from a sliding window of data. The options for the period can be one of the following Minutes, Hours, Days, Months or Years.

**Time Filter**  
Select a time period over which the KPI should be calculated.

Metric: Start Time

Time period:

None  Repeating  Rolling  Fixed

Last \*  \*

- Fixed – This setting builds KPI data from metrics sourced over a fixed period of time. The start and end dates/times can be supplied.

**Time Filter**  
Select a time period over which the KPI should be calculated.

Metric: Start Time

Time period:

None  Repeating  Rolling  Fixed

Start date:   End date:

Time zone: \*  Location (daylight saving):

A KPI usually has a "target" associated with it. This is our "goal". It is what we would hope to achieve or expect. By setting a target, we can see if we are higher or lower than the desired amount.

In addition to defining a target, we can define "ranges" for our data. These ranges provide visual indications of where are relative to our goal.

**KPI Target and Ranges**

Specify a target, which is an exact value for the KPI to achieve, or ranges against which to track the KPI, or both.

Target:  [Details...](#)

Ranges:  [▼](#)

Range Name	Start Value	End Value	Color
Low	0	< 300	#ff0000
Medium	300	< 700	#ffff00
High	700	< 1200	#00ff00

[Add](#) [Remove](#) [Sort](#)

## Alerts and Actions

If monitor detects an event that should be made known to a user, it can use the concept called "Situation Events".

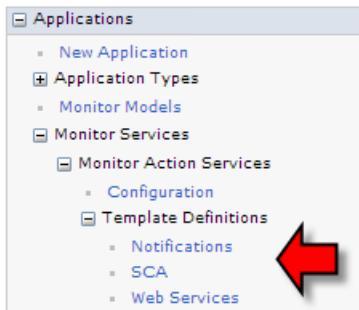
Action processing is managed by the WAS application called "`IBM_WBM_ACTIONSERVICES`".

### Action Service templates

When a situation event occurs, it can be serviced by one of three possible types of action:

- Notifications
- SCA
- Web Services

These are defined in the WAS Admin Console:



### Notification Templates

The notification template describes HOW we wish to notify someone when a business situation happens. These notifications are available in four distinct types:

- alert – Add an alert to a dashboard
- email – Send an email
- cell phone – Call a cell phone

- pager – Call a pager

They all use the same format of template. Core within the template are places where we can define what the "Subject" and "Body" of the notification will be composed from. Think of this as describing WHAT the user will see when the notification is made.

When creating a new template definition, it looks as follows:

**Notifications > New...**

Use this page to specify the configuration properties of a notification template.

**Notification Template Configuration**

**General Properties**

\* **Template name**

**Description**

The additional properties will not be available until the general properties for this item are applied or saved.

**Additional Properties**

Template XML namespaces

**Default action service type**

Dashboard Alert  
 Cell phone  
 Email  
 Pager

**'To' query type**

Federated repositories query  
 LDAP query  
 Email address  
 User id

To

Query base

Owner kolban

Priority 3

Subject

Body

Apply OK Reset Cancel

- Template name – Every template must have a unique named and is used as the key to look it up.

- 'To' query type – Defines the algorithm used to determine a set of users to which the notification will be delivered. See also the "To" field.
- To – If supplied, this will be the expression used to define which users that the notification will be sent. If left empty, users can subscribe to the notification from the dashboard.
- Owner – If there are no subscribers to the notification, this field can be used to name an owner who will be alerted.
- Priority – A field on a notification that can be used to set the priority of the notification.
- Subject – The subject line of the notification. Can use meta code to describe data.
- Body – The body of the notification. Can use meta code to describe data.

Both the "Subject" and "Body" properties can use meta code to describe data. The meta code can be:

- %<CBE Variable>% - A variable within the CBE
- %%<XPath Expression>% - An Xpath expression to data. Note that when using this feature, you may have to define additional XML namespaces and prefix your own data. For example, if the payload of a business situation is an XML document, then the fields will have to be prefixed with the namespace.
- #% - The "%" symbol itself
- ## - The "#" symbol itself

Let us look at an example of this:

Consider an outbound event definition that looks as follows:

**Outbound Event Details**

Edit the details of the outbound event, which is sent by the monitoring context. The type must be an event definition.

ID:	<input type="text" value="HighValueSale"/> *	<a href="#">Edit...</a>
Name:	<input type="text" value="HighValueSale"/>	
Description:	<input type="text"/>	

**Event Type Details**

Specify the event type or the XML schemas that together describe the structure of this outbound event. You can specify an extension name, event parts, or both.

Extension name:	<input type="text" value="ActionServicesEvent"/> <a href="#">Browse...</a> <a href="#">Clear</a>												
Event parts:	<table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Type</th> <th>Path</th> </tr> </thead> <tbody> <tr> <td>OrderID</td> <td>OrderID</td> <td>xs:string</td> <td>cbe:CommonBaseEvent/ss</td> </tr> <tr> <td>Amount</td> <td>Amount</td> <td>xs:decimal</td> <td>cbe:CommonBaseEvent/amount</td> </tr> </tbody> </table>	ID	Name	Type	Path	OrderID	OrderID	xs:string	cbe:CommonBaseEvent/ss	Amount	Amount	xs:decimal	cbe:CommonBaseEvent/amount
ID	Name	Type	Path										
OrderID	OrderID	xs:string	cbe:CommonBaseEvent/ss										
Amount	Amount	xs:decimal	cbe:CommonBaseEvent/amount										
<input type="button" value="Add"/> <input type="button" value="Remove"/>													

**Outbound Event Content**

Specify the triggers that cause the event to be sent. Use the Expression column to specify the value for each event attribute when the event is sent.

Name	Type	Expression
AmountChanged		
Property Data		
Extended Data		
OrderID	xs:string	<input type="text" value="ABC"/> <a href="#">X+Y</a> <a href="#">Z?</a>
Amount	xs:decimal	<input type="text" value="bmon_f_amount"/> <a href="#">X+Y</a> <a href="#">Z?</a>

Notice specifically the addition of an event property called "Amount" that is populated from a Monitor Model metric. This means that when this event is published, it will also contain event specific data.

In our template, we can refer to the value of that property via the meta string:

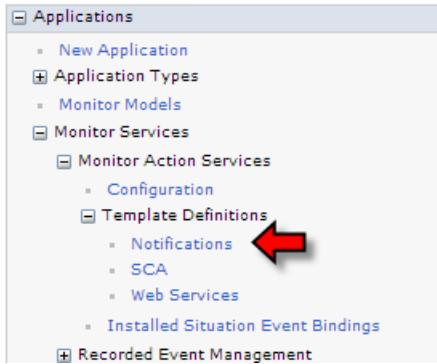
```
%%cbe:CommonBaseEvents/cbe:CommonBaseEvent/amount/text()%
```

With these concepts in mind, let us now see if we can put some of them together. Consider a scenario where you want to be informed of a high value sale on your dashboard. This is an example of an alert.

The first thing we will want to do is to create a "template" used by Monitor's action services subsystem. This template describes "what" we want to happen when a situation such as a high value sale is detected.

Templates can be added, viewed or deleted from the WAS admin console at:

[Applications > Monitor Services > Monitor Action Services > Template Definitions > Notifications](#)



Within the template definition, we have choices of what type of action service to built. We will concentrate on the "Dashboard Alert".

The template describes what shall be the content of the Subject and Body.

Once the template has been built, we have an abstract definition of what the user might see if a business situation is encountered but we have not yet said "which" events will result in the creation of such an alert. That is where the next part comes in. What we now do is bind a "Business Situation Event" (a technical thing) with the template that defines what should happen when that technical thing is detected. We again perform this task through the IBM WAS Admin Console.

Applications > Monitor Services > Installed Situation Event Bindings



Within this area, we can add, view and delete the bindings/relationships between events and templates. The administration user interface here is a little confusing so let us take some time. When we first visit the link, we will see an empty table that looks as follows:

Installed Situation Event Bindings	
Use this page to manage situation event bindings.	
<input checked="" type="checkbox"/> Preferences	
<input type="button" value="New..."/> <input type="button" value="Delete"/>	
Select	Situation Event Name
None	
Total 0	

If we click the New button to create a new entry, we are presented with the following screen:

[Installed Situation Event Bindings](#) > New Situation Event Binding

Use this page to bind one or more templates and their respective action services to a situation event.

New Situation Event Binding

**General Properties**

\* Situation event name

Description

Apply OK Reset Cancel

**Preferences**

Add Remove

Select	Binding Name ▾	Category Name ▾	Template Name ▾	Action Service Type ▾
None				
Total	0			

The first thing we need to do is enter the value for the "Situation event name". This is the technical part that is the "key" in the data that says "Hey... an event has happened that I want you to tell someone about!". Once we enter a value here, we **must** hit the Apply button before we can go further.

[Installed Situation Event Bindings](#) > HighValueSale

Use this page to bind one or more templates and their respective action services to a situation event.

New Situation Event Binding

**General Properties**

\* Situation event name

Description

Apply OK Reset Cancel

**Preferences**

Add **R** Remove

Select	Binding Name ▾	Category Name ▾	Template Name ▾	Action Service Type ▾
None				
Total	0			

Once done, we can now define the association between THIS event and a template that describes how we want to inform the user. Clicking the Add button opens a new screen. Within this screen we get to provide a name for our binding association and finally, the ability to specify which action template to use to handle the event.

[Installed Situation Event Bindings](#) > HighValueSale > Add template to situation event binding

Use this page to add a template to a situation event binding.

Add template to situation event binding

**General Properties**

\* Binding name  
My Alert

Description

Category

Template name  
Not1 ▾

Apply OK Reset Cancel

The result will be:

[Installed Situation Event Bindings](#) > HighValueSale

Use this page to bind one or more templates and their respective action services to a situation event.

New Situation Event Binding

**General Properties**

\* Situation event name  
HighValueSale

Description

Apply OK Reset Cancel

**Preferences**

Add Remove

Select Binding Name Category Name Template Name Action Service Type

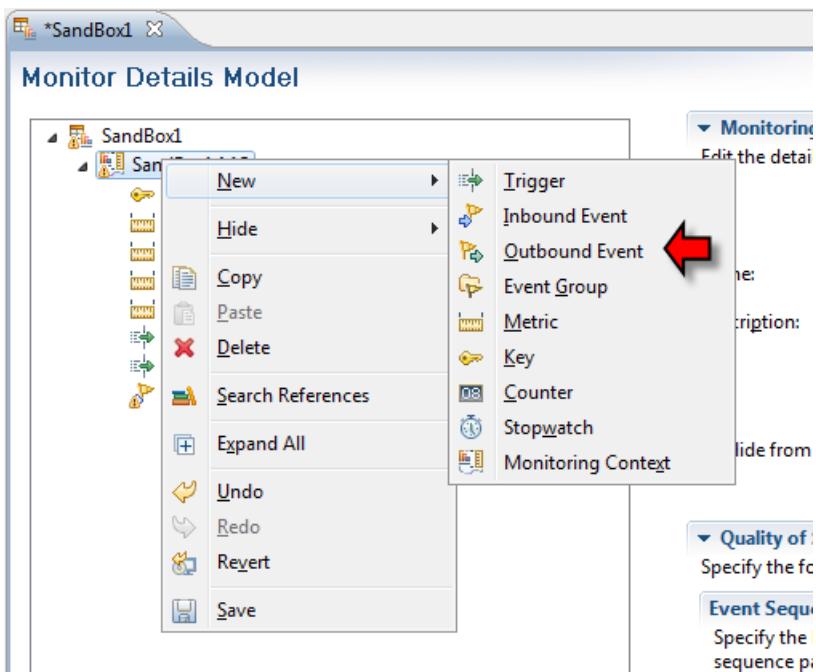
You can administer the following resources:

My Alert	Not1	AlertHandler
Total 1		

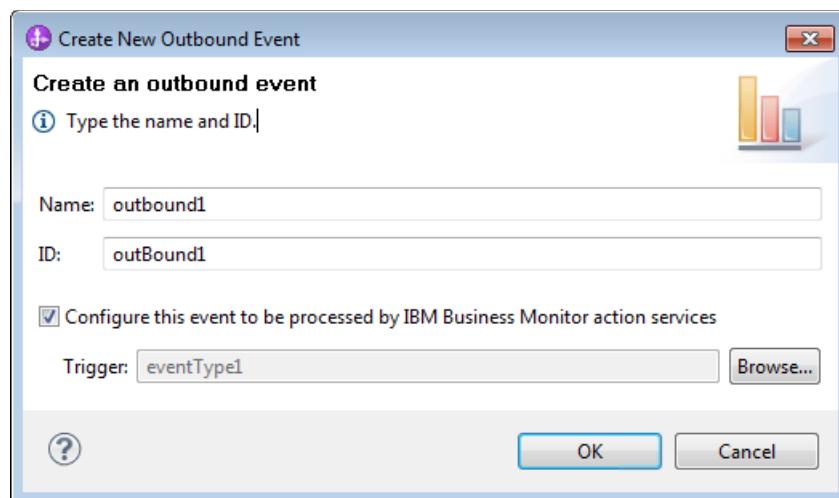
That sure is a lot of indirections!! The way to interpret this important page is as follows:

"When a business situation event of type 'HighValueSale' happens, then execute the alert service described by the template called 'Not1'. This binding is known by the name of 'My Alert' and it so happens that the template type is an 'AlertHandler'".

With all this in place, we now think about how an event is generated by the monitor model. What we do is create an instance of an "Outbound Event"



During creation, we have the option to check the box labeled "Configure this event to be processed by IBM Business Monitor action services". It this that defines this event as the source of an alert. The outbound event must also be associated with a Monitor Model trigger to determine when the event should be fired.



Because we checked the box that said this was a monitor action event, the payload of the event has been partially filled in for us. Among its properties is one called "BusinessSituationName". This property is very important. This is the key used to determine which template to invoke. This maps to the "Situation event name" defined in a previous WAS admin panel.

Outbound Event Content		
Specify the triggers that cause the event to be sent. Use the Expression column to specify the value for each event attribute when the event is sent.		
Name	Type	Expression
eventType1		
Property Data		
Extended Data		
BusinessSituationName	string	X+Y =? 'outbound1'

[Add](#) [Remove](#)

## Monitor SVG Diagrams

Graphic images are commonly found in raster formats such as JPEG and PNG. An alternative format is the Scalable Vector Graphic (SVG).

Shapes in an SVG diagram can:

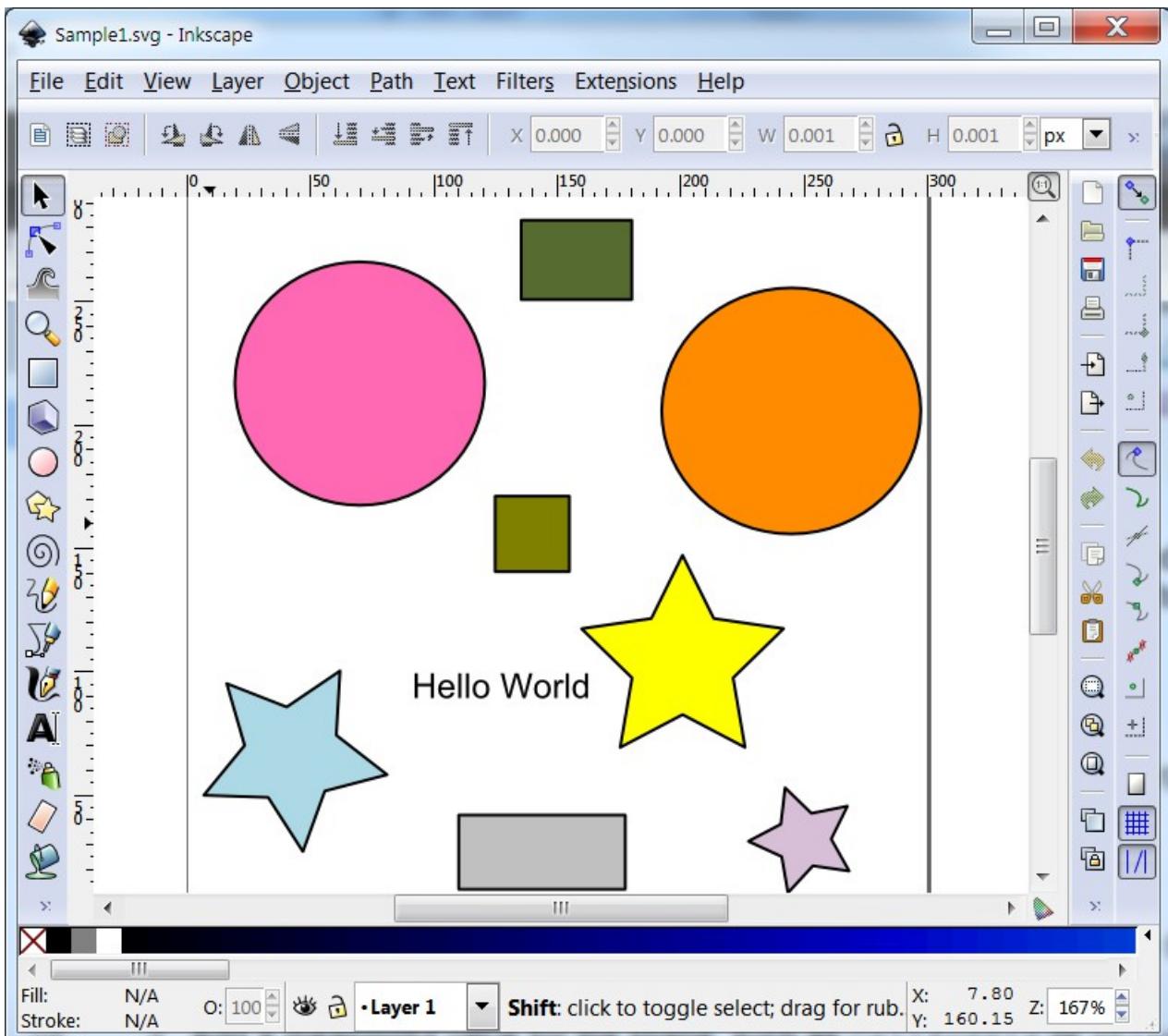
- Change shape fill color
- Change shape outline color
- Display Text
- Link to other diagrams
- Hidden/displayed

An SVG file (which is really an XML document) must be modified for it to be used by Monitor. Edit the file and add the following to the <svg> tag:

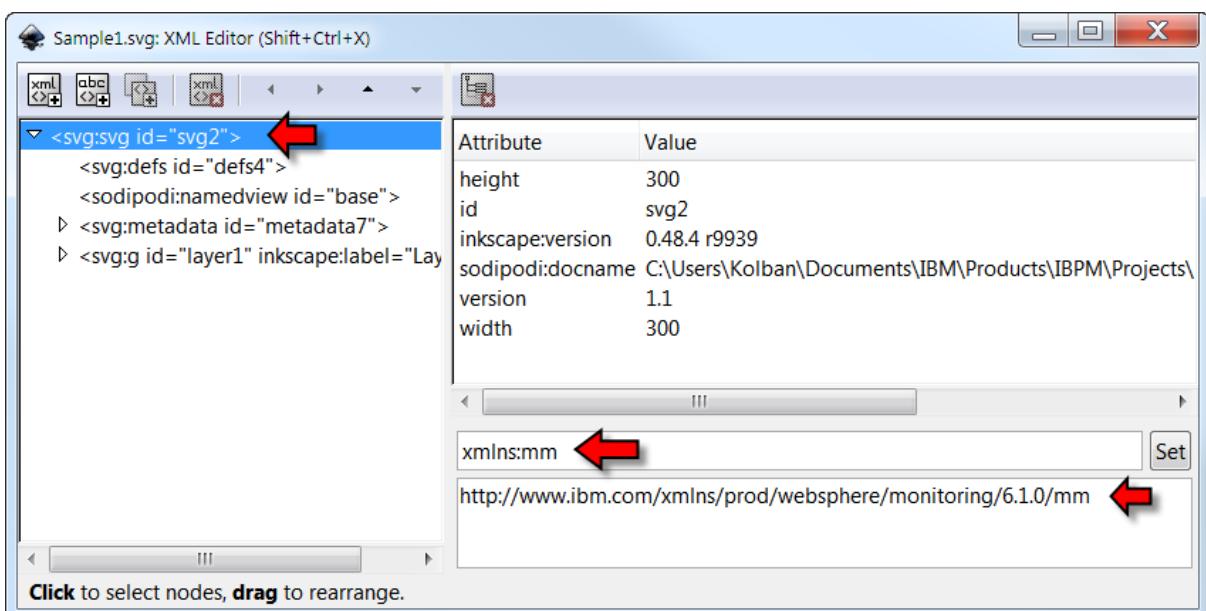
```
xmlns:mm="http://www.ibm.com/xmlns/prod/websphere/monitoring/6.1.0/mm"
```

The most popular tool for working with SVG diagrams is called "Inkscape". We can edit our diagram within that tool to prepare it for Monitor usage.

After opening our base diagram in Inkscape, we will be presented with a view similar to the following:



From the Edit menu, select XML Editor. Select the root of the document and set the property as shown in the following screen shot:



Next you can click on parts of the diagram and give them identifiers by adding an attribute called "mm:id" and a value corresponding to the ID name. Once the SVG is saved and loaded into IID's Monitor editor,

From the Instances widget, be sure and change its configuration by checking the "Enable this widget to send events over wires" from the wiring tab. This will add a "synchronize" icon to the Instances widget. When clicked, the diagram will be changed to reflect the data associated with that instance.

For each SVG shape entry that was marked with an identifier, we have the ability to define one or more actions.

The screenshot shows the 'TestModel' Visual Model editor window. At the top, it displays the 'Selected Context: P1' and 'SVG File: /BUSMONMonitoringProject/Sample1.svg'. Below this are two main sections: 'Add and Edit Shape Set Actions' and 'Sample Metric Values'.

**Add and Edit Shape Set Actions:** This section allows defining actions for shape sets. It lists actions for shapes 'sq1' and 'txt1'. For 'sq1', there are two 'Set color' actions: one for condition 'bmon\_f\_var1 = "A"' (false) and another for 'B' (true). For 'txt1', there is one 'Set text' action with condition 'bmon\_f\_var1 = "B"' and value 'Hello World'.

Action Attribute Value	Test Result
bmon_f_var1 = "A"	false
#00ff00	
bmon_f_var1 = "B"	true
#ff0000	
"Hello World"	Hello World

**Sample Metric Values:** This section shows metric values for 'var1' (string type, value 'B').

Metric Name	Type	Value
var1	string	B

At the bottom, there are buttons for 'Add...', 'Remove', 'Expand All', 'Collapse All', and 'Test'.

The types of actions are described in the following.

See also:

- DeveloperWorks - [Model and simulate business processes, Part 2: Displaying SVG diagrams on the WebSphere Business Monitor dashboards](#) - 2007-06-05

## SVG Action – Set Color

This action sets the color of the shape. It includes the following properties:

- condition – An expression which, if it evaluates to true, will result in the action being performed.
- fill color – The color to be used to fill the shape.
- outline color – The color to be used for the stroke of the shape.

## ***SVG Action – Set Text***

This actions sets the text of a text area. It includes the following properties:

- condition – An expression which, if it evaluates to true, will result in the action being performed.
- text color – The color to be used for the text.
- text value – The string to be used for the text. Can be evaluated as a function.

## ***SVG Action – Hide Shapes***

This action hides (or shows) the shape as a function of an expression.

- condition – An expression which, if it evaluates to true, will result in the action being performed.

## ***SVG Action – Set Diagram Link***

This action changes the diagram being shown to a different context when the shape entry is clicked.

- condition – An expression which, if it evaluates to true, will result in the action being performed.
- target context – ??

## ***SVG Action – Send Human Task Notification When Clicked***

This action sends a human task notification.

- condition – An expression which, if it evaluates to true, will result in the action being performed.
- human task instance ID – ??
- human task name – ??
- human task namespace – ??

## ***SVG Action – Send Notification When Clicked***

This action sends a notification when the shape entry is clicked.

- condition – An expression which, if it evaluates to true, will result in the action being performed.
- event code – ??

# **Generating events from a BPMN process**

Before a Process Application can generate events for Monitor, it must be instructed that it can communicate with Monitor. This is achieved through the Process Application settings in a process application:

\* Process App Settings

Overview Environment Servers

## Process App Settings

**Common**

Name: Book Orders  
Acronym: BK2  
Description: Click [Edit](#) to add or edit text.

**Exposed Items**

These items have been exposed from this Process App. They are normally the starting points for the design and implementation.

- Business Process Diagrams: <none>
- Human Services: [Order Entry](#)
- Web Services: <none>

**Authorization Settings**

**Heritage Coach Designer Settings**

**Monitor Settings**

Enable process monitoring through IBM Business Monitor:

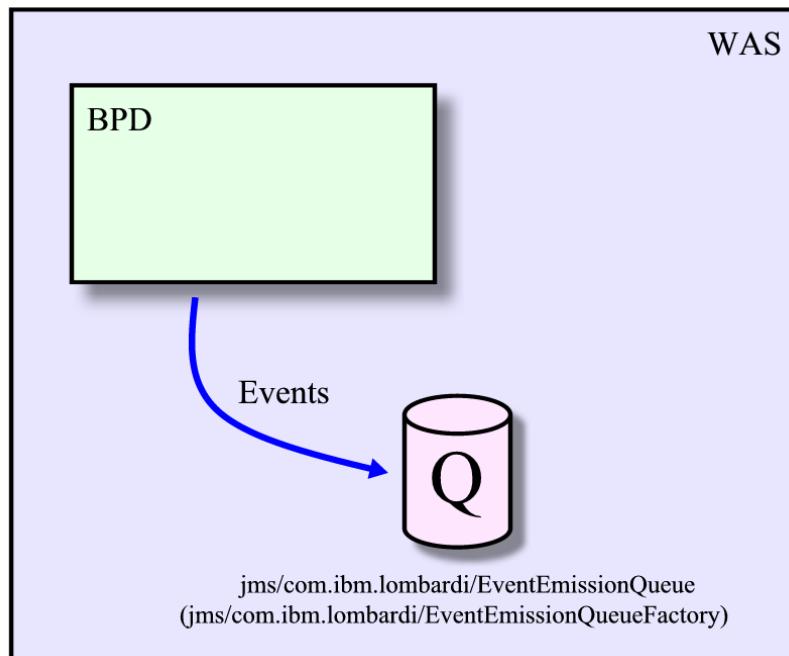
**Advanced XML Settings**

Once this change has been made, when either a new snapshot is deployed or an update of the tracking definitions is performed, the generated Monitor Model EAR will be deployed to the Process Server.

The name of the generated application will be:

bmon\_<ProcessAppSynonym>\_Main

A BPD in a Process Application can emit events. It does this by building and sending event messages to a JMS queue.



The identity and details of the queue can be found in JNDI at:

jms/com.ibm.lombardi/EventEmissionQueue

An instance of this resource is created when a Process Server profile is built. Its default WAS resource name is LombardiEventEmissionQueue which resolves to an SIBus queue name of LombardiEventEmitterInputQueue.

There must also be a JMS Queue Connection factory found at JNDI:

jms/com.ibm.lombardi/EventEmissionQueueFactory

An instance of this resource is created when a Process Server profile is built. Its default resource name is LombardiEventEmissionQCF. This points to an SI Bus called "MONITOR.<cellname>.Bus"

By default, Process Center does not have the event emission resources defined to it. However, if the JNDI resources are defined, it will start using them.

In summary, these resources are needed:

- jms/com.ibm.lombardi/EventEmissionQueueFactory – A Queue Connection Factory
- jms/com.ibm.lombardi/EventEmissionQueue – A Queue

Resource	Type	Value
LombardiEventEmissionQCF	JMS Queue Connection Factory	jms/com.ibm.lombardi/EventEmissionQueueFactory
LombardiEventEmissionQueue	JMS Queue	jms/com.ibm.lombardi/EventEmissionQueue

It is important to take a pause here and fully understand the story so far. In my experience, it is important to understand the architecture in order to be fully successful. What we have is that IBM BPM generates messages that are written to an SI Bus queue. The content of these messages are XML documents. It is here that IBM BPM ends its part of the Monitor interaction. At this point, the events have **not** made their way to monitor and merely reside in a queue. In the next part we will see how the messages are consumed by Monitor but before going there we have one more thought.

What if Monitor is **not** installed on the same cell or machine as IBM BPM? We will shortly see that Monitor has to be able to read the messages from the queue and if Monitor is remote we seem to have a problem as a remote machine can't read a queue that is local to IBM BPM. The solution to this comes from the IBM WAS SI Bus technology. SI Bus has the concept of "Foreign Connections". This means that we can define a network link from the WAS server hosting IBM BPM to the remote WAS server hosting Monitor.



From BPM's perspective, it continues to think that it is writing messages to a named queue and from Monitor's perspective, it thinks it is reading from a named queue but WAS is managing the movement of messages from WAS 1 to WAS 2.

Mechanically, the way we achieve this is through the SI Bus notion called the "foreign bus connection".

### **The IBM\_BPM\_EMITTER\_SERVICE application**

The messages sent to the JMS queue by BPM are (obviously) written to the target queue and are also not in a format that Monitor can directly consume. IBM has supplied an application called "IBM\_BPM\_EMITTER\_SERVICE" that watches a JMS queue for such messages and, when they arrive, formats them for Monitor and delivers them to Monitor. This application should be installed on a machine which is part of the Monitor environment and which can also read from the BPM written queue. Since 8.5 of BPM can't co-exist on the same cell as Monitor, we normally set up a cross cell queuing environment where BPM writes the message to a queue which is actually remote from BPM. The message then ends up on the machine hosting Monitor which is where the IBM\_BPM\_EMITTER\_SERVICE will be installed.

The application uses three WAS level JMS resource definitions:

- jms/com.ibm.lombardi/JMSEmitterInputQueueFactory
  - WAS Resource Name – LombardiEventEmitterInputQCF
  - Points to MONITOR.<Cellname>.Bus
- jms/com.ibm.lombardi/JMSEmitterInput
  - WAS Resource Name – LombardiEventEmitterInputQueue
  - Points to SIBus Queue: LombardiEventEmitterInputQueue
- jms/com.ibm.lombardi/JMSEmitterInputActivationSpec
  - WAS Resource Name – LombardiEventEmitterInputAS

To install the IBM\_BPM\_EMITTER\_SERVICE application, a wsadmin command has been supplied called "wbmDeployBPMEmitterService":

```
AdminTask.wbmDeployBPMEmitterService(['-cluster', '<clustername>'])
AdminConfig.save()
```

eg.

```
AdminTask.wbmDeployBPMEmitterService(['-cluster', 'SingleCluster'])
AdminConfig.save()
```

### ***Diagnosing problems***

If when you run a process and the data from that process does not show as instance data within a model, obviously, data is not moving from the process to monitor. Here is a check-list of items that can be checked.

### **BPM side definitions**

Within the IBM BPM side of the house we have the following JMS definitions:

- jms/com.ibm.lombardi/EventEmissionQueueFactory – A Queue Connection Factory
- jms/com.ibm.lombardi/EventEmissionQueue – A Queue

## Monitor side definitions

The following JMS definitions:

- jms/com.ibm.lombardi/JMSEmitterInputQueueFactory
  - WAS Resource Name – LombardiEventEmitterInputQCF
  - Points to MONITOR.<Cellname>.Bus
- jms/com.ibm.lombardi/JMSEmitterInput
  - WAS Resource Name – LombardiEventEmitterInputQueue
  - Points to SIBus Queue: LombardiEventEmitterInputQueue
- jms/com.ibm.lombardi/JMSEmitterInputActivationSpec
  - WAS Resource Name – LombardiEventEmitterInputAS

## Configuring CEI on BPM

```
AdminTask.wbmDeployCEIEventService(['-busMember [-cluster <clusterName>
-datasourceJndiName <jndiName> -datasourceAuthAlias <authAlias> -databaseSchema
<schemaName> -createTables true] -eventService [-cluster <clusterName>] -jmsAuthAlias [-user
<userName> -password <password>]]')
```

for example

```
AdminTask.wbmDeployCEIEventService(['-busMember [-cluster SingleCluster
-datasourceJndiName jdbc/CEI -datasourceAuthAlias BPM_DB_ALIAS -databaseSchema
db2admin -createTables true] -eventService [-cluster SingleCluster] -jmsAuthAlias [-user admin
-password admin]]')
```

AdminConfig.save()

Notes:

It may be that the datasourceJndiName has to be a new unique entry

## Configuring for remote Monitor ↔ BPM

In many cases, the Process Servers running IBM BPM will be separate from the server running Monitor. Because of this, the monitoring events in the form of JMS messages will have to be remotely "shipped" from a queue on IBM BPM to a queue on the Monitor server.

Collect the following:

Property	BPM	Monitor
Host Name	localhost	localhost
SOAP Port	8879	8885
WAS Admin Userid	cadmin	admin

WAS Admin Password	password	admin
Link Userid	monlink	monlink
Link Password	password	password
Cluster Name	SingleCluster	

WBM provides scripts and sample properties files to assist in setting this up.

Create a userid called “admin” (if it doesn't already exist) on both servers.

A properties file was created that contained the following (a template for such can be found at <Root>/scripts.wbm/crossCell/configRemoteMonitorBus.props):

```

SECURE_CONFIGURATION=true
LOCAL_WAS_HOST=localhost
LOCAL_WAS_PORT=8882
LOCAL_WAS_USERID=admin
LOCAL_WAS_PASSWORD=admin
LOCAL_WAS_MESSAGING_ENGINE=win7-x64Node03.server1-MONITOR.win7-x64Node03Cell.Bus

REMOTE_WAS_HOST=localhost
REMOTE_WAS_PORT=8880
REMOTE_WAS_USERID=admin
REMOTE_WAS_PASSWORD=admin
REMOTE_WAS_BUS_USERID=admin
REMOTE_WAS_BUS_PASSWORD=admin

LINK_USERID=admin
LINK_USERID_REMOTE_PASSWORD=admin
LINK_USERID_LOCAL_PASSWORD=admin

REMOTE_WAS_ME_NODE=win7-x64Node01
REMOTE_WAS_ME_SERVER=server1
#REMOTE_WAS_ME_CLUSTER=
REMOTE_WAS_ME_STORE=default

```

Next, the command "configRemoteMonitorBus.bat -props <fileName>" was executed.

The servers were then restarted.

At this point I gave up ... not because it was broken but because I opted instead for augmenting my BPM server with WBM.

New instructions

Configure Server to Server SSL

Start both BPM and CEI

Login to WAS Admin Console

Go to

Security > SSL certificate and key management > (Related Items) Key stores and certificates

Select NodeDefaultTrustStore

> (Additional Properties) Signer Certificates

Click Retrieve from port

Enter details

[Ports](#)

Port Name	Port	Details
BOOTSTRAP_ADDRESS	9810	
SOAP_CONNECTOR_ADDRESS	8881	
ORB_LISTENER_ADDRESS	9102	
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	9405	
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	9406	
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	9407	
WC_adminhost	9062	
WC_defaulthost	9081	
DCS_UNICAST_ADDRESS	9355	
WC_adminhost_secure	9045	
WC_defaulthost_secure	9444	
SIP_DEFAULTHOST	5061	
SIP_DEFAULTHOST_SECURE	5062	
OVERLAY_UDP_LISTENER_ADDRESS	11008	
OVERLAY_TCP_LISTENER_ADDRESS	11009	
IPC_CONNECTOR_ADDRESS	9634	
SIB_ENDPOINT_ADDRESS	7287	
SIB_ENDPOINT_SECURE_ADDRESS	7287	
SIB_MQ_ENDPOINT_ADDRESS	5559	
SIB_MQ_ENDPOINT_SECURE_ADDRESS	5579	

[Messages](#)

Click OK to add the new entry

Go to Security > SSL certificate and key management > manage endpoint security configurations  
Realm

defaultWIMFileBasedRealm

Results after execution:

```
C:\IBM\WebSphere80\AppServer\scripts.wbm\crossCell>configRemoteMonitorBus.bat -props
configRemoteMonitorBus.props
IBM WebSphere Application Server, Release 8.0
ava EE Application Client Tool
copyright IBM Corp., 1997-2009
SCL0012I: Processing command line arguments.
SCL0013I: Initializing the Java EE Application Client Environment.
12/29/13 10:09:54:570 CST] 00000000 W UOW=null source=com.ibm.ws.ssl.config.SSLConfig org=IBM
prod=WebSphere component=Application Server thread=[P=394289:O=0:CT]
CWPKI0041W: One or more key stores are using the default password.
SCL0035I: Initialization of the Java EE Application Client Environment has completed.
SCL0014I: Invoking the Application Client class com.ibm.wbmonitor.sib.configassist.Launcher
```

```
This utility will create a foreign bus link between the Monitor bus and a remote cell
Reading properties from file configRemoteMonitorBus.props
Creating remote Bus MONITOR.PCcell1.Bus
Adding application server Node1 : SingleClusterMember1 to the bus
Creating the foreign buses
Creating the foreign bus links
Saving the configuration changes
Saved
You must restart both application servers for this change to take effect
C:\IBM\WebSphere80\AppServer\scripts.wbm\crossCell>
```

After execution, the following resource changes will be found:

## On BPM

Resource Name	Resource Type	Example
MonitorBusAuth	J2C authentication data	cadmin
MonitorBusAuthLink	J2C authentication data	monlink
MONITOR.<Cell>.Bus	SI Bus	MONITOR.PCell1.Bus
MONITOR.<MonCell>.Bus	Foreign SI Bus Link	MONITOR.win7-x64Node01Cell.bus

## On Monitor

Resource Name	Resource Type

See also:

- [InfoCenter article on "configRemoteMonitorBus"](#) – 8.0.1
- TechNote - [1424259](#) - Parameters for the configRemoteMonitorBus utility – 2010-07-03
- DeveloperWorks - [Business process monitoring in WebSphere Process Server V6.2 using WebSphere Business Monitor V6.2, Part 1: Configuring the integration](#) - 2009-12-16

## Event data format

The format of the event data written to the JMS queue by IBPM's BPDs is an XML document. This data is well documented in a number of detailed articles in the IBM BPM InfoCenter.

Here is an example of an event message:

```
<?xml version="1.0" encoding="UTF-8"?>
<mon:monitorEvent mon:id="R7aa8b5a191e031792162189"
 xmlns:bpmn="http://schema.omg.org/spec/BPMN/2.0"
 xmlns:bpmnx="http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring"
 xmlns:ibm="http://www.ibm.com/xmlns/prod/websphere/monitoring/7.5/extensions"
 xmlns:mon="http://www.ibm.com/xmlns/prod/websphere/monitoring/7.5"
 xmlns:wle="http://www.ibm.com/xmlns/prod/websphere/lombardi/7.5"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
<mon:eventPointData>
 <mon:kind mon:version="2010-11-11">bpmnx:PROCESS_STARTED</mon:kind>
 <mon:time mon:of="occurrence">2011-06-30T12:21:16.728-05:00</mon:time>
 <ibm:sequenceId>0000000002</ibm:sequenceId>
 <mon:model mon:id="f4396aed-63c7-4da9-b66d-bcc051c7dd41"
 mon:type="bpmn:process"
 mon:version="2064.a3ac4487-a8f3-47ee-9bfd-51491d944d6T">
 <mon:name>Sale</mon:name>
 <mon:instance mon:id="653">
 <mon:state>Active</mon:state>
 </mon:instance>
 </mon:model>
 <mon:model mon:id="68c2430a-bc66-4460-8e1a-df27cbcd7706"
 mon:type="wle:processApplication" mon:version="2064.a3ac4487-a8f3-47ee-9bfd-251491d944d6T">
 <mon:name>Mon1</mon:name>
 <mon:documentation/>
 </mon:model>
 <mon:correlation>
 <mon:ancestor mon:id="f4396aed-63c7-4da9-b66d-bcc051c7dd41.2064.a3ac4487-a8f3-47ee-9bfd-251491d944d6T.653"/>
 <wle:starting-process-instance>f4396aed-63c7-4da9-b66d-bcc051c7dd41.2064.a3ac4487-a8f3-47ee-9bfd-251491d944d6T.653</wle:starting-process-instance>
 </mon:correlation>
</mon:eventPointData>
</mon:monitorEvent>
```

At a high level, it is composed of:

mon:monitorEvent – The root element of the event.	
• @mon:id – A unique identifier for this event instance	
mon:eventPointData – The details of the event. There is only one of these per event.	
mon:kind – Defines the kind of event	
mon:time – The time the event was generated	
ibm:sequenceId -???	
mon:model – An executable model element	
• @mon:type – ???	
mon:instance	
mon:correlation	
mon:applicationData	
wle:tracking-point	
wle:kpi-data	
wle:tracked-field	

The monitoringEvent/eventPointData/kind is a critical entry. It defines the kind of the event that is received. The options include:

bpmnx:PROCESS_STARTED	An instance of a process has been started. The models supplied include: <ul style="list-style-type: none"><li>• type=bpmnProcess</li><li>• type=wle:processApplication</li></ul>
bpmnx:PROCESS_COMPLETED	An instance of a process has completed
bpmnx:PROCESS_TERMINATED	An instance of a process has terminated
bpmnx:PROCESS_DELETED	An instance of a process has been deleted
bpmnx:PROCESS_FAILED	An instance of a process has failed
bpmnx:ACTIVITY_READY	An activity is ready. The models supplied include: <ul style="list-style-type: none"><li>• type="bpmn:userTask"</li><li>• type="bpmn:process"</li><li>• type="wle:processApplication"</li></ul>
bpmnx:ACTIVITY_ACTIVE	An activity is active. The models supplied include: <ul style="list-style-type: none"><li>• type = "bpmn:userTask"</li><li>• type = "bpmn:process"</li><li>• type = "wle:processApplication"</li></ul>
bpmnx:ACTIVITY_COMPLETED	An activity has completed
bpmnx:ACTIVITY_TERMINATED	An activity has terminated
bpmnx:ACTIVITY_FAILED	An activity has failed
bpmnx:ACTIVITY_RESOURCE_ASSIGNED	A user or group has been associated with an activity. The models supplied include: <ul style="list-style-type: none"><li>• type="bpmn:userTask"</li><li>• type="bpmn:process"</li><li>• type="wle:processApplication"</li></ul>
ACTIVITY_LOOP_CONDITION_TRUE	

ACTIVITY_LOOP_CONDITION_FALSE	
ACTIVITY_PARALLEL_INSTANCES_STARTED	
bpmn:EVENT_CAUGHT	<p>The models supplied include:</p> <ul style="list-style-type: none"> <li>• mon:type = "bpmn:startEvent"</li> <li>• mon:type = "bpmn:process"</li> <li>• mon:type = "wle:processApplication"</li> </ul>
bpmn:EVENT_EXPECTED	
bpmn:EVENT_THROWN	
bpmn:GATEWAY_ACTIVATED	
bpmn:GATEWAY_COMPLETED	

The model type can be:

bpmn:process	
bpmn:intermediateThrowEvent	
wle:processApplication	

A common pattern will be to create a Monitoring Context when a bpmnx:PROCESS\_STARTED event is detected and close the Monitoring Context when a bpmnx:PROCESS\_COMPLETED event is detected.

The XML Schema files that represent these events can be found in the InfoCenter at:

[http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.admin.doc/topics/rmon\\_eventschemaext.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.admin.doc/topics/rmon_eventschemaext.html)

These three files should be built, saved and added to the Monitor projects.

## Common patterns

A Process has started

mon:monitorEvent/mon:eventPointData/mon:kind = bpmx:PROCESS\_STARTED

## ***Examining events issued from BPM***

When a BPM process runs, it generates events that are sent to Business Monitor. From time to time, we may wish to examine these events and ensure that the data we expected to be sent arrived. Alternatively, we may simply be interested in seeing what is passing.

One way to achieve this is to switch on Event Recording. This can be done through the IBM WAS Admin Console in the

Applications > Monitor Services > Recorded Event Management > Event Recording section. Within there we can enable or disable event recording:

Event Recording				
Specify whether to record the events that are sent to the specified monitor models.				
<input checked="" type="checkbox"/> Preferences				
<input type="button" value="Start Event Recording"/> <input type="button" value="Stop Event Recording"/>				
Select	CEI Host Name	CEI Event Service	Model	Status
<input type="checkbox"/>	localhost	/Cell:win7-x64Node01Cell/Node:win7-x64Node01/Server:server1	ABC_Ingest2014-02-14T13:20:24 BUSMONMonitoringModel2014-01-27T16:11:49 BusinessMonitorSamplesMonitoringModel2014-03-16T13:24:41 MyModel12014-03-08T08:54:54 TestModel2014-01-28T14:37:59	Enabled
Total 1				

Once enabled and new events generated, we can review those events from

Applications > Monitor Services > Recorded Event Management > Event Management

Event Management				
Use this page to manage recorded events.				
<input checked="" type="checkbox"/> Preferences				
<input type="button" value="Delete"/> <input type="button" value="Delete All"/> <input type="button" value="Export"/> <input type="button" value="Export All"/> <input type="button" value="Import Events"/>				
Select	Event	Time Recorded		
<input type="checkbox"/>	6	2014-03-19T13:35:16.223+00:00		
<input type="checkbox"/>	5	2014-03-19T13:30:26.598+00:00		
<input type="checkbox"/>	4	2014-03-19T13:30:26.462+00:00		
<input type="checkbox"/>	3	2014-03-19T13:30:26.331+00:00		
<input type="checkbox"/>	2	2014-03-19T13:30:26.190+00:00		
<input type="checkbox"/>	1	2014-03-19T13:30:26.001+00:00		
Total 6				

Clicking on an event will show its XML content:

Event Management > View Event XML	
Use this page to view the event XML data.	
<input type="button" value="Expand All"/>	<input type="button" value="Collapse All"/>
<pre>&lt;CommonBaseEvent creationTime="2014-03-19T13:30:26.329Z" extensionName="com.ibm.wbmonitor.EventEmitter" globalInstanceId="CE16CD9DC9EBBF02FA1E3AF6AA1913890" sequenceNumber="988903408753975" version="1.0.1" &gt; &lt;sourceComponentId component="WPS4Platform 8.0.0.7 [ND 8.0.0.7 c071329.03][WBM 8.0.1.0 20121102-1733]" componentIdType="ProductName" executionEnvironment="Windows 7[amd64]#6.1" instanceId="win7-x64Node01Cell\win7-x64Node01\server1" location="win7-x64" locationType="Hostname" processId="13932" subComponent="APT" threadId="SIBJMSRAThreadPool : 0" componentType="http://www.ibm.com/namespaces/autonomic/Workflow_Engine" /&gt; <input type="checkbox"/> &lt;situation categoryName="ReportSituation" &gt;   &lt;situationType xsi:type="ReportSituation" reasoningScope="EXTERNAL" reportCategory="ecode" /&gt; &lt;/situation&gt; <input type="checkbox"/> &lt;mon:monitorEvent mon:id="W1b6ad55298ad441792162189" &gt;   <input type="checkbox"/> &lt;mon:eventPointData&gt;     &lt;mon:kind mon:version="2010-11-11" &gt; bpmnx:ACTIVITY_READY&lt;/mon:kind&gt;     &lt;mon:time mon:of="occurrence" &gt; 2014-03-19T08:30:24.988-05:00&lt;/mon:time&gt;     &lt;ibm:sequenceId&gt; 0000000006&lt;/ibm:sequenceId&gt;     <input type="checkbox"/> &lt;mon:model mon:type="bpmn:userTask" mon:id="bpnid:c546ae322ac77f8a-79f454f0:1445fc8481a-7ff2" mon:version="2064.847490d3-f4b6-477a-8a64-8762a69fe579" &gt;       &lt;mon:name&gt; Receive Claim&lt;/mon:name&gt;       &lt;mon:instance mon:id="3" /&gt;     &lt;/mon:model&gt;     <input type="checkbox"/> &lt;mon:model mon:type="bpmn:process" mon:id="abc7872b-fd44-4c7e-95c1-5455c1e417b0" mon:version="2064.847490d3-f4b6-477a-8a64-8762a69fe579" &gt;       &lt;mon:name&gt; Insurance Claim&lt;/mon:name&gt;       <input type="checkbox"/> &lt;mon:instance mon:id="808" /&gt;</pre>	

## Capturing additional data from BPM

The IBM generated monitor models are a great start but there are times when we might wish to capture additional information. Here we will document some recipes for additional items that may be of interest.

## Capturing the Task ID of a task

When a task is created in the process, we may wish to know the Task ID for the task in the monitor data. This can be achieved by adding a new metric called "Task ID" in the "Steps" monitoring context. The data type of the metric should be "String" and the expression used to populate it should be:

```
if
(fn:exists(ACTIVITY_RESOURCE_ASSIGNED_Event/EventPointData/mon:model[1]/mon:instance/wle:taskInstanc
eId)) then
 ACTIVITY_RESOURCE_ASSIGNED_Event/EventPointData/mon:model[1]/mon:instance/wle:taskInstanceId
else
 Task_ID
```

## Capturing the Process ID of a process

When a process instance is created, we may wish to know the Process ID of that process in the monitor data. This can be achieved by adding a new metric called "Process ID" in the top of the monitoring context for the generated model. The data type of the metric should be "String" and the expression used to populate it should be:

```
PROCESS_STARTED_Event/EventPointData/mon:model[1]/mon:instance/@mon:id
```

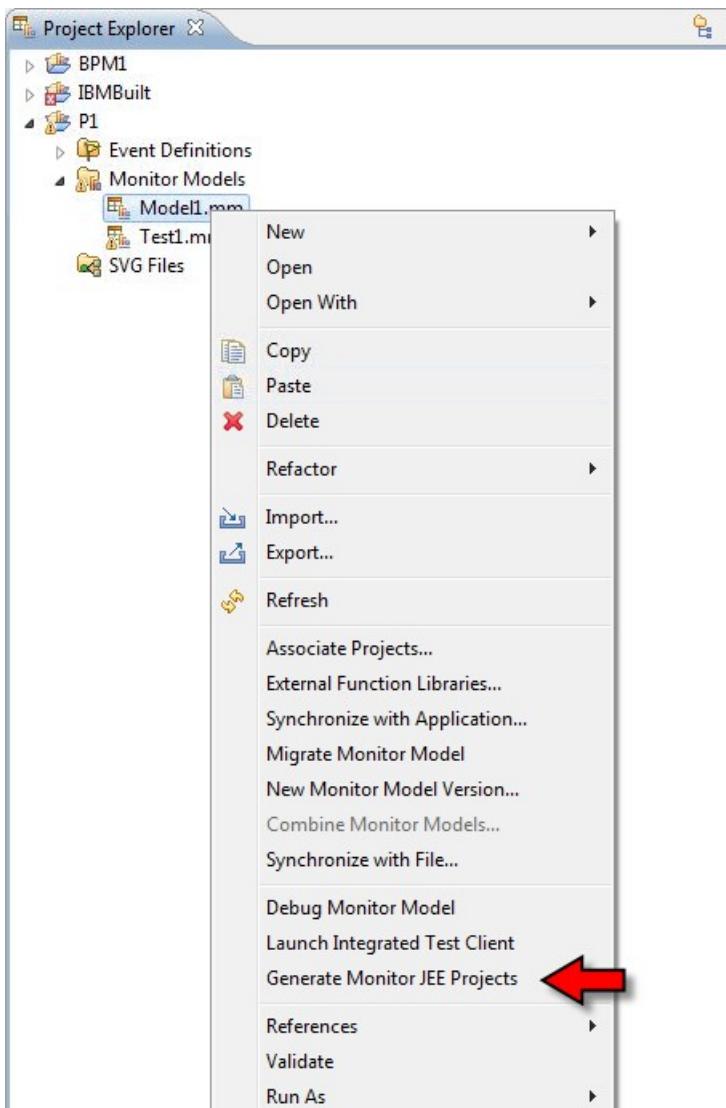
## Capturing the Process Name of a process

When a process instance is created, we may wish to know the Process Name of that process in the monitor data. This can be achieved by adding a new metric called "Process Name" in the top of the monitoring context for the generated model. The data type of the metric should be "String" and the expression used to populate it should be:

```
PROCESS_STARTED_Event/EventPointData/mon:model[1]/mon:name
```

## Compiling Monitor Models

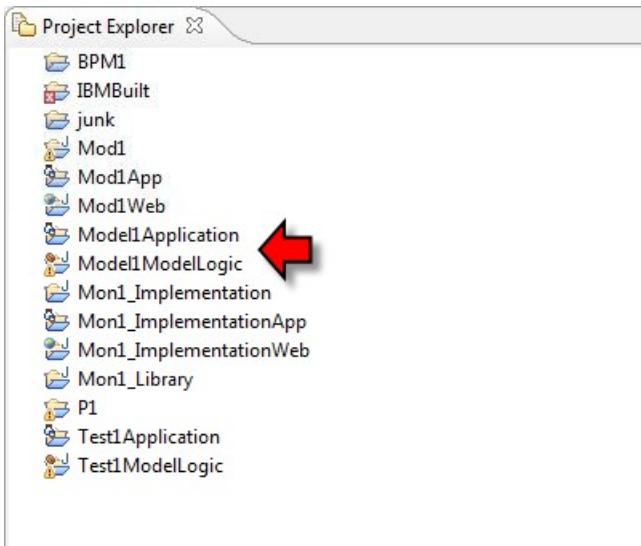
Once a monitor model has been edited in the Monitor Model editor, the EAR file that represents the deployable unit of the model must be built. Right-clicking on the Monitor Model and selecting Generate Monitor JEE Projects will achieve this task.



Once completed, the application can be deployed on the Monitor server.

In rare cases, the compilation can get confused in ID. A solution to this is to delete the generate code and then re-compile from scratch. To achieve this, switch to the Resource Eclipse perspective and delete the two projects which are called:

- <ModelName>Application
- <ModelName>ModelLogic



When the monitor model is changed, this recipe needs to be repeated in order for the changes to take effect. In principle, it should be possible to simply re-deploy the new EAR with the existing one still published but practice has shown this to be extremely problematic. My strongest recommendation is to:

1. Un-install the existing EAR model
2. Generate a new JEE EAR from the model
3. Re-deploy the new EAR model

If after making a change to your model you find that there is no obvious runtime change, carefully check that you have rebuilt and republished the EAR model. It is not an uncommon error to make a change to the model and neglect to rebuild and republish.

It takes a few minutes to un-deploy, build and re-publish. You will need a machine with sufficient resources to make this a workable recipe.

It has also been noticed that even though the deployment of the new model appears completed from the ID Servers view, work is still on-going as shown in the console log. When I publish new models, I keep an eye on the console log to ensure all work has completed before progressing with new tests. Specifically, I continuously see the EAR being published, it saying it is started ... a minute passes ... it says it is stopped and then finally it says it is started again.

## ***Versioning Monitor Models***

If we wish to create a new version of a Monitor Model while maintaining the data already collected, we do this by we update the Model Version timestamp to be a later value.

**▼ Monitor Details**

Edit the details of the model. The timestamp is required to identify the version of the model.

ID:	<input type="text" value="MonitorMonitoringModel"/> 
Name:	<input type="text" value="MonitorMonitoringModel"/>
Description:	<input type="text"/>
Time Stamp (UTC):	<input type="text" value="2014-01-05T01:18:02Z"/>  

## Building Monitor models using the Model Editor

The Monitor Model editor is heavily dependent on XPath expressions.

It uses a number of namespaces. The common prefixes for these namespaces are show below as well as their expanded values. The expanded values are needed when building models that have data types of QName.

Prefix	Namespace
mon	http://www.ibm.com/xmlns/prod/websphere/monitoring/7.5
wle	http://www.ibm.com/xmlns/prod/websphere/lombardi/7.5
bpmn	http://schema.omg.org/spec/BPMN/2.0
bpmnx	http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring

See also:

- Building a Monitor Model for BPMN events

## Monitor Process Start event

When a Process instance is started, an event is generated. The mon:kind has the value bpmnx:PROCESS\_STARTED. An example of the event is shown below:

```
<mon:monitorEvent ...>
 <mon:eventPointData>
 <mon:kind mon:version="2010-11-11">bpmnx:PROCESS_STARTED</mon:kind>
 ...
 <mon:model mon:type="wle:processApplication">
 <mon:name>ProcessAppName</mon:name>
 ...
 </mon:model>
```

**Something is wrong here. Using just this algorithm below, we would capture ALL Process Starts not just the ones we are interested in modeling.**

A suitable Monitor Model inbound event definition would have:

Event Type Details
<ul style="list-style-type: none"> <li>ID: EventPointData</li> <li>type: mon:eventPointData</li> <li>Path: cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData</li> </ul>
Filter Condition
xs:string(PROCESS_STARTED_Event/EventPointData/mon:kind) = '{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS_STARTED'
Correlation Expression
PROCESS_STARTED_Event/EventPointData/mon:correlation/wle:starting-process-instance = piid

Here we assume that the key for the MC is called piid and is configured:

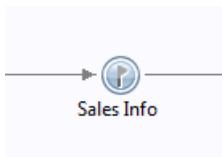
Key Value Expressions
PROCESS_STARTED_Event/EventPointData/mon:correlation/wle:starting-process-instance

## Correlating BPMN process events

mon:monitorEvent/mon:eventPointData/mon:correlation/wle:starting-process-instance

## Monitor Tracking point data

When a tracking point is reached in a BPD, a bpmnx:EVENT\_THROWN is generated.



The body of the event contains the name of the tracking group plus the values of the tracking group parts.

Here is an example of the event edited for clarity:

```

<mon:monitorEvent>
 <mon:eventPointData>
 <mon:kind mon:version="2010-11-11">bpmnx:EVENT_THROWN</mon:kind>
 <mon:time mon:of="occurrence">2011-07-03T11:40:52.778-05:00</mon:time>
 <ibm:sequenceId>000000007</ibm:sequenceId>
 <mon:model mon:type="bpmn:intermediateThrowEvent">
 mon:id="bpdid:56d35..."
 mon:version="2064.d9d1e..."
 <mon:name>Untitled1</mon:name>
 <mon:instance mon:id="4"></mon:instance>
 </mon:model>
 ...
 </mon:eventPointData>
 <mon:applicationData>
 <wle:tracking-point wle:time="2011-07-03T11:40:52.779-05:00"
 wle:name="Untitled1"
 wle:id="ded130..."
 wle:version="2064.d9d1e3..."
 wle:groupName="sale2"
 </wle:tracking-point>
 </mon:applicationData>
</mon:monitorEvent>

```

```

wle:groupId="18e915..."
wle:groupVersion="2064.d9d1e3..."
<wle:tracked-field
 wle:name="usState"
 wle:id="09d4da..."
 wle:type="xs:string">TX</wle:tracked-field>
<wle:tracked-field
 wle:name="saleDate"
 wle:id="08365..."
 wle:type="xs:dateTime">2011-07-22T05:00:00.00Z</wle:tracked-field>
</wle:tracking-point>
</mon:applicationData>
</mon:monitorEvent>

```

The XPath to get a named tracking data field is:

```
mon:monitorEvent/mon:applicationData/wle:tracking-point/wle:tracked-field[@wle:name='???']
```

Here is a suggested set of inbound event details.

Event Type Details
<ul style="list-style-type: none"> <li>• ID: EventPointData</li> <li>• Type: mon:eventPointData</li> <li>• Path: cbeCommonBaseEvent/mon:monitorEvent/mon:eventPointData</li>   <li>• ID: TrackingPoint</li> <li>• Type: TrackingPoint</li> <li>• Path: cbeCommonBaseEvent/mon:monitorEvent/mon:applicationData/wle:tracking-point</li> </ul>
Filter Condition
<pre> xs:string(TRACKING_GROUP_name/EventPointData/mon:kind) = '{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}EVENT_THROWN' and xs:string(TRACKING_GROUP_name/EventPointData/mon:model[1]/@mon:type) = '{http://schema.omg.org/spec/BPMN/2.0}intermediateThrowEvent' and TRACKING_GROUP_Sale2/TrackingPoint/@wle:groupName = 'name' </pre>
Correlation Expression
TRACKING_GROUP_name/EventPointData/mon:correlation/wle:starting-process-instance = piid

The way to read this inbound event is that it is the tracking group we are looking for if all of the following are true:

- The kind of event is bpmnx:EVENT\_THROWN
- The sub-type is bpmn:intermediateThrowEvent
- The tracking event is for our named tracking group

A metric can then be populated from the Inbound event with the expression:

Metric Value Expressions
TRACKING_GROUP_name/TrackingPoint/wle:tracked-field[@wle:name = 'field']

Take care not to populate fields when there is no data present. One solution is to create a trigger from the inbound event and associate a condition with that trigger so that it only fires if the field is

present. We can then use the trigger as the indicator that a metric should be updated.

See also:

- Tracking Intermediate Event

## Closing the Monitoring Context

When a BPD completes, it should close the associated monitoring context. The completion of the process is detected by an event having mon:kind set to bpmnx:PROCESS\_COMPLETED. An example fragment of the data is shown next:

```
<mon:monitorEvent ...>
 <mon:eventPointData>
 <mon:kind mon:version="2010-11-11">bpmnx:PROCESS_COMPLETED</mon:kind>
 ...
 </mon:eventPointData>
</mon:monitorEvent>
```

Example monitoring definitions:

Event Type Details
<ul style="list-style-type: none"><li>• ID: EventPointData</li><li>• type: mon:eventPointData</li><li>• Path: cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData</li></ul>
Filter Condition
xs:string(PROCESS_COMPLETED_Event/EventPointData/mon:kind) = '{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS_COMPLETED'
Correlation Expression
PROCESS_COMPLETED_Event/EventPointData/mon:correlation/wle:starting-process-instance = piid

A Trigger is required to close the context:

Trigger Sources
Source: PROCESS_COMPLETED Event
Trigger Condition
N/A

## Removing monitor 'Problems' view errors and warnings

By default, the Monitor Model editor shows warnings in the Problems view. These messages are not needed.

Monitoring Flow Properties Problems Servers Server Logs Console				
0 errors, 141 warnings, 1 other (Filter matched 101 of 142 items)				
Description	Resource	Path	Location	Type
<b>Warnings (100 of 141 items)</b>				
CWMMV0743W: This inbound event can both...	Model1.mm	/P1	line 5	Monitor Proble...
CWSKA8010E: The Output import has no bind...	Output.import	/Mod1	line 2	SCDL Warning...
The field MonitorContext.CREATION_TIME is n...	MonitorConte...	/Model1ModelLogi...	line 39	Java Problem
The field MonitorContext.PARENT_MCIID is n...	MonitorConte...	/Model1ModelLogi...	line 38	Java Problem
The field MonitorContext.ROUTING_PARTITION...	MonitorConte...	/Model1ModelLogi...	line 44	Java Problem
The field MonitorContext.TERMINATION_TIM...	MonitorConte...	/Model1ModelLogi...	line 40	Java Problem
The field Svc.Xct is deprecated	BaseBeanMod...	/Model1ModelLogi...	line 75	Java Problem

When building Monitor models, a series of superfluous warning messages are logged in the Problems view. These messages can distract from more significant issues. These warnings can be disabled in ID through:

Window > Preferences > Java > Compiler > Errors/Warnings

From there, switch the relevant types of warning from "Warning" to "Ignore" and the messages/warnings are removed.

- Code style
  - Non-static access to static member
- Potential programming problems
  - Serializable class without serialVersionUID
- Deprecated and restricted API
  - Deprecated API
- Unnecessary code
  - Local variable is never read
  - Unused local or private member
  - Unused import
- Annotations
  - Unused '@SuppressWarnings' annotations

The generated J2EE projects can also cause warnings to be written. These can also be switched off. In ID go to

Window > Preferences > Validation

From there, switch off (uncheck) the entries next to:

- EJB Validator

## Debugging Monitor Models

### Recorded Events Management

Monitor has the capability to record events that have previously been presented to it for later examination. This is enabled or disabled from within the WAS admin console. The default is

disabled.

The screenshot shows the Integrated Solutions Console interface. The left sidebar has a tree view with 'Applications' expanded, showing 'Monitor Services' and its sub-options: 'Monitor Action Services', 'Recorded Events Management' (which is expanded), and 'Play Back Events'. A red arrow points to the 'Enable/Disable Events Record' link under 'Recorded Events Management'. The main content area is titled 'Enable/Disable Events Record' with the sub-section 'Enable/Disable Events Record'. It contains a table with one row, showing 'localhost' as the CEI Host Name, the CEI Event Service path as '/Cell:win7-x64Node01Cell/Node:win7-x64Node01/Server:server1', the Model as 'Model12011-06-29T16:54:38', and the Status as 'Enabled'.

Once enabled, events received can also be seen in the admin console:

The screenshot shows the Integrated Solutions Console interface. The left sidebar has a tree view with 'Applications' expanded, showing 'Monitor Services' and its sub-options: 'Monitor Action Services', 'Recorded Events Management' (which is expanded), and 'Play Back Events'. A red arrow points to the 'Events Management' link under 'Recorded Events Management'. The main content area is titled 'Events Management' with the sub-section 'Events Management'. It contains a table with three rows, each representing an event. The columns are 'Select', 'Event' (containing values 3, 2, and 1), and 'Time Recorded' (containing dates and times: 2011-07-01T20:29:40.349+00:00, 2011-07-01T20:29:39.443+00:00, and 2011-07-01T20:29:38.412+00:00). A total of 3 events are listed.

From here individual events can be viewed, deleted or exported.

## Using the Monitor Model debugger

The Monitor Model debugger is a capability of IID that allows Monitor Models to have break points associated with them. When events arrive at the Monitor Server, these breakpoints are honored and a debugging perspective is shown that allows the programmer to see the event, conditions and other items associated with the processing of that event.

Two applications must be installed in the server in order to use the Monitor Model debugger. These are:

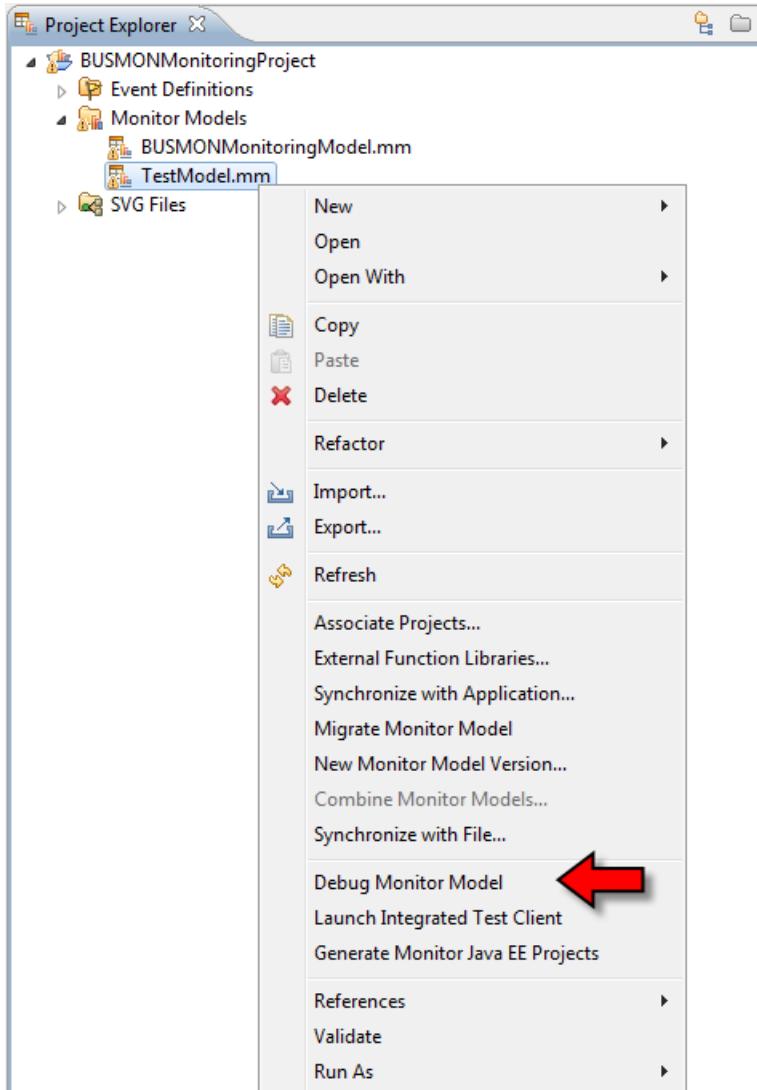
- com.ibm.wbimonitor.debug.server.application
- com.ibm.wbimonitor.ice.machine.application

Both can be found in the <WAS>/installableApps.wbm/debugger folder.

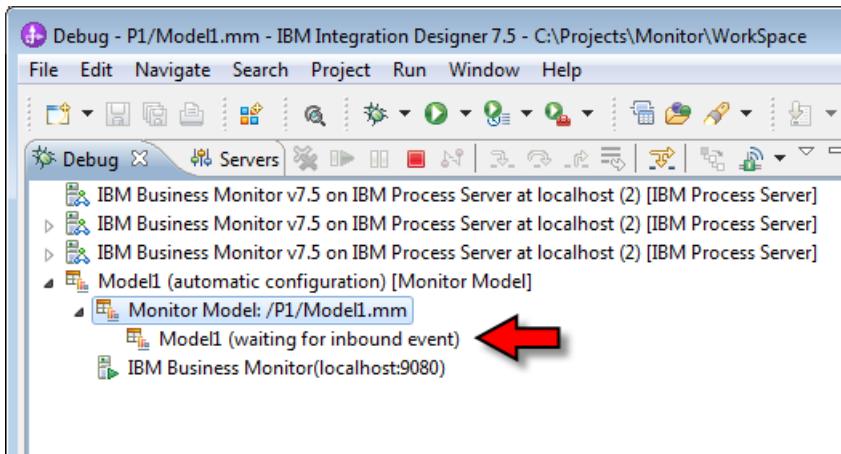
The first provides an application called `IBM_WBM_DBG_SERVICES` the second provides an application called `IBM_WBM_IVM`. Before running debugging on a new server, validate that these applications are installed and running.

You can administer the following resources:		
<input type="checkbox"/>	<a href="#">IBM_WBM_ACTIONSERVICES</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_DATA_SERVICES</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_DBG_SERVICES</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_EMITTER_SERVICES</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_IVM</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_MM_DEPLOYMENT_SERVICE</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_MOBILE_DASHBOARD</a>	
<input type="checkbox"/>	<a href="#">IBM_WBM_MSS_SERVICE</a>	
Total 8 Filtered total: 8		

To launch the debugger, right click on the Monitor model and select "Debug Monitor Model":

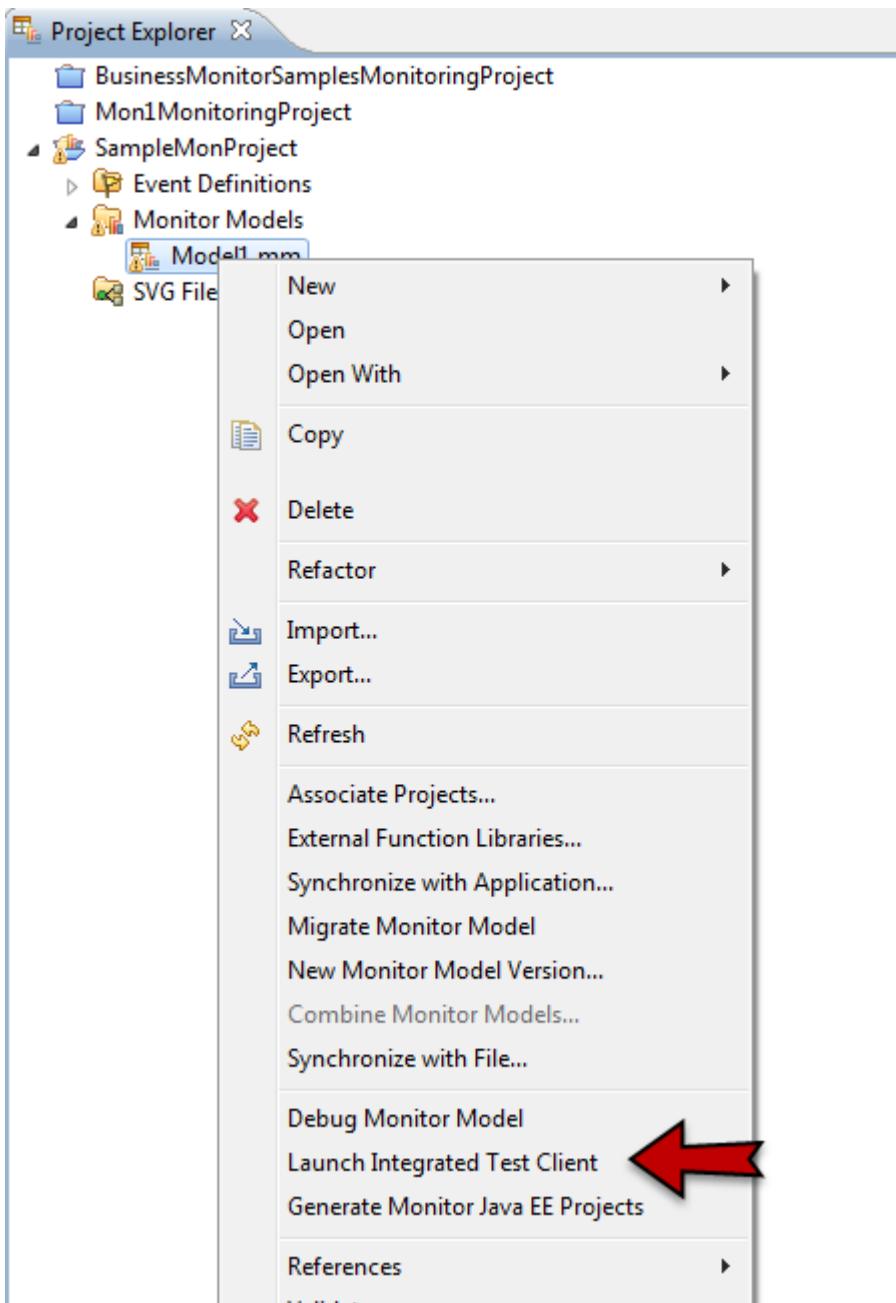


We know that the monitor debugger is ready when the following is shown:

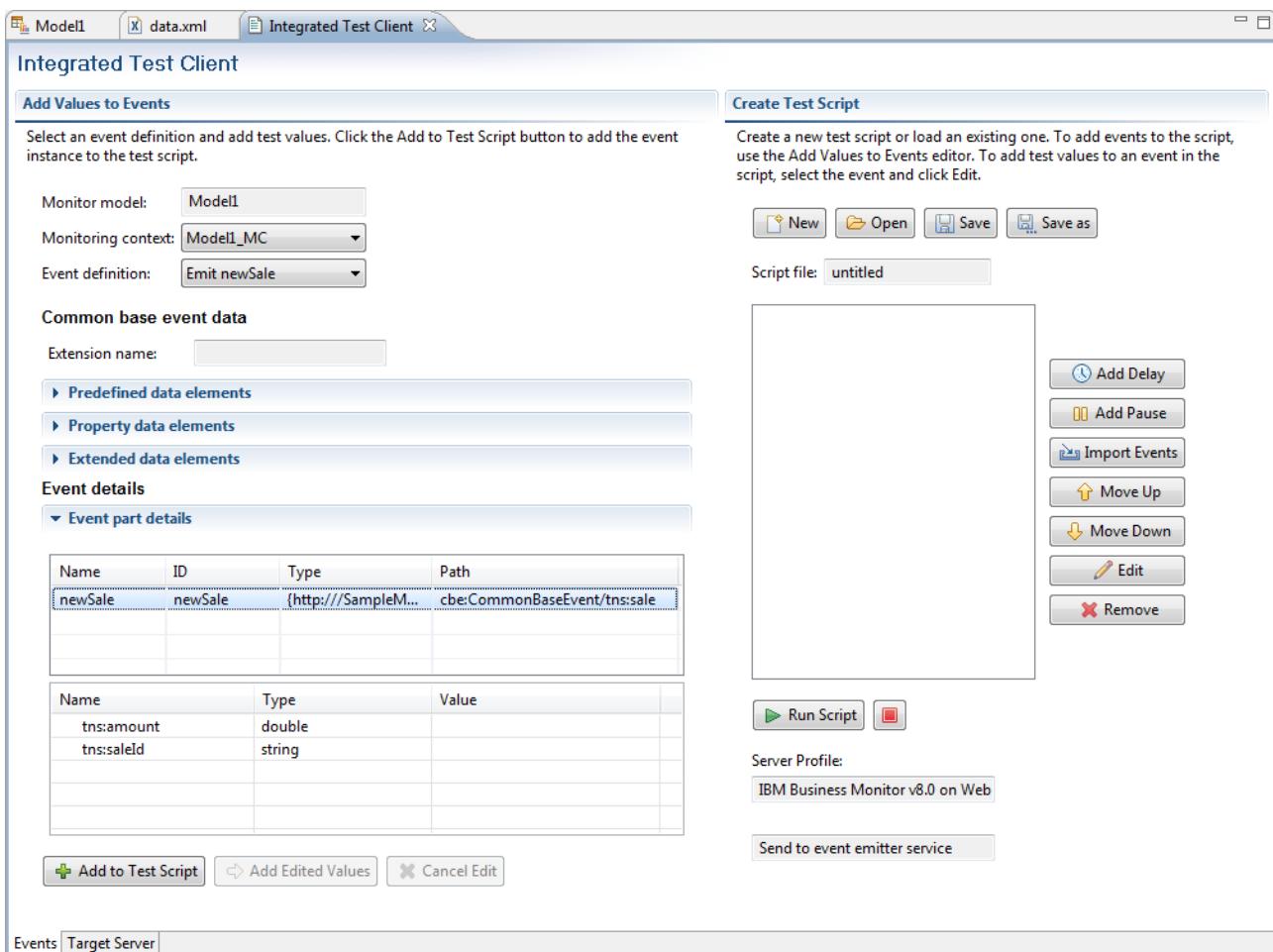


## ***Integrated Test Client***

Another diagnostic tool at our disposal is the Integrated Test Client. This is launched from the context menu of a Monitor Model:



Once launched, it looks as follows:



The high level model of its usage is that you add a series of events that are to be published. An individual event is configured in the left panel. Once defined, the "Add to Test Script" button can be pressed which will add it to the set of tests to be executed as shown on the right. Once the tests have been defined, clicking the "Run Script" button will actually run the tests. The set of tests can be saved as a group.

## Uninstalling Monitor Apps

If we wish to uninstall a Monitor Application, we may use the standard WAS based techniques. To see what models are installed, visit Applications > Monitor Models.

This will then show you the applications corresponding to each of the models:

Select	Model	Version	Deployment	Application	Data Security	Status
<input type="checkbox"/>	BUSMONMonitoringModel	2014-01-27T16:11:49	OK	BUSMONMonitoringModelApplication	Model associated	

From there you can visit Applications > Application Types > WebSphere enterprise applications. The associated application can be selected and uninstalled.

# Using Monitor REST APIs

One of the REST interfaces exposed by Monitor is the ability to send in data that, when received, is considered to be an Event. This page provides notes on that topic.

REST requests are simple HTTP requests that carry payload and return response data. REST requests can be sent from any language capable of sending an HTTP request. This includes Java and JavaScript.

## ***The Monitor REST Event catcher***

When a client sends a REST Event to Monitor, monitor must "catch" that event and process it. We will call the application that receives the event the "catcher". The catcher is a Java EE application that when it receives the data, forwards that through CEI for normal Monitor processing. The catcher is listening on the URL at the context root of /rest/bpm/events.

## ***The Monitor REST request***

The request should be sent to /rest/bpm/events. It must have a MIME type of "text/xml". As of 8.0.1, there appears to be a need to send it to https.

A particularly good free tool for sending REST requests to monitor is "postman". To use this with monitor, the following screen shot shows some sample settings;

The screenshot shows the Postman REST client interface. The request details are as follows:

- Method: POST
- URL: https://localhost:9448/rest/bpm/events
- Content-Type: text/xml
- Body (raw XML):

```
1 <tns:sale xmlns:tns="http://SampleMonProject/data.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://SampleMonProject/data.xsd data.xsd ">
2 <tns:saleId>X2</tns:saleId>
3 <tns:amount>99.9</tns:amount>
4 <tns:usState>tns:usState</tns:usState>
5 <tns:widgetType>tns:widgetType</tns:widgetType>
6 </tns:sale>
```

- Headers (1):

Header	Value
Content-Type	text/xml

- Send button

## ***The Monitor REST response***

The response back from Monitor following a REST request is a JSON string. It contains the following fields:

- Event Emission - String - Unknown meaning
- Request ContentType - String - The encoding of the request message
- Event IDs - Array of Strings - Event IDs

On an error, the response contains:

- Emission Completion Status - Integer
- Error - String - Description of the problem

## ***Sending a Monitor REST request from Java***

In Java, the core class of interest to us is called URL. This class provides the ability to name an HTTP target, send data and receive a response. When sending a secure request, the HTTPS protocol should be used.

```
String data = "<data to send as the event>";
URL url = new URL("http://localhost:9080/rest/bpm/events");
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
httpURLConnection.setRequestMethod("POST");
httpURLConnection.setDoOutput(true);
httpURLConnection.setRequestProperty("content-type", "text/xml");

OutputStreamWriter wr = new
OutputStreamWriter(httpURLConnection.getOutputStream());
wr.write(data);
wr.flush();
InputStreamReader inReader = new InputStreamReader(httpURLConnection.getInputStream());
BufferedReader br = new BufferedReader(inReader);
String line = br.readLine();
while (line != null) {
 System.out.println("Line is: " + line);
 line = br.readLine();
}
```

## **Monitor REST Security**

When sending a request to a secure Monitor, authentication must be performed. Adding the following before connecting to the server will achieve this:

```
String wasUserName = "<WAS Userid>";
String wasPassword = "<WAS Userid Password>";
String authorization = "Basic " + Base64.encode(new String(wasUserName + ":" +
wasPassword).getBytes());
httpURLConnection.setRequestProperty("Authorization", authorization);
```

If the caller is running in an authenticated context, then an LTPA token can be passed. Adding the following before connecting to the server will achieve this:

```
Set s = WSSubject.getCallerSubject().getPublicCredentials(WSCredential.class);
WSCredential creds = (WSCredential) s.iterator().next();
byte token[] = creds.getCredentialToken();
httpURLConnection.setRequestProperty("cookie", "LtpaToken=" + Base64.encode(token));
```

## Notes

- **Do not include the <XML> header that is commonly found at the start of XML documents. Monitor appears to choke on this.**

When sending a request over HTTPS, certificates may need to be exchanged. One of the easiest ways to set this up is the following recipe.

First, obtain the Monitor Server certificate using the WAS admin console:

```
Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore >
Personal certificates
```

To insert the

```
keytool -import -file ProcCtr.pem -alias ProcCtr -keystore cacerts
```

The initial password for the Java keystore is "changeit"

## ***REST API Components***

### **Getting models**

This API returns a list of monitor models known to Monitor with their version information:

```
GET /rest/bpm/monitor/models
```

Example:

```
[
 {
 "Model ID": "MonitorMonitoringModel",
 "Versions": [
 {
 "Model Display Name": "MonitorMonitoringModel",
 "Version": 20140105011802
 }
]
 }
]
```

### **Getting monitoring contexts**

```
GET /rest/bpm/monitor/<ModelID>/mcs
```

Example:

```
{
 "Model ID": "MonitorMonitoringModel",
 "MC Array": [
 {
 "MCID": "Handle_Order",
 "MC Display Name": "Handle Order",
 "Cube Name": "MONITORMONITORINGMODEL_HANDLE_ORDER",
 "Child MCs": [],
 "Diagram Exists": false
 }
]
}
```

### **Get Metrics**

```
GET /rest/bpm/monitor/<ModelID>/mcs/<MC ID>/metrics
```

Example:

```
{
 "MCID": "Handle_Order",
 "Model ID": "MonitorMonitoringModel",
 "Metric Array": [
 {
 "Metric Sortable": "false",
 "Metric Type": "DECIMAL",
 "Metric Name": "amount",
 }
]
}
```

```

 "Metric ID": "bmon_f_amount"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "STRING",
 "Metric Name": "Aux Starting Process Instance ID",
 "Metric ID": "Aux_StartProcess_Instance_ID"
 },
 {
 "Metric Sortable": "true",
 "Metric Type": "BOOLEAN",
 "Metric Name": "COMPLETED",
 "Metric ID": "COMPLETED"
 },
 {
 "Metric Sortable": "true",
 "Metric Type": "DATETIME",
 "Metric Name": "CreationTime",
 "Metric ID": "CreationTime"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "DATETIME",
 "Metric Name": "End Time",
 "Metric ID": "bmon_EndTime"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "STRING",
 "Metric Name": "Handle Order Instance ID",
 "Metric ID": "Handle_Order_Instance_ID"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "STRING",
 "Metric Name": "orderType",
 "Metric ID": "bmon_f_orderType"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "STRING",
 "Metric Name": "salesRep",
 "Metric ID": "bmon_f_salesRep"
 },
 {
 "Metric Sortable": "false",
 "Metric Type": "DATETIME",
 "Metric Name": "Start Time",
 "Metric ID": "bmon_StartTime"
 },
 {
 "Metric Sortable": "true",
 "Metric Type": "DATETIME",
 "Metric Name": "TerminationTime",
 "Metric ID": "TerminationTime"
 }
]
}

```

## Get instances

GET /rest/bpm/monitor/<ModelID>/mcs/<MC ID>/metrics

Example:

```
{
 "Record Count": 2,
 "MCID": "Handle_Order",
 "Page Size": 10,
 "Model ID": "MonitorMonitoringModel",
 "FGSSecurityFilterApplied": false,
 "Metric ID Array": [
 "bmon_f_amount",
 "bmon_f_amount Localized",
 "Aux_StartProcess_Instance_ID",
 "COMPLETED",
 ...
]
}
```

```

 "CreationTime",
 "CreationTime Localized",
 "bmon_EndTime",
 "bmon_EndTime Localized",
 "Handle_Order_Instance_ID",
 "bmon_f_orderType",
 "bmon_f_salesRep",
 "bmon_StartTime",
 "bmon_StartTime Localized",
 "TerminationTime",
 "TerminationTime Localized"
],
"Instance Data": [
{
 "Instance ID": 1,
 "Version": 20140105011802,
 "Metric Data": [
 100,
 "100.00",
 "8f5aff20-7b1f-4f6b-b30b-fcfea5fef005.2064.f35f2b68-0c54-46e8-bfd4-e07d53a82db.561",
 true,
 "2014-01-05T07:28:05",
 "January 5, 2014 1:28:05 AM",
 "2014-01-05T07:28:24",
 "January 5, 2014 1:28:24 AM",
 "8f5aff20-7b1f-4f6b-b30b-fcfea5fef005.2064.f35f2b68-0c54-46e8-bfd4-e07d53a82db.561",
 "Red",
 "cadmin",
 "2014-01-05T07:28:05",
 "January 5, 2014 1:28:05 AM",
 "2014-01-05T07:28:24",
 "January 5, 2014 1:28:24 AM"
],
 "Children": []
},
{
 "Instance ID": 2,
 "Version": 20140105011802,
 "Metric Data": [
 200,
 "200.00",
 "8f5aff20-7b1f-4f6b-b30b-fcfea5fef005.2064.f35f2b68-0c54-46e8-bfd4-e07d53a82db.562",
 true,
 "2014-01-05T07:28:31",
 "January 5, 2014 1:28:31 AM",
 "2014-01-05T07:28:43",
 "January 5, 2014 1:28:43 AM",
 "8f5aff20-7b1f-4f6b-b30b-fcfea5fef005.2064.f35f2b68-0c54-46e8-bfd4-e07d53a82db.562",
 "Blue",
 "kolban",
 "2014-01-05T07:28:31",
 "January 5, 2014 1:28:31 AM",
 "2014-01-05T07:28:43",
 "January 5, 2014 1:28:43 AM"
],
 "Children": []
}
],
"Page Number": 1,
"DisplayFGSIndication": false
}

```

## Getting KPI Contexts

GET /rest/bpm/monitor/<ModelID>/version/<Version>/kcs

```
[
{
 "Model ID": "MonitorMonitoringModel",
 "KPI Context Name": "Handle Order",
 "Version": 20140105011802,
 "KPI Context ID": "Handle_Order",
 "Diagram Exists": true
}
]
```

## Getting KPI Lists

GET /rest/bpm/monitor/<ModelID>/version/<Version>/kpis

```
[
 {
 "Aggregated Metric Name": "Amount",
 "KPI Description": "",
 "KPI Origin": "runtime",
 "KPI Display Name": "My KPI",
 "Enable KPI Prediction": false,
 "KPI Context ID": null,
 "Target Localized": null,
 "Aggregated Metric ID": "bmon_f_amount",
 "User ID": "uid=admin,o=defaultWIMFileBasedRealm",
 "KPI Calc Method": "aggregated",
 "Target": null,
 "Model ID": "MonitorMonitoringModel",
 "Aggregated Function": "avg",
 "KPI Data Type": "decimal",
 "Calculated KPI Expression": null,
 "KPI ID": "My_x0020_KPI",
 "Version": 20140105011802,
 "KPI Context Name": null,
 "Version Aggregation": "allVersions",
 "Enable KPI History": true,
 "View Access": "personal"
 }
]
```

## Getting KPI Value

GET /rest/bpm/monitor/<ModelID>/version/<Version>/kpis/value/<KPI ID>

This API retrieves the value of the KPI as well as its definition. The following is an example of what might be returned.

```
{
 "Aggregated Metric MC Name": "Handle Order",
 "Aggregated Metric Name": "Amount",
 "Fixed Period End Localized": null,
 "KPI Display Name": "My KPI",
 "FGSKpiHistoryTrackingEnabled": null,
 "Effective End Date": null,
 "FGSKpiPredictionsEnabled": null,
 "Rolling Period Quantity": null,
 "Fixed Period Start": null,
 "Model ID": "MonitorMonitoringModel",
 "Aggregated Function": "avg",
 "Rolling Period Duration": null,
 "KPI Range Type": "actualValue",
 "Format Currency": null,
 "Repeating Period Timezone": null,
 "Model Display Name": "MonitorMonitoringModel",
 "Target Localized": null,
 "Fixed Period Timezone": null,
 "Time Period Method": null,
 "Aggregated Metric Type": "DECIMAL",
 "Aggregated Metric MC ID": "Handle_Order",
 "Aggregated Metric ID": "bmon_f_amount",
 "KPI Value Localized": "150",
 "Target": null,
 "KPI Value": 150,
 "KPI ID": "My_x0020_KPI",
 "Version": 20140105011802,
 "KPI Context Name": null,
 "KPI History Defaults": {
 "History Display Ranges": false,
 "History Granularity": "daily",
 "History Rolling Period Quantity": null,
 "History Repeating Period Basis": "periodInProgress",
 "History Repeating Period Quantity": 2,
 "History Timezone": null,
 "History All Versions": false,
 }
}
```

```

 "History Time Range Method": "repeatingPeriod",
 "History Rolling Period Duration": null,
 "History Repeating Period Duration": "monthly",
 "History Time Range End": null,
 "History Time Range Start": null,
 "History Valid From": null,
 "History Display Target": false
},
"Repeating Period Duration": null,
"Time Period Metric ID": null,
"View Access": "personal",
"KPI Cache Override Interval": null,
"KPI Description": "",
"KPI History Granularity Options": [
{
 "KPI History Granularity Supported": [
 {
 "Granularity Value": "hourly"
 },
 {
 "Granularity Value": "daily"
 },
 {
 "Granularity Value": "weekly"
 },
 {
 "Granularity Value": "monthly"
 },
 {
 "Granularity Value": "quarterly"
 },
 {
 "Granularity Value": "yearly"
 }
],
 "KPI History Granularity Hours": 0,
 "KPI History Granularity Default": "hourly"
},
{
 "KPI History Granularity Supported": [
 {
 "Granularity Value": "hourly"
 },
 {
 "Granularity Value": "daily"
 },
 {
 "Granularity Value": "weekly"
 },
 {
 "Granularity Value": "monthly"
 },
 {
 "Granularity Value": "quarterly"
 },
 {
 "Granularity Value": "yearly"
 }
],
 "KPI History Granularity Hours": 48,
 "KPI History Granularity Default": "daily"
},
{
 "KPI History Granularity Supported": [
 {
 "Granularity Value": "daily"
 },
 {
 "Granularity Value": "weekly"
 },
 {
 "Granularity Value": "monthly"
 },
 {
 "Granularity Value": "quarterly"
 },
 {
 "Granularity Value": "yearly"
 }
]
}
]
}

```

```

 "Granularity Value": "yearly"
 }
],
"KPI History Granularity Hours": 336,
"KPI History Granularity Default": "daily"
},
{
"KPI History Granularity Supported": [
{
 "Granularity Value": "daily"
},
{
 "Granularity Value": "weekly"
},
{
 "Granularity Value": "monthly"
},
{
 "Granularity Value": "quarterly"
},
{
 "Granularity Value": "yearly"
}
],
"KPI History Granularity Hours": 2208,
"KPI History Granularity Default": "monthly"
},
{
"KPI History Granularity Supported": [
{
 "Granularity Value": "weekly"
},
{
 "Granularity Value": "monthly"
},
{
 "Granularity Value": "quarterly"
},
{
 "Granularity Value": "yearly"
}
],
"KPI History Granularity Hours": 8784,
"KPI History Granularity Default": "monthly"
}
},
"Effective End Date Localized": null,
"Fixed Period End": null,
"DisplayFGSIndication": true,
"Time Period Metric Name": null,
"KPI Data Type": "decimal",
"Calculated KPI Expression": null,
"Repeating Period Basis": null,
"KPI Range Array": [
{
 "KPI Range Color": "#73bfe5",
 "KPI Range Start Value": 0,
 "KPI Range Icon": null,
 "KPI Range End Value Localized": "100",
 "KPI Range Display Name": "low",
 "KPI Range ID": "kpiRangeId###0##1389452567044",
 "KPI Range End Value": 100,
 "KPI Range Start Value Localized": "0"
},
{
 "KPI Range Color": "#add9c1",
 "KPI Range Start Value": 100,
 "KPI Range Icon": null,
 "KPI Range End Value Localized": "150",
 "KPI Range Display Name": "medium",
 "KPI Range ID": "kpiRangeId##1##1389452573771",
 "KPI Range End Value": 150,
 "KPI Range Start Value Localized": "100"
},
{
 "KPI Range Color": "#feb76b",
 "KPI Range Start Value": 150,

```

```

 "KPI Range Icon": null,
 "KPI Range End Value Localized": "200",
 "KPI Range Display Name": "high",
 "KPI Range ID": "kpiRangeId##2##1389452583442",
 "KPI Range End Value": 200,
 "KPI Range Start Value Localized": "150"
 }
],
"Version Aggregation": "allVersions",
"Enable KPI History": true,
"History Include Predictions": false,
"KPI Origin": "runtime",
"Format Percentage": false,
"Default Prediction Model ID": null,
"KPI Metric Filter Array": [],
"Format Decimal Precision": null,
"KPI Context ID": null,
"Enable KPI Prediction": false,
"KPI Calc Method": "aggregated",
"User ID": "uid=admin,o=defaultWIMFileBasedRealm",
"Prediction Model Array": [],
"FGSSecurityFilterApplied": false
}

```

As we can see there is a bewildering number of properties returned. Here are some of the most important:

- "KPI Value" – The current value of the KPI.
- "KPI Range Array" – An array of objects defining the ranges that have been defined. Each object contains:
  - "KPI Range Color" – The color to show the range.
  - "KPI Range Start Value" – The start of the range.
  - "KPI Range End Value" – The end of the range.
  - "KPI Display Name" – A text display for this range.
  - ... others ...

## Creating new KPI definitions

In addition to viewing and listing KPI data, we also have the ability to dynamically create new KPI definitions.

```
POST /rest/bpm/monitor/models/{model id}/versions/{version}/kpis/config/{kpi id} [?locale={string}]
[&mode={string}]
```

The payload of the request contains the details of the definition. The following table is reproduced from the InfoCenter:

Name	Type	Required	Description
KPI ID	string	Yes*	The key performance indicator (KPI) ID. This value is required for KPI updates. It is not a required field to create a KPI. When a KPI is created, the KPI ID is derived from the KPI Display Name.
Model ID	string	Yes	The monitor model ID
Version	number	Yes	The monitor model version
Locale	string	No	The locale. This value consists of lowercase ISO language code (ISO 639) and the uppercase ISO country code (ISO 3166) joined by an underscore (for example, en_US).

			Results will be returned in the locale specified. If no locale is specified, the locale of the REST server will be used.
KPI Context ID	string	No	The key performance indicator (KPI) context ID
KPI Display Name	string	Yes	The key performance indicator (KPI) display name
KPI Cache Override Interval	number	No	Time in minutes for the KPI to cache the value. If set, this time interval overrides the KPI Cache Refresh set at the Model/version level on the Admin console. A value of zero indicates that the KPI should not be cached.
KPI Description	string	No	The key performance indicator (KPI) description
KPI Origin	string	Yes	The key performance indicator (KPI) origin. Valid values are "modeled" and "runtime"
KPI Data Type	string	Yes	The key performance indicator (KPI) data type. Valid values are "decimal" and "duration"
Target	number	No	The key performance indicator (KPI) target value. If the data type is "duration", the target is in milliseconds
KPI Range Type	string	Yes	The key performance indicator (KPI) range type. Valid values are "actualValue" and "percentage". Percentage indicates a percent of the target, where 100 = 100% of target. Required if KPI Range Array is not empty.
KPI Calc Method	string	Yes	The key performance indicator (KPI) calculation method. Valid values are "aggregated" and "calculated"
Aggregated Metric ID	string	Yes*	The ID of the aggregated metric Required if KPI Calc Method is "aggregated". Required if KPI Calc Method is "aggregated".
Aggregated Metric MC ID	string	Yes*	The ID of the monitoring context of the aggregated metric. Required if KPI Calc Method is "aggregated".
Aggregated Function	string	Yes*	The aggregated function. Valid values are "avg", "sum", "min", "max", and "count". Required if KPI Calc Method is "aggregated".
Version Aggregation	string	Yes	The scope of the metric aggregation, whether it is for instances within the same model version as the KPI or for instances across all model versions. Valid values are "singleVersion" and "allVersions". Required if KPI Calc Method is "aggregated".
Time Period Metric ID	string	Yes*	The ID of metric used for time period qualification. This is a date or datetime metric. Required if KPI Calc Method is "aggregated" and a Time Filter is used.
Time Period Method	string	Yes*	The time period method. Valid values are "repeatingPeriod", "rollingPeriod", and "fixedPeriod". Required if KPI Calc Method is "aggregated" and a Time Filter is used.
Repeating Period Duration	string	Yes*	The repeating period duration. Valid values are "yearly", "quarterly", "monthly", and "daily". Required if KPI Calc Method is "aggregated" and Time Period Method is

			"repeatingPeriod".
Repeating Period Basis	string	Yes*	The repeating period basis. Valid values are "previousPeriod" and "periodInProgress". For example, for year-to-date, use "periodInProgress". Required if KPI Calc Method is "aggregated" and Time Period Method is "repeatingPeriod".
Repeating Period Timezone	string	Yes*	The repeating period time zone. This is a Java timezone identifier (for example, America/Los Angeles). Required if KPI Calc Method is "aggregated" and Time Period Method is "repeatingPeriod".
Rolling Period Duration	string	Yes*	The rolling period duration. Valid values are "years", "months", "days", "hours", and "minutes". Required if KPI Calc Method is "aggregated" and Time Period Method is "rollingPeriod".
Rolling Period Quantity	number	Yes*	The number of rolling periods. Required if KPI Calc Method is "aggregated" and Time Period Method is "rollingPeriod".
Fixed Period Start	string	Yes*	The fixed period start time. Valid formats are '2007-01-01' and '2007-01-01T00:00:00' Either Fixed Period Start or Fixed Period End is required if KPI Calc Method is "aggregated" and Time Period Method is "fixedPeriod"
Fixed Period End	string	Yes*	The fixed period end time. Valid formats are '2007-01-01' and '2007-01-01T00:00:00' Either Fixed Period Start or Fixed Period End is required if KPI Calc Method is "aggregated" and Time Period Method is "fixedPeriod"
Fixed Period Timezone	string	Yes*	The fixed period time zone. This is a Java timezone identifier (for example, America/Los Angeles). Required if KPI Calc Method is "aggregated" and Time Period Method is "fixedPeriod".
Calculated KPI Expression	string	Yes*	The calculated key performance indicator (KPI) expression. This must be a valid XPath expression. Other KPIs and UDFs can be referenced. Required if the KPI Calc Method is "calculated".
User ID	string	No	The user ID of the key performance indicator (KPI) owner
View Access	string	Yes	The view access, whether the KPI can be viewed by others or not. Valid values are "public" and "personal".
Format Decimal Precision	number	No	The number of digits that are displayed after the decimal point for numeric KPIs.
Format Currency	string	No	The currency code to be used for formatting numeric KPIs. This 3-letter code must conform to ISO 4217 standards.
Format Percentage	boolean	No	The format percentage, whether the KPI will be displayed as a percentage or not. Valid values are "false" and "true".
Enable KPI History	boolean	No	Flag to indicate if KPI History is enabled. Valid values are true and false . Default value is true .
History Include Predictions	boolean	No	Flag to indicate if KPI History should include predictions.

			Valid values are "true" and "false". This parameter is only available for KPI Update. It defaults to "false" upon KPI create.
KPI History Defaults	array	No	Array to contain KPI history default values. This parameter and all sub-parameters within the array are only available for KPI Update.
History Time Range Start	string	No	Fixed period start date for retrieving KPI History. Valid formats are '20081201T123000' , '2008-12-01' and '20081201'.
History Time Range End	string	No	Fixed period end date for retrieving KPI History. Valid formats are '20081201T123000' , '2008-12-01' and '20081201'.
History Repeating Period Quantity	number	No	The number of KPI History periods to retrieve. For a periodInProgress query, the current period would count as 1. It defaults to 2 upon KPI create.
History Rolling Period Duration	string	No	The rolling period duration. Valid values are "yearly", "quarterly", "weekly", "monthly", and "daily". It defaults to "monthly" upon KPI create.
History Repeating Period Basis	string	No	The repeating period basis. Valid values are "previousPeriod" and "periodInProgress". For example, for year-to-date, use "periodInProgress". It defaults to "periodInProgress" upon KPI create.
History All Versions	boolean	No	Flag to indicate if KPI History should include all version of the KPI. This flag must be false if the KPI is a single version KPI. Valid values are "true" and "false". It defaults to "false" upon KPI create.
History Time Range Method	string	No	The time period method. Valid values are "repeatingPeriod", "rollingPeriod", and "fixedPeriod". It defaults to "repeatingPeriod" upon KPI create.
History Rolling Period Quantity	number	No	The number of KPI History periods to retrieve.
History Display Ranges	string	No	Flag to indicate if KPI Ranges should be displayed in the KPI History widget. Valid values are "true" and "false". It defaults to "false" upon KPI create.
History Granularity	string	No	The value can be yearly, quarterly, monthly, weekly, daily and hourly. It defaults to "daily" upon KPI create.
History Valid From	string	No	Timestamp used to indicate the earliest date/time that KPI History is valid. For example, this is initially set to the KPI creation date. KPI History would not be valid prior to the creation of the KPI.
History Timezone	string	No	Timezone used by KPI History. This is a Java timezone identifier (for example, America/Los Angeles). It defaults to the server timezone upon KPI create.
History Repeating Period Duration	string	No	The repeating period duration. Valid values are "yearly", "quarterly", "weekly", "monthly", and "daily".

History Display Target	string	No	Flag to indicate if KPI Target should be displayed in the KPI History widget. Valid values are "true" and "false". It defaults to "false" upon KPI create.
Enable KPI Prediction	Boolean	No	Flag to indicate if KPI Prediction is enabled. Valid values are "true" and "false". This parameter is only available for KPI Update. It defaults to "false" upon KPI create.
Default Prediction Model ID	string	No	The ID for the default prediction model. This parameter is only available for KPI Update.
KPI Metric Filter Array	string	No	Array of filters of aggregated key performance indicators (KPIs)
KPI Metric Filter ID	string	No	The filter ID for each of the metric filters. This value is used if provided. Otherwise, it is automatically generated.
Filter Metric ID	string	No	The ID of the metric used for filtering
Filter Operator	string	Yes	The filter operator. Valid values are "equals", "lessThan", "lessThanOrEquals", "greaterThan", "greaterThanOrEquals", "notEquals", "in", "notIn", "isNull", "isNotNull", "like", "notLike".
Filter Operator Case Sensitive	boolean	Yes	Whether the filter operator is case-sensitive or not when using string-based metric filters. Valid values are "false" and "true".
Filter Value	string	Yes*	The filter value. For information on the format for each data type, see the section on filtering. Required unless the operators isNull or isNotNull are specified. The filter value can be specified as an array. For example, a string would be represented as ["Smith"]. An array of values included with the "in" operator would be represented as ["Smith", "Jones"]. Note that string values are surrounded by a single quotation marks for the XPath formatting and double quotation marks for the JSON formatting.
KPI Range Array	array	No	Array to contain any KPI ranges
KPI Range ID	string	No	The range ID for each of the key performance indicator (KPI) ranges. This value is used if provided. Otherwise, it is automatically generated.
KPI Range Display Name	string	Yes	The key performance indicator (KPI) range display name
KPI Range Start Value	number	Yes	The key performance indicator (KPI) range start value. This value can be defined as an actual value or as a percent of a target as defined by KPI Range Type. If KPI Range Type is "percentage", use a value such as 100 to represent 100%. If duration KPI and if KPI Range Type is an "actualValue", then the start value is in milliseconds
KPI Range End Value	number	Yes	The key performance indicator (KPI) range end value. This value can be defined as an actual value or as a percent of a target as defined by KPI Range Type. If KPI Range Type is "percentage" use a value such as 100 to represent 100%. If duration KPI and if KPI Range Type is an "actualValue", then the end value is in milliseconds
KPI Range Color	string	No	The display color in key performance indicator (KPI)

			gauges. Valid values are #000000 - #FFFFFF (omit the # sign)
KPI Range Icon	string	No	The key performance indicator (KPI) range icon display icon in key performance indicator (KPI) tables, e.g. images/kpi/monitorIcons/IBM_down_red.gif

This is a lot of data and some additional discussion is needed. Thankfully, all of these parameters can be grouped into sections.

### **Time Filters**

Time filters are used to define which data will be aggregated together to build a KPI. There are four methods of filtering possibilities:

- <none>
- repeatingPeriod
- rollingPeriod
- fixedPeriod

The method of filtering used is stored in the "Time Period Method" property. If this property is NOT set then no time filter is assumed. Depending on which method is chosen, other properties come into play:

- repeatingPeriod
  - "Time Period Metric ID"
  - "Repeating Period Duration"
    - "yearly"
    - "quarterly"
    - "monthly"
    - "daily"
  - "Repeating Period Basis"
    - previousPeriod
    - periodInProgress
  - "Repeating Period Timezone"
- rollingPeriod
  - "Time Period Metric ID"
  - "Rolling Period Duration"
    - "years"
    - "months"
    - "days"

- "hours"
- "minutes"
- "Rolling Period Quantity" – a numeric
- fixedPeriod
  - "Time Period Metric ID"
  - "Fixed Period Start" – The date/time at which the fixed period begins. The format of this value is "YYYY-MM-DDTHH:MM:SS"
  - "Fixed Period End" – The date/time at which the fixed period ends. The format of this value is "YYYY-MM-DDTHH:MM:SS"
  - "Fixed Period Timezone"

## Get Historical KPI data

This REST API can be used to retrieve the historical values of a specific KPI.

```
GET /rest/bpm/monitor/models/{modelId}/versions/{version}/kpis/history/{kpiId}
```

An example of the data returned is shown below:

```
{
 "Model ID": "BUSMONMonitoringModel",
 "Model Display Name": "BUSMONMonitoringModel",
 "Version": 20140127161149,
 "KPI Display Name": "A1",
 "KPI ID": "A1ID",
 "KPI Data Type": "decimal",
 "KPI Description": "A1 Desc",
 "Target": 200.0,
 "Target Localized": "200.00",
 "Range Start Timestamp": "2014-01-01T06:00:00",
 "Range Start Timestamp Localized": null,
 "Range Start Timestamp In Timezone": "2014-01-01T00:00:00",
 "Range End Timestamp": "2014-02-19T06:00:00",
 "Range End Timestamp Localized": null,
 "Range End Timestamp In Timezone": "2014-02-19T00:00:00",
 "DisplayFGSIndication": true,
 "FGSSecurityFilterApplied": false,
 "KPI Prediction Array": [],
 "KPI Value Array": [
 {
 "KPI Value Localized": null,
 "Target": 200.0,
 "KPI Period Timestamp": "2014-01-02T06:00:00",
 "KPI Value": null,
 "KPI Period Timestamp In Timezone": "2014-01-02T00:00:00",
 "Target Localized": "200.00",
 "KPI Period Timestamp Localized": "Jan 1, 2014"
 }
],
 ...
 "KPI Range Array": [
 {
 "KPI Range Color": "#b22222",
 "KPI Range Start Value": 0.0,
 "KPI Range Icon": null,
 "KPI Range End Value Localized": "100.00",
 "KPI Range Display Name": "Low",
 "KPI Range ID": "Low",
 "KPI Range End Value": 100.0,
 "KPI Range Start Value Localized": "0.00"
 },
 ...
]
}
```

There are other options that can also be supplied. These include:

- **granularity** – This is definition of how many data points within a time range should be returned. The values of this option are:
  - hourly
  - daily
  - weekly
  - monthly
  - quarterly
  - yearly
- **timerangemethod** – This definition determines the range of time over which the history is retrieved. The values of this option are:
  - repeatingPeriod
  - rollingPeriod
  - fixedPeriod

For `timerrangemethod = rollingPeriod`, the following apply:

- **rollingperiodduration** – This defines the duration of a rolling period unit. The choices are one of the following:
  - hours
  - days
  - weeks
  - months
  - quarters
  - years
- **rollingperiodquantity** – A numeric value defining how "many" of the `rollingperiodduration` intervals should be included.

For `timerangemethod = repeatingPeriod`, the following apply:

- **repeatingperiodbasis** – A choice of whether to include or exclude the current period. Choices are one of:
  - previousPeriod
  - periodInProgress
- **repeatingperiodduration** – The repeating period duration. Allowable values are one of:
  - daily
  - monthly
  - quarterly

- yearly
- repeatingPeriodQuantity – The number of periods to include.

For timerangemethod = fixedPeriod, the following apply:

- timerangeStart – The start of the time period in the format of either "YYYY-MM-DD" or "YYYY-MM-DDTHH:MM:SS".
- timerangeEnd – The end of the time period in the format of either "YYYY-MM-DD" or "YYYY-MM-DDTHH:MM:SS".

## Get Dashboard Alerts

Monitor maintains a list of alerts to be shown to the user on a dashboard. This API call retrieves that list.

GET /rest/bpm/monitor/alerts/dashboardalerts

The following is an example response:

```
{
 "Record Count": 5,
 "Page Size": 10,
 "Dashboard Alert Array": [
 {
 "Creation Timestamp Localized": "February 4, 2014 10:48:09 AM",
 "Model Display Name": "TestModel",
 "Model ID": "TestModel",
 "Status": "Available",
 "Priority": 3,
 "Creation Timestamp": "2014-02-04T16:48:09",
 "Instance ID": "1",
 "Acknowledged": true,
 "ID": "508E2F5B25C5AFAEA7C4F1AA2",
 "Context ID": "P1",
 "Subject": "Hi Amount to Left"
 },
 {
 ...
 }],
 "Page Number": 1
}
```

## Get Alert Subscriptions

GET /rest/bpm/monitor/alerts/subscriptions

## Create Alert Definition

POST /rest/bpm/monitor/situation/alerts/config

Model ID	string	The model ID this alert is based on
Version	number	The version of the model this alert is based on
Name	string	The display name given by the user
User ID	string	The owner of this situation. Only users with KPI-Administrator will be allowed to create situations with an owner other then themselves.
State	string	The state of this situation. Possible states are: inactive, active, invalid.
Description	string	The display description given by the user

Subject	string	The subject line of the alert message
Body	string	The body of the alert message
GeneratedContent	boolean	This will indicate if the body and subject were system generated, Default is false.
View Access	number	This will control access to view and subscriptions to this alert. 0 = private, 1 = public, Default is 0 (private).
TimingInterval	string	<p>Describes the base timing interval, options are:</p> <ul style="list-style-type: none"> <li>• MINUTE: indicates that minutes are the base timing amount</li> <li>• HOUR: indicates that hours are the base timing amount</li> <li>• DAY; indicates that days are the base timing amount</li> <li>• WEEK: indicates that weeks are the base timing amount</li> <li>• MONTH: indicates that months are the base timing amount</li> <li>• PERIOD: indicates that the KPI period will be used</li> </ul>
TimingStartOffset	string	<p>When TimingInterval MINUTE: intervals will be calculated starting at the specified minutes after the top of the hour.</p> <p>When TimingInterval HOUR: intervals will be calculated starting at the specified hours and minutes after midnight in the TimingTimeZone property.</p> <p>When TimingInterval DAY: will be the time of day in the TimingTimeZone property to evaluate the conditions.</p> <p>When TimingInterval WEEK: will be the time of day and the day of the week in the TimingTimeZone property to evaluate the conditions.</p> <p>When TimingInterval MONTH: will be the time of day and the day of the month in the TimingTimeZone property to evaluate the conditions.</p> <p>When TimingInterval PERIOD: does not offset the evaluation time, the end of period time must be honored</p>
TimingMultiple	number	<p>Describes a multiple of the base timing interval, optional will default to one.</p> <p>When TimingInterval MINUTE: the number of minutes in the interval. (only allow even factors of the hour, 1, 2,3, 4, 5, 6, 10, 12, 15, 20 ,30 )</p> <p>When TimingInterval HOUR: the number of hours in the interval (only allow even factors of the day, 1, 2, 3, 4, 6, 8, 12 )</p> <p>When TimingInterval DAY: the number of days in the interval (allow 1 to 365).</p> <p>When TimingInterval WEEK: the number of weeks in the interval (allow up to 52)</p> <p>When TimingInterval MONTH: the number of months (allow up to</p>

		<p>12).</p> <p>When TimingInterval PERIOD: TimingMultiple property is not used.</p>
TimingRepeat	string	<p>When the conditions will be allowed to send an alert.</p> <p>REPEATING = whenever the condition evaluates to true</p> <p>NONREPEATING = when the condition evaluates to true and not again until the condition evaluates to false</p> <p>ONCEINPERIOD = when the condition evaluates to true and not again until the next KPI period</p>
TimingKPI	string	Describes the ID of the KPI to use for the period time evaluations, required when TimingRepeat=ONCEINPERIOD and TimingInterval=PERIOD.
TimingTimeZone	string	Describes the Timezone to use in calculating the actual start time and next evaluation times, must use the JAVA based time zone strings, Optional, will use the server time zone.
TriggerArray	array	Describes a conditional trigger. All triggers will be 'anded' together to determine if the situation will fire. Each trigger is described as an array item of the following properties.
CheckKPI	string	Indicates which KPI should be used in this evaluation
CheckPrediction	string	Indicates which Prediction KPI to be used in this evaluation
CheckPredictionType	string	<p>Indicates which Prediction KPI type to be used in this evaluation.</p> <p>Types are:</p> <p>end: for the end of the prediction period,</p> <p>any: for any value within the prediction set</p>
CheckCondition	string	Indicates which type of condition to use. The operations will be done using the check KPI and the check field.
CheckFieldType	string	Indicates what type of field will be compared against the KPI.
CheckField	string	<p>Indicates actual value or where to get the value to use in the evaluation.</p> <p>When CheckFieldType is 'value' check field will be the actual value to use.</p> <p>When CheckFieldType is 'target' check field will be ignored."</p> <p>When CheckFieldType is 'range' check field will be the name of the KPI range.</p>
Alert Subscription Array	array	Describes subscribers to this situation. Each subscription is described as an array item of the following setting.
User ID	string	The user who is subscribed to this alert

E-mail	string	The alert format is E-mail
Dashboard	boolean	The alert format is Dashboard
Cell	boolean	The alert format is Cell
Pager	boolean	The alert format is Pager

## ***Useful Monitor/REST JavaScript***

Here are some useful JavaScript fragments for working with monitor data in the context of a JavaScript environment that is likely to be found within Web Browsers or Business Space Widgets.

### **Building a map of Instance data**

The instance data returned by REST calls is an array format and not a "map". On occasion, map formatted data is better to work with. This JavaScript function will take a REST returned instances object and produce an Array of objects where each object is a map of name->value of instance metrics:

```
/**
 * Given an instances object retrieved from monitor, build a map of the data
 *
 * @param instancesObject An object returned from a REST query against Monitor
 * @returns An array of instance objects
 */
function buildInstanceMap(instancesObject)
{
 var retArray = new Array();
 // Iterate over each instance
 for (var i in instancesObject["Instance Data"])
 {
 var currentInstance = new Object();
 // Iterate over each metric in the instance
 for (var m in instancesObject["Metric ID Array"])
 {
 var metricName = instancesObject["Metric ID Array"][m];
 currentInstance[metricName] = instancesObject["Instance Data"][i]["Metric Data"][m];
 }
 retArray.push(currentInstance);
 }
 return retArray;
}
```

## **Monitor Security**

If security is enabled on the Monitor server, authorizations must be supplied to allow users to work with the models.

In the Admin console under Security > Monitor Data Security



configuration panels can be found. From there we drill down to the mysterious "root" entry:

This screenshot shows the 'Monitor Data Security Administration' page. At the top, there are 'Configure' and 'Diagnose' tabs, with 'Configure' selected. Below the tabs, the title 'Monitor Data Security Administration' is displayed with a subtitle: 'Use this page to perform administrative functions for WebSphere Business Monitor data security. You can add a new resource group, select a resource group to delete, or view models in a resource group.' Under the heading 'Resource Groups', there is a table with the following data:

Resource Groups		
New	Delete	
Select	Name	Description
<input type="checkbox"/>	<a href="#">root</a>	The 'root' resource group is the default, parent resource group
1 Total		

and finally get to where we need to be:

Selecting a category of permissions and then clicking either the users or groups buttons allows us to add/remove users/groups from that category of permissions.

Permissions normally need to be set if when you create an Instances widget in Business Space and no models are shown.

## Advanced Model development

### Expressions

The XPath functions available in Monitor are a subset of the XPath specification. The functions supported are shown in the following lists.

#### Data functions

- `fn:empty` - Takes any type of argument and returns a boolean. The boolean is true if the argument is an empty sequence, and false otherwise.
- `fn:exists` - Takes any type of argument and returns a boolean. The boolean is true if the argument is not an empty sequence, and false if the argument is an empty sequence.
- `fn:false` – Always results in a false value.
- `fn:not` – Inverts a boolean value.
- `fn:true` – Always results in a true value.

## **String functions**

- fn:compare
- fn:concat
- fn:contains
- fn:ends-with
- fn:lower-case
- fn:normalize-space
- fn:starts-with - Takes two strings and tests whether the first string starts with the second string. For example, fn:starts-with("12345", "12345") returns true, fn:starts-with("12345", "123") returns true, and fn:starts-with("12345", "234") returns false.
- fn:string-length
- fn:substring
- fn:upper-case

## **Math functions**

- fn:abs - Two variations. One for xs:decimal and one for xs:integer.
- fn:avg
- fn:max
- fn:min
- fn:round
- fn:sum

## **Time functions**

Dates are formatted using XMLSchema format

See:

<http://www.w3.org/TR/xmlschema-2/#date-lexical-representation>

- fn:current-date
- fn:current-time
- fn:day-from-date

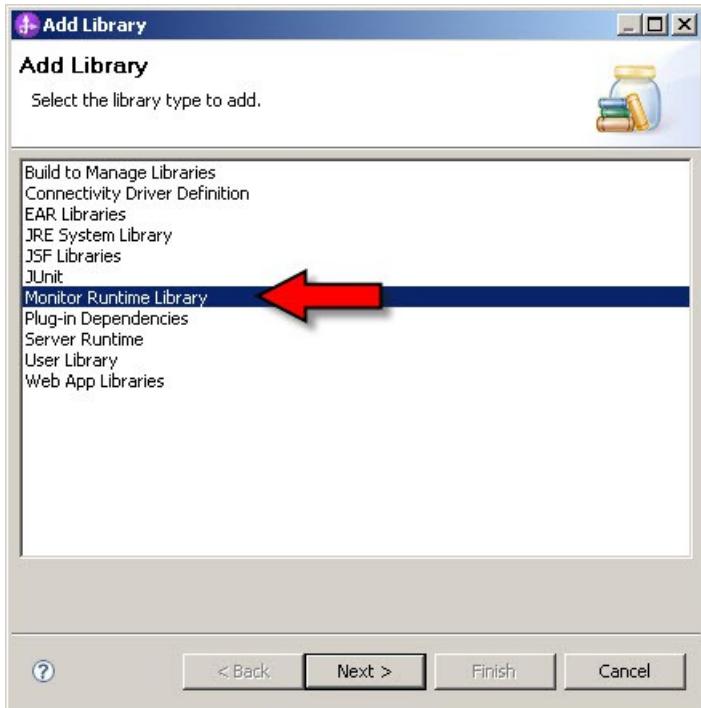
- fn:day-from-datetime
- fn:days-from-duration – Returns the number of days in the specified duration.
- fn:adjust-date-to-timezone
- fn:adjust-datetime-to-timezone
- fn:adjust-time-to-timezone
- fn:hours-from-datetime
- fn:hours-from-duration – Returns the number of hours in the specified duration.
- fn:hours-from-time
- fn:minutes-from-datetime
- fn:minutes-from-duration – Returns the number of minutes in the specified duration.
- fn:minutes-from-time
- fn:month-from-date
- fn:month-from-datetime
- fn:seconds-from-datetime
- fn:seconds-from-duration – Returns the number of seconds in the specified duration.
- fn:seconds-from-time
- fn:timezone-from-date
- fn:timezone-from-datetime
- fn:timezone-from-time
- fn:year-from-date
- fn:year-from-datetime

## **IBM Monitor extensions**

- wbm:escape-special-characters - Returns the argument with occurrences of the special characters <, >, &, ", <newline> replaced with the HTML equivalents <, >, &, ', <br />. Tabs are replaced with four non-breaking spaces ( ). For example, to format a shipping address so that it displays in a Web browser, you might use "wbm:escape-special-characters(wbm:serialize(newOrderEvent/orderPart/ord:shippingAddress))".
- wbm:evaluate – Evaluate an XPath 2.0 expression on an XML document fragment and returns a string.
- wbm:send-events
- wbm:serialize – Converts an XML document fragment from an event into a string representation.

## Authoring User Defined XPath functions

Custom XPath functions can be written in Java. Create a Java project in WID/RAD and define a class. In that class, code methods that are to be executed as XPath functions. Each method should be declared as static. Add the Monitor Runtime to the project's dependencies. To achieve this, open the project's build path and switch to the Libraries tab. Click the Add Library button and select Monitor Runtime Library from the list of options.



In front of each method that is to be exposed, add a Java annotation that looks as follows:

```
@XPathFunction(namespaceName = "http://mynamespace",
 localName = "myLocalName",
 description = "My Description",
 isSelfContained = true,
 isDeterministic = true)
```

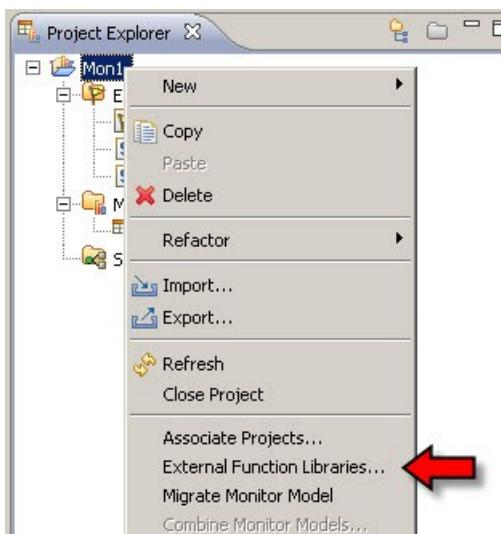
Where:

- namespaceName is a namespace for the function
- localName is the name of the function as it will appear in the XPath editor
- description is a human readable description for the function. It is the description information that shows up when the function is selected and hovered over with the mouse.

The annotations come from the import of com.ibm.wbimonitor.xml.expression.udf.XPathFunction.

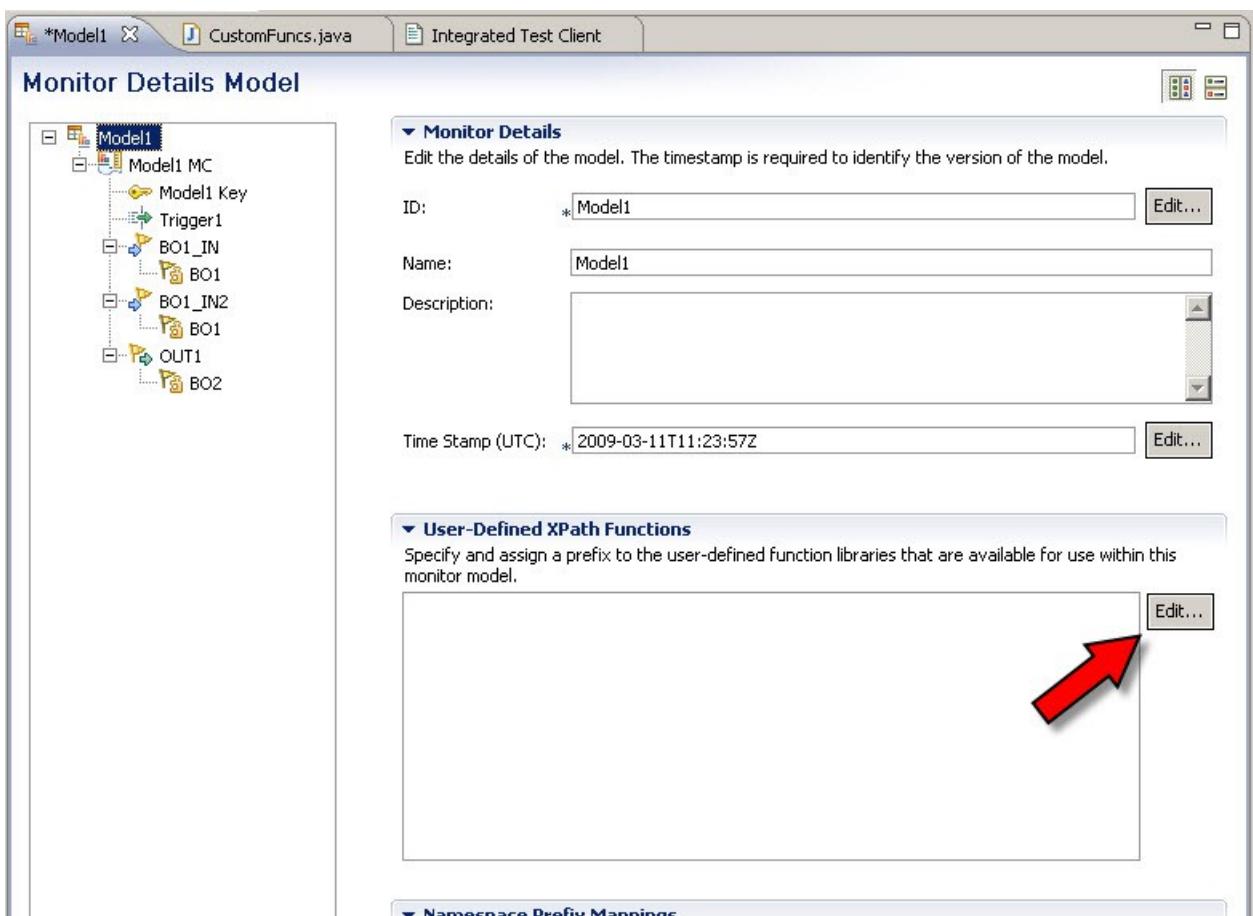
Each function should return a value that will be the outcome of the expression. The functions can also take parameters. When the functions have been written, export the Classes as a JAR file. The JAR file should then be copied into the Monitor project.

In the ID editor, right-click on the Monitor project and select External Function Libraries... from the menu.

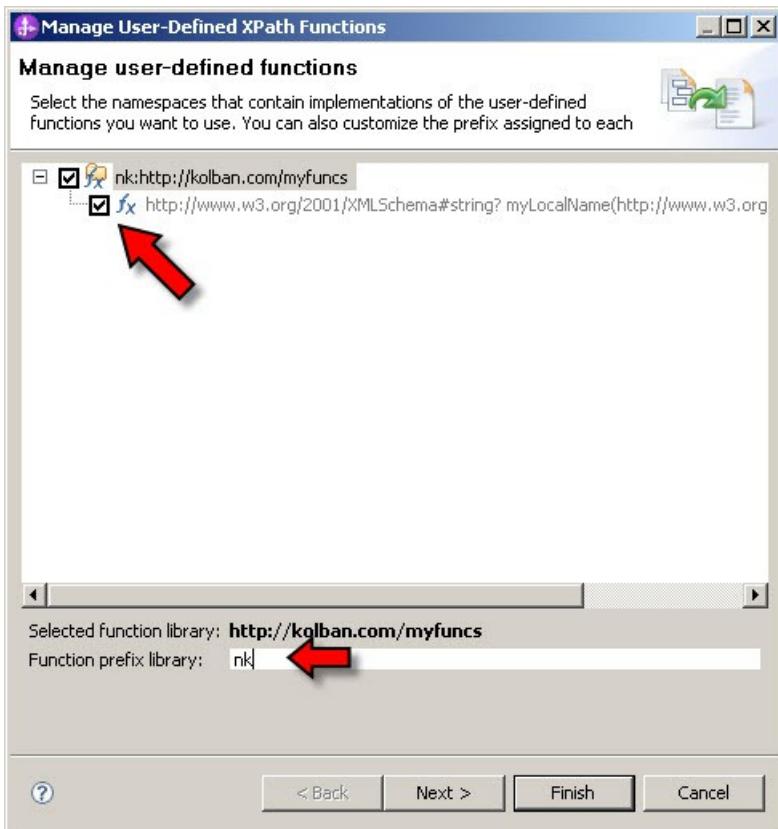


In the dialog, select Add JARs and pick the JAR that was added to the Monitor project.

Next, open the Monitor model and in the User Defined XPath Functions, click the Edit... button.



Select each of the function libraries to be used in this project. For each library, a prefix must be entered.



The custom user defined XPath function has now been added. In the XPath editors and expression builders, the Java code can now be referred to by its prefix and localname. At runtime, the call to the associated Java code will occur.

#### UDF Debugging Notes:

Getting an NPE during loading of the UDF usually means something is wrong. Currently there's not a good error message to help id the problem. Check the annotation, types, etc. For instance, accidentally setting a return type of boolean rather than Boolean will cause an NPE upon loading.

Make sure that you compile with the correct java compiler. 1.5 for 6.2. Trying to run code compiled with a 1.6 compiler will give you the following error: Caused by:  
`java.lang.UnsupportedClassVersionError: (com/mts/BusinessDay) bad major version at offset=6`

#### Data Types

##### Method return types

XML datatype	Java return type
xs:string	java.lang.String
xs:integer, xs:decimal	java.math.BigInteger
xs:decimal	java.math.BigDecimal
xs:boolean	java.lang.Boolean

xs:date, xs:time, xs:dateTime	javax.xml.datatype.XMLGregorianCalendar
xs:duration, xs:dayTimeDuration, xs:yearMonthDuration	javax.xml.datatype.Duration
xs:dateTime	java.util.Calendar
xs:string	java.net.URI
xs:string	java.util.UUID

## Parameter Types

XML datatype	Java return type
xs:string	java.lang.String
xs:integer	java.math.BigInteger
xs:decimal, xs:integer	java.math.BigDecimal
xs:boolean	java.lang.Boolean
xs:date, xs:time, xs:dateTime	javax.xml.datatype.XMLGregorianCalendar
xs:duration, xs:dayTimeDuration, xs:yearMonthDuration	javax.xml.datatype.Duration

## Specific Notes

### Duration

If a metric is defined as Duration, the value assigned to that metric must be cast to `xs:dayTimeDuration()`.

### ***Two Stage Modeling***

Choose which events you wish to propagate. For example, if we care about when a task starts and ends, we need only publish an event on that concept.

## Monitor and Cognos

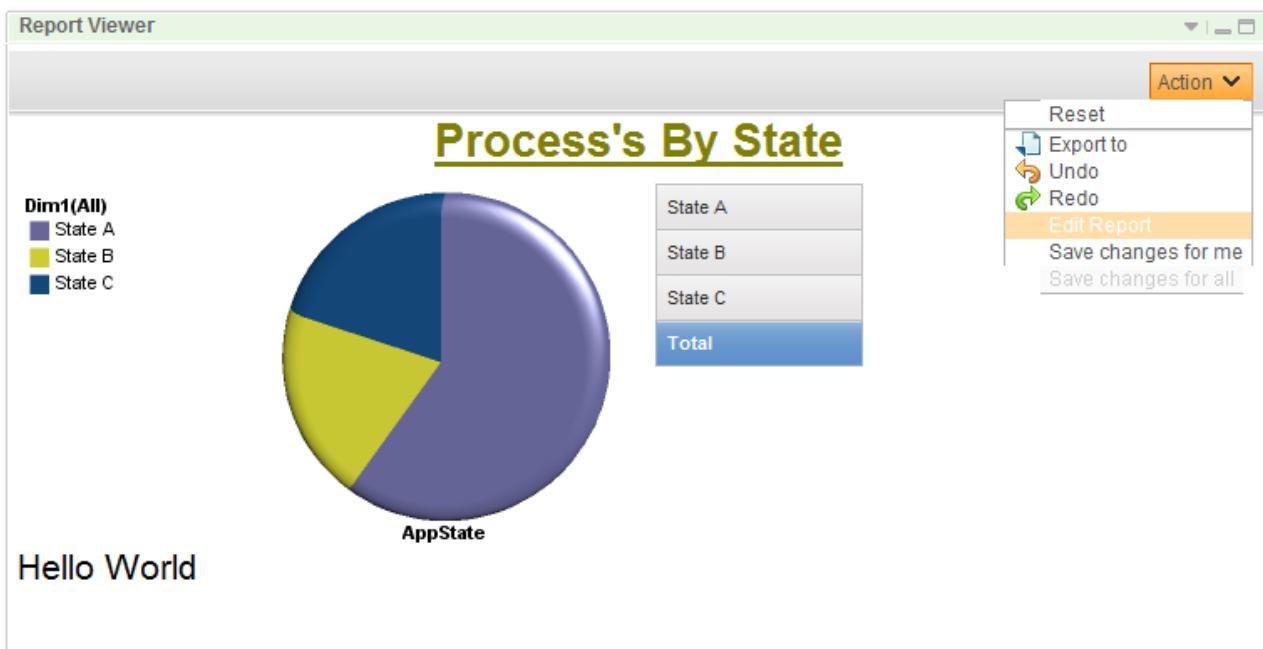
IBM Business Monitor supplies a copy of IBM Cognos as part of the product solution. Cognos is a Business Intelligence and Analytics engine. From a Monitor perspective, Cognos provides the capability to define and view dimensional models.

It is the Business Space Report Viewer widget that displays the model.

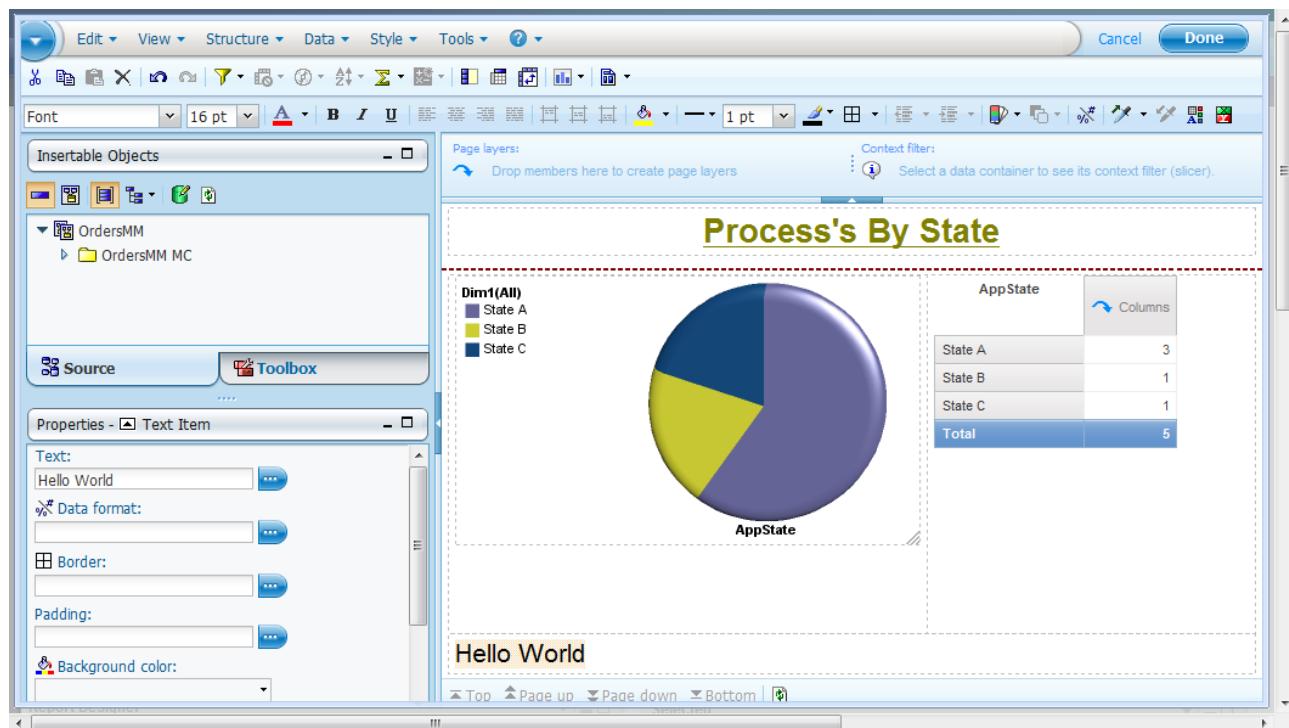
See also:

- [IBM Cognos Home Page](#)
- [IBM Cognos BI 10.1 Information Center](#)

Cognos has a bewildering number of components and tools. Many of these don't identify themselves so you simply have to know which one is which. For example, when looking at the Report Viewer widget, the Edit Report menu item:



Will bring up a web based tool called IBM Cognos Business Insight Advanced but there is nothing on the page to let you know that *this* is what you are in. (Later: Actually, I take that back ... in the Help menu there is an about box).



## Cubes, Categories, Dimensions and Levels

Experience has shown that the following topic can be difficult for many folks to understand. Let us examine a scenario. Imagine a sales record that may contain fields as follows:

- US State of sale
- City of sale
- Amount of sale
- Sales rep
- Category of sale
  - Hardware
  - Software
  - Services

Now imagine that over time instances of such records will arrive. After, say, 6 months worth of data, we now want to perform some analysis of what we have found. What kinds of questions can we answer? Here are some possibilities:

- How much revenue did we make? (Sum of all sales)
- Which state was the most profitable? (Sum of all sales by state)
- Which sales rep sold the most in New York?
- Which sells more in March, Hardware or Software?

Answering such questions from the examination of this data is where Cognos comes into play. In order to also start answering these questions we need to delve into cubes, measures and dimensions.

Lets start with measures. Measures can be thought of as a roll-up of a metric such as sales amount. This takes all the instance data values of the metric and has a function applied to them. The functions available include:

- Minimum – Value is the minimum value of the metric over the range
- Maximum – Value is the maximum value of the metric over the range
- Sum – Value is the arithmetic sum of all of the values over the range
- Count – Value is the count of metrics that exist over the range
- Average – Value is the arithmetic average of the metric over the range
- Standard deviation – Value is the arithmetic standard deviation over the range

Measures are defined in the Dimensional Model tab of the Monitor Model editor. We define the name of the measure to be created and which metric is used for its creation along with the aggregation function to be applied to those metrics.

### ▼ Measures

Work with the measures for this cube. Measures are calculations based on a metric, key, counter, or stopwatch.

Measure	Source Metric	Aggregation Function	
sales amount	saleAmount	Sum	<a href="#">New...</a>
			<a href="#">Remove</a>

When charting, think of the measure as the value of the Y axis.

If measures are aggregates of metrics, dimensions can be thought of as the selections of which measures to aggregate together. For example, if we have a measure that is the sum of sales amounts and we have a dimension which is the US state in which the sale occurred, then we can determine the value of all sales by state.

When we define a dimension, we also define levels. The Levels define the levels of drill down. Every dimension must have at least one level. For example, if we imagine a dimension called "location" used to define the location in which a sale occurred, we might define a level as being the "US State". We will then be able to see all sales by US State. If we define a second level which is "City", then we will be able to drill down each US State to the cities from that state which have records.

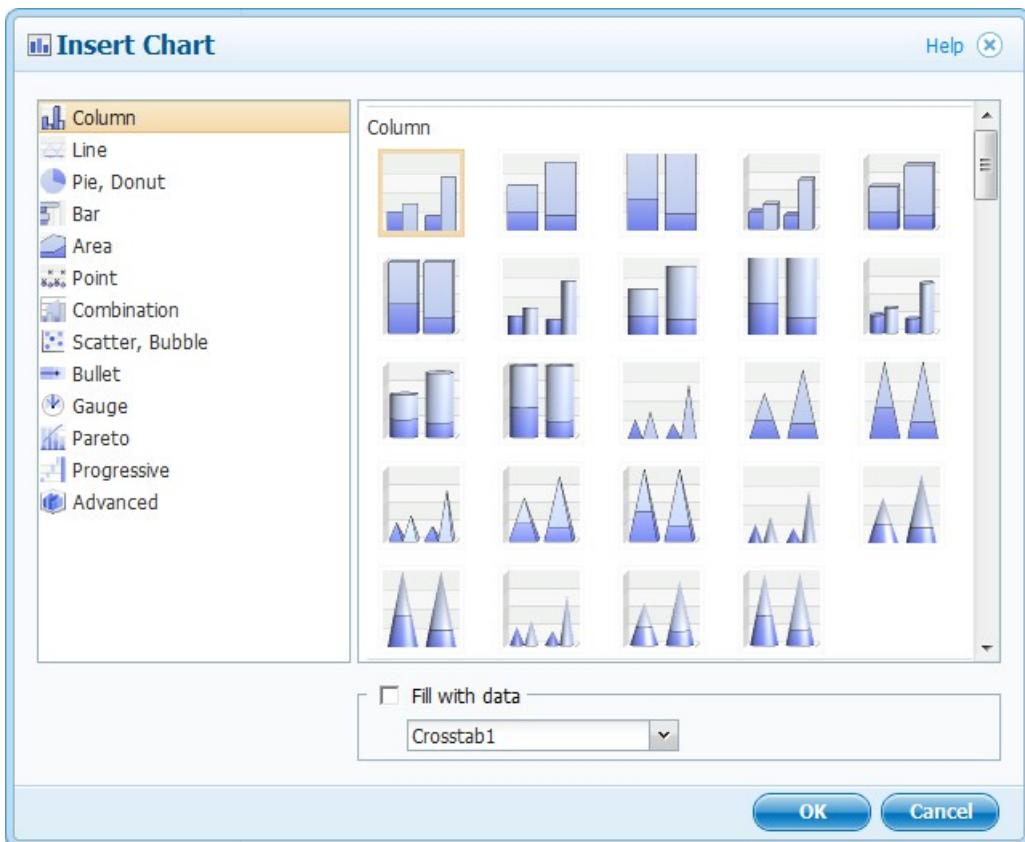
### ▼ Dimensions

Work with the dimensions and dimension levels of this cube. Dimensions are data categories made up of hierarchical dimension levels.

Dimension / Dimension Level	Source Metric	
Location		<a href="#">New Dimension...</a>
usState	usState	<a href="#">New Level...</a>
city	city	<a href="#">Remove</a>
		<a href="#">Move Up</a>
		<a href="#">Move Down</a>

## Cognos Charting

There are an amazing number of chart types available in Cognos. To see a summary list, select the Insert Chart icon and have a look:



Here are some key terms to help us understand charting.

### Data Series

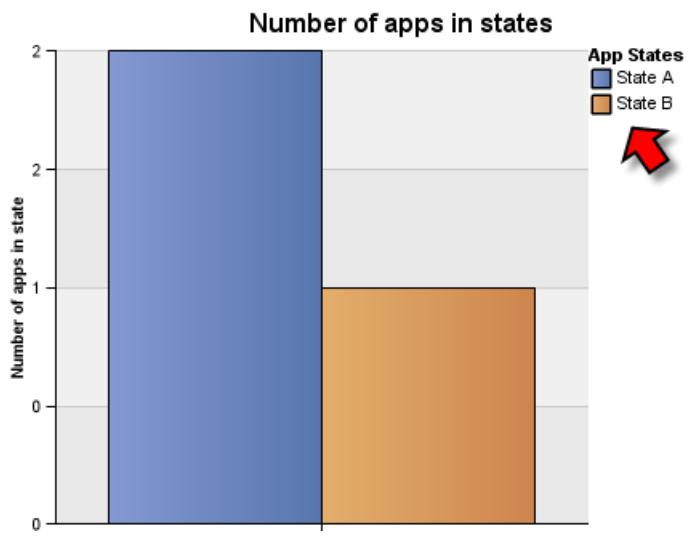
The Data Series can be thought of (loosely) as the X-Axis values in a bar chart. They are the entries that have specific values. For example, a data series for sales may be 2009, 2010, 2011 where each element in the series has a value (for examples, sales in that year). Each element in the Data Series has its own color or distinguishing characteristic.

### Primary Axis

The primary axis can be thought of as the Y-Axis. This is where values will be plotted against the elements in the Data Series. It may help to think of the X-Axis as holding qualitative data (years, regions, etc) while the Y-Axis holds quantitative data (sales, number of customers, etc).

### Legend

The Legend is the key to the patterns or colors used to indicate the Data Series. The following image illustrates a Legend.



## Business Space Widgets for Monitor

The following widgets are supplied by Monitor for use in Business Space:

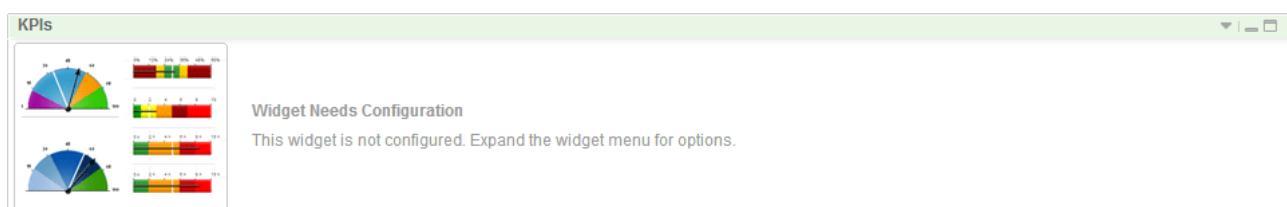
- Alerts
- Diagrams
- Human Tasks
- Instances
- KPIs
- KPI History and Prediction
- Report Viewer
- Reports

### KPIs

The KPI widget looks as follows from the palette of available widgets:



once added to the page, it shows as:



indicating that it needs to be configured. We can configure it from the editing settings on the right. When selected we are presented with a list of modeled KPIs:

**KPIs**

- KPIs    Layout    Wiring

Select the models for which you want to retrieve the KPIs:

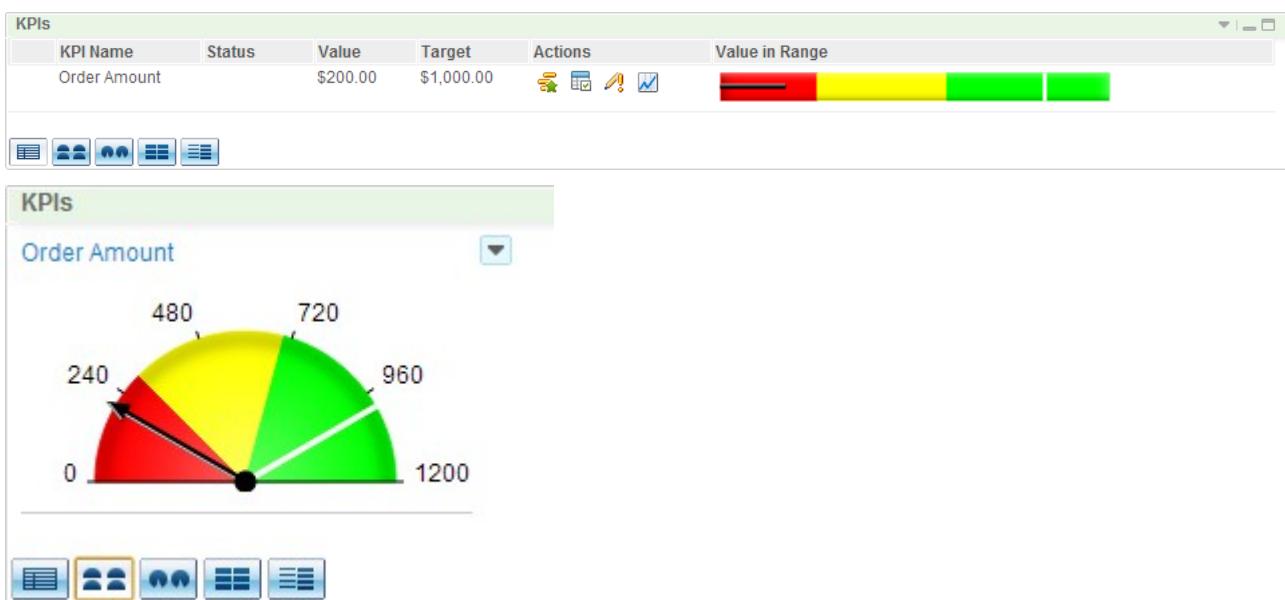
- Latest version     All versions

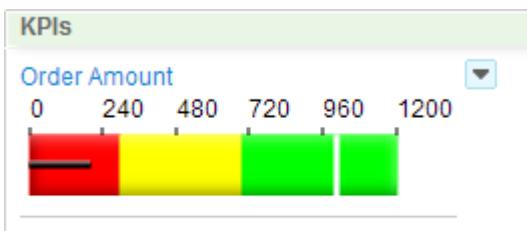
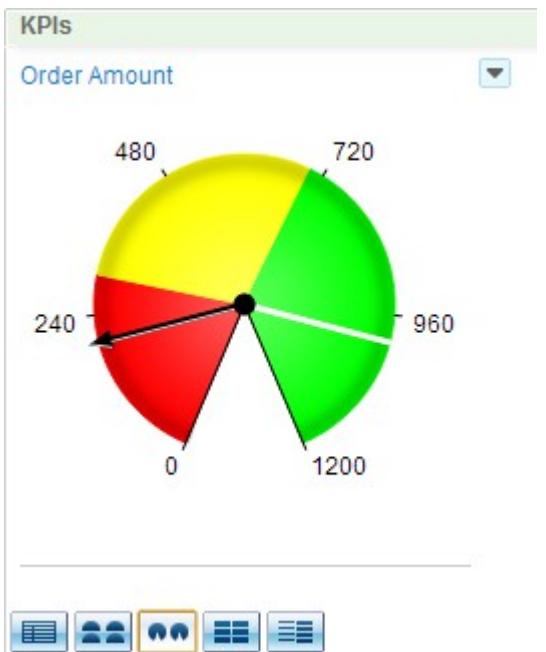
MonitorMonitoringModel 2014-01-04 05:11:34

- End Count
- Enter Order Details Average Cost
- Enter Order Details Average Execution Time (Clock)
- Enter Order Details Average Labor Cost
- Enter Order Details Average Resource Cost
- Enter Order Details Average Rework (percent true)
- Enter Order Details Average Total Time (Clock)
- Enter Order Details Average Value Add (percent true)
- Enter Order Details Average Wait Time (Clock)
- Enter Order Details Count
- Handle Order Average Total Time (Clock)
- Handle Order Count
- Order Amount
- Start Count

OK    Apply    Restore    Cancel

from which we can select one or more. The KPI can be visualized a number of different ways that can be selected from the icons at the bottom left of the widget.





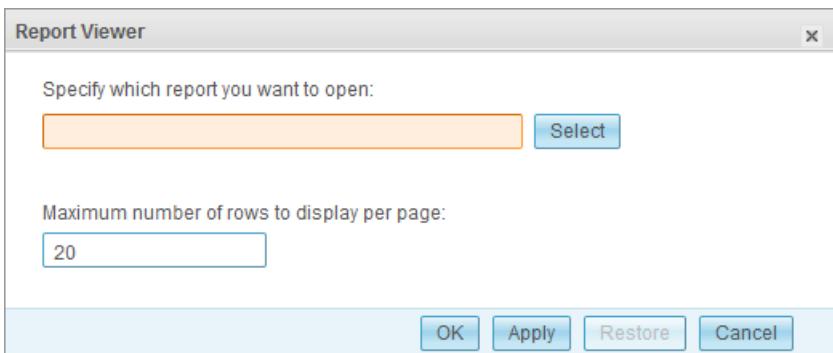
## ***Report Viewer Business Space Widget***

When initially added to a page, this widget needs to be configured.

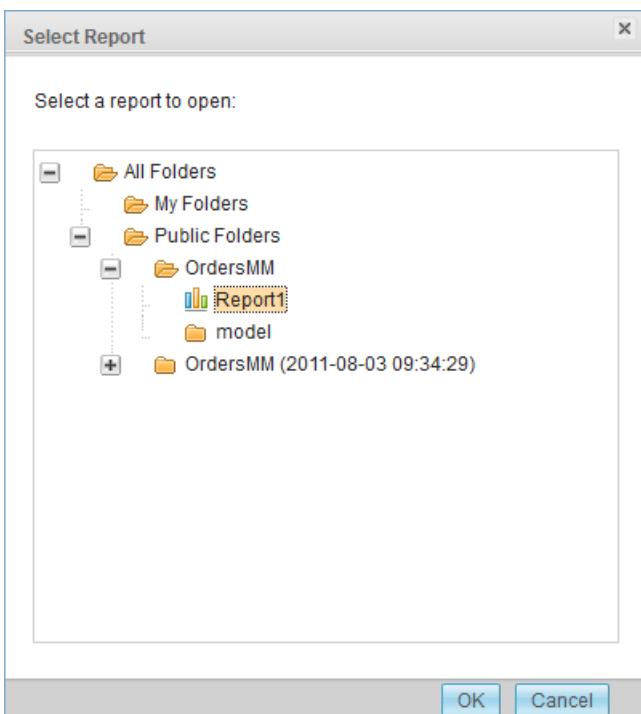
Report Viewer

Widget Needs Configuration  
This widget is not configured. Expand the widget menu for options.

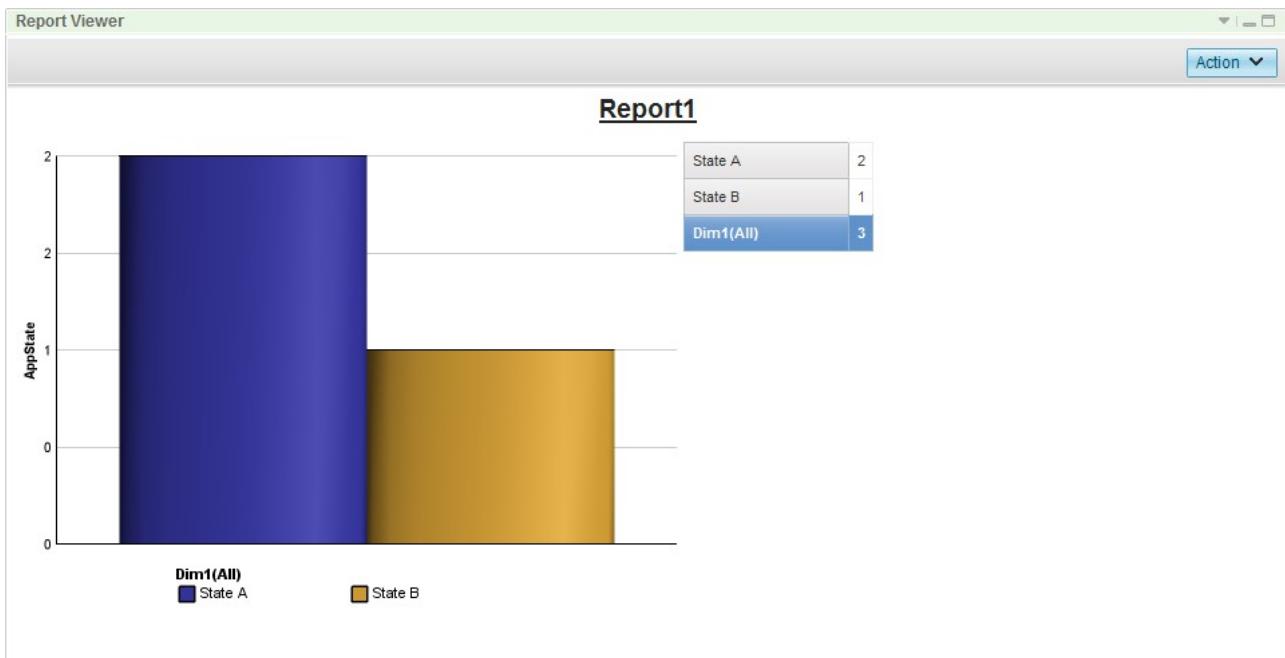
In its settings, it asks for a "report". This report is created/defined in the Report Designer widget.



When the Select button is pressed to select a report a tree dialog is displayed from which the report can be found:

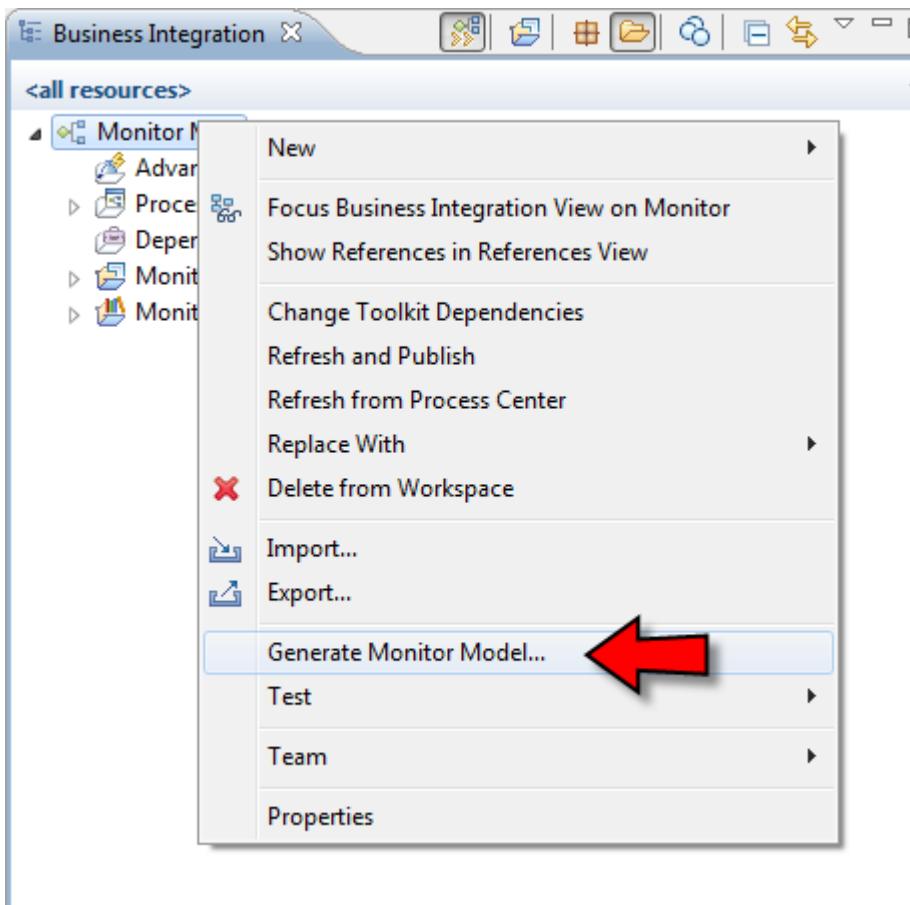


Once selected, a bar chart display is shown.



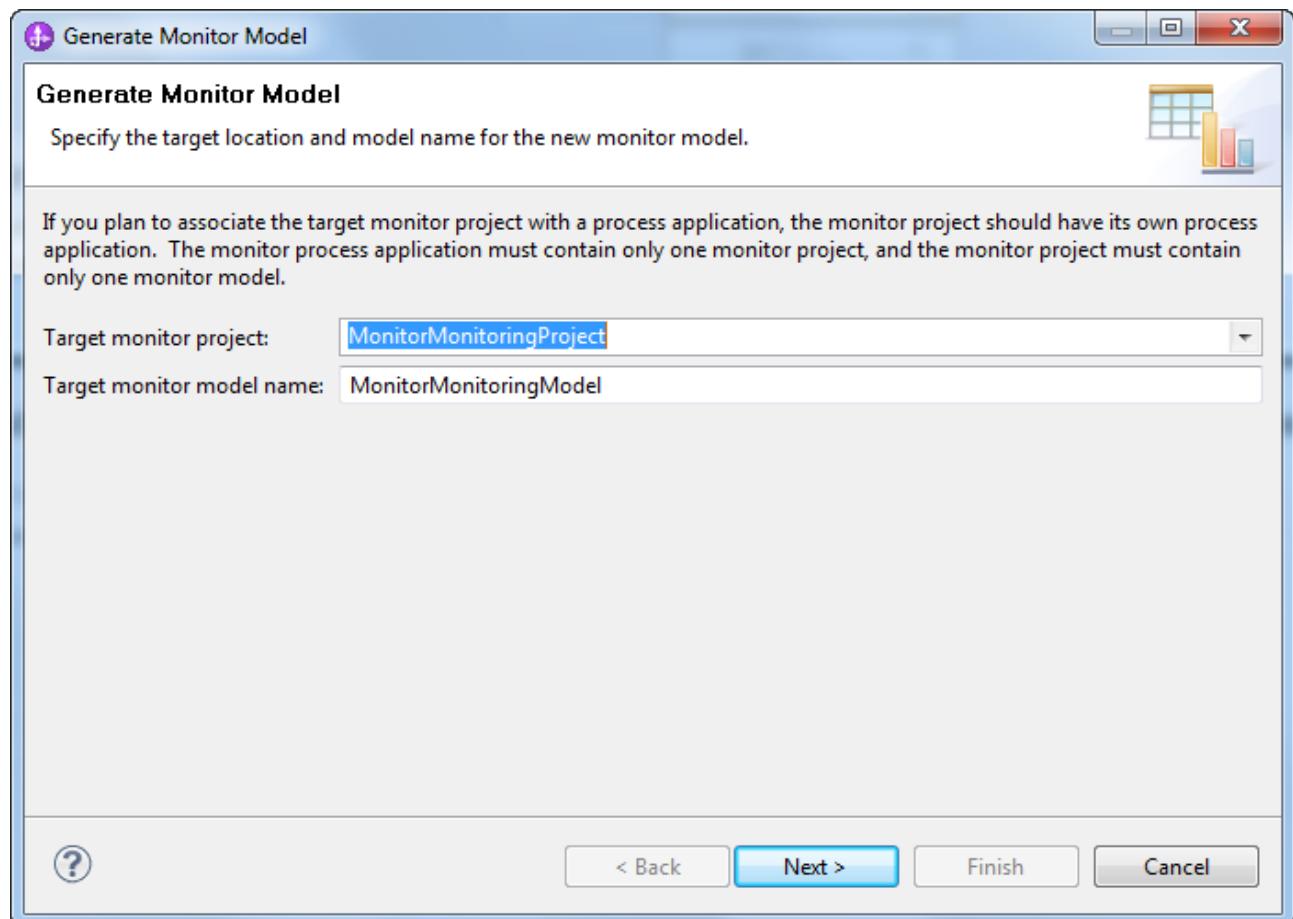
## Generated Monitor Models for a Process Application

When a process application is built within Process Designer, we have the option of importing that application into Integration Designer. Once available in ID, we can generate a Monitor Model using that Process Application as a source. Right clicking on the Process App produces a menu entry that includes:

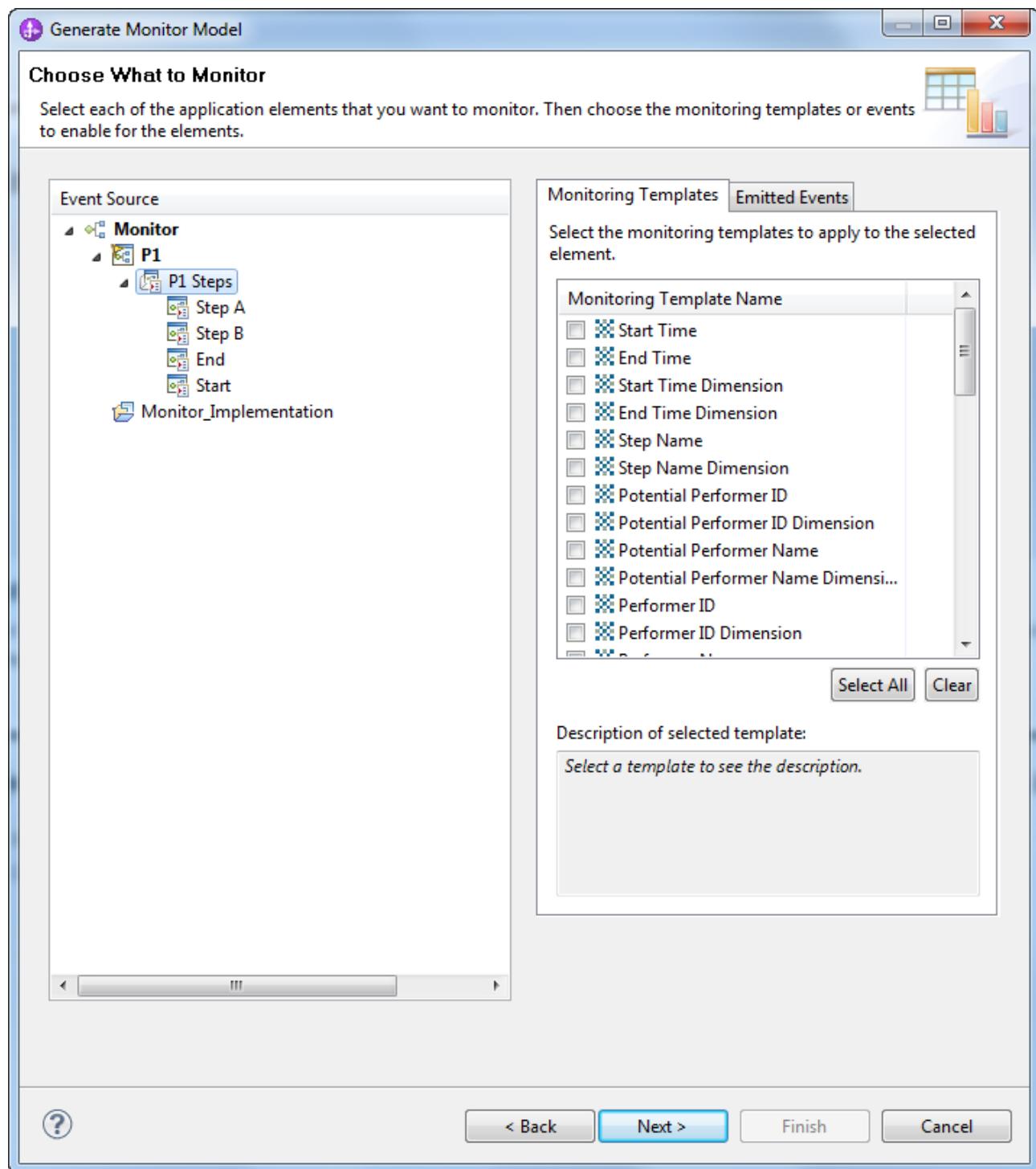


If we select "Generate Monitor Model" a dialog is presented into which the names of the Monitor

Project and the Monitor Model can be entered:



Next we are offered the opportunity for the Monitor Model to be constructed for us.



Once we have a monitor model built, we can build the Java EE projects and deploy them as normal.

## ***Instance Metrics***

### **BPD level**

- Active Step Names – A list of the steps currently active in the process. The list is a comma separated set of values.
- Aux Active Step Instance Ids – A list of the ids for active steps in the process
- Aux Last Completed Step Instance ID – The id of the last completed step

- Aux Last Completed Step Name – The name of the last completed step
- Aux Last Started Step Instance ID – The id of the last started step
- Aux Last Started Step Name – The name of the last started step
- Aux Starting Process Instance ID – Private key for the monitoring context
- <BPD> Instance ID – The BPD instance ID
- COMPLETED
- CreationTime – The date/time that the monitor context was created
- End Time – The end time of the process
- Snapshot ID – The snapshot id of the process
- Snapshot Name - The snapshot name of the process
- Start Time – The start time of the process
- State – The current state of the process step – COMPLETED, RUNNING,??
- Steps Active – The number of currently active steps in the process.
- Steps Completed – The number of completed steps.
- Termination Time – The date/time that the monitor context was terminated
- Total Time (Clock) – The wall clock time of the process
- <BPD> Steps\*
- <Variables> - Exposed tracked variables.

## <BPD> Steps Level

Metric ID	Name	Description
Step_Instance_ID	Step Instance ID	Monitoring Context key
	Aux Last Event Time	
COMPLETED	COMPLETED	Whether or not the monitoring context has been closed/completed.
bmon_Cost	Cost	KPI cost of running this process step
CreationTime	CreationTime	The time the activity was created.
bmon_EndTime	End Time	End time of this process step
bmon_ExecutionTime_Clock	Execution Time (Clock)	Execution time for running this process step
bmon_LaborCost	Labor Cost	Labor cost of running this process step
bmon_PerformerID	Performer ID	The id of the person who performed the step.
bmon_PerformerName	Performer Name	The name of the person who performed the work
bmon_PotentialPerformerID	Potential Performer ID	The id of the user who can work on this step
bmon_PotentialPerformerName	Potential Performed Name	The name of the user who can work on this step
bmon_ResourceCost	Resource Cost	Resource cost of running this process step

bmon_Rework	Rework	Rework KPI
Snapshot_ID	Snapshot ID	The snapshot ID of this step
Snapshot_Name	Snapshot Name	The name of the snapshot of this step
bmon_StartTime	Start Time	Start time of this process step
bmon_State	State	The state of the step. Values seen include: <ul style="list-style-type: none"><li>• COMPLETED</li></ul>
bmon_Step_Name	Step Name	The name of this process step
TerminationTime	TerminationTime	The time that the activity ended.
bmon_TotalTime_Clock	Total Time (Clock)	Total Time for running this process step
bmon_ValueAdd	Value Add	Value Add KPI
bmon_WaitTime_Clock	Wait Time (Clock)	Wait time for running this process step
bmon_f_<Variable>	<Variable>	Exposed variable.

## KPIs

- <Step> Average Cost
- <Step> Average Resource Cost
- <Step> Average Labor Cost
- <Step> Average Total Time (Clock)
- <Step> Average Wait Time (Clock)
- <Step> Average Execution Time (Clock)
- <Step> Average Rework
- <Step> Average Value Add
- <Step> Count
- <BPD> Average Total Time (Clock)
- <BPD> Count

## Working with Changed Variables

Consider a BPD variable called "amount" that defines the amount of a sale within a process. This variable is flagged as "tracked". Imagine we wish to fire an outbound event from Monitor when the amount is changed to be greater than \$5000.

This seems simple enough. We can define a trigger that is sourced when the value of the "amount" metric changes. For example:

**Trigger Details**  
Edit the details of the trigger, which detects an occurrence and initiates an action in response.

ID:	<input type="text" value="AmountChanged"/> <a href="#">Edit...</a>
Name:	<input type="text" value="AmountChanged"/>
Description:	<input type="text"/>

Trigger is repeatable  
 Terminate monitoring context

**Trigger Sources**  
Specify the source of this trigger.

Source Type	Source
Value change	 amount

[Add](#) [Remove](#)

**Trigger Condition**  
Specify the condition that determines whether the trigger will fire.

 |

Unfortunately, there is a problem. The architecture of the events sent to Business Monitor is that EVERY event contains the *current* value of the "amount" metric. This has the side effect of causing the "amount" metric to flag as changed continuously. The "Value changed" moniker of the trigger source is misleading as it is actually when the value is updated ... even if the actual value of the metric doesn't change.

This would mean that our trigger would fire every time BPM publishes an event which is at the start and end of a BPD activity ... even if that activity doesn't change the value of the "amount" variable.

Fortunately, there is a solution. What we do is introduce a new metric that contains the "old" value of the variable. We will call this new metric "amount\*".

**Metric Details**

Edit the details of the metric, which is a holding spot for information used in other calculations.

ID:	* amountPRE	<a href="#">Edit...</a>
Name:	amount*	
Description:	The original value of the "amount" metric.	
Type:	Decimal	<a href="#">▼</a>

A value is required for this metric

Default Value: 0 [Edit...](#)

This metric can be used for sorting

Hide from dashboards

**Metric Value Expressions**

Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.

Trigger	Expression
AmountChanged	X+Y :=? bmon_f_amount

[Add](#) [Remove](#)

We say that this metric is updated when our "AmountChanged" trigger occurs. Now we can add a condition to our trigger which reads:

**Trigger Details**  
Edit the details of the trigger, which detects an occurrence and initiates an action in response.

ID: \* AmountChanged

Name: AmountChanged

Description:

Trigger is repeatable  
 Terminate monitoring context

**Trigger Sources**  
Specify the source of this trigger.

Source Type	Source
Value change	 amount

**Trigger Condition**  
Specify the condition that determines whether the trigger will fire.

bmon\_f\_amount != amountPRE



What this says is that we only fire the "AmountChanged" trigger when the old amount is not the same as the new amount (i.e. when the value changes).

## Sample Monitor Solutions

Here we capture some sample monitor solutions.

### Task Escalation alerts

Consider the notion of a human task. We may wish this task to be completed within a certain period of time. If it isn't, then we may wish to raise an alert to have it looked into. This is a good use case for Business Monitor.

When a BPD runs, a model is built of each of the steps performed by the process. This includes when the step starts and when it ends. Our design for the solution is broken into two phases. The first will be the determination that a task is late and the second on how an alert will be issued.

To determine if a task is late, we will create a new trigger that will fire:

- One time after the escalation interval has passed
- Will check that the step name is one we are escalating
- Will check that the step has not yet completed

An example trigger definition would be:

```
<trigger displayName="Task Overdue" id="Task_Overdue" isRepeatable="false">
```

```

<evaluationTime hours="0" minutes="2"/>
<gatingCondition expression="bmon_Step_Name = 'Receive Claim' and fn:empty(bmon_EndTime)"/>
</trigger>

with a corresponding outbound event definition off:

<outboundEvent displayName="Task Escalation" id="Task_Escalation"
extensionName="ActionServicesEvent" rootElement="cbe:CommonBaseEvent">
 <eventPart displayName="escalation" id="escalation" path="cbe:CommonBaseEvent/tns2:escalation"
type="tns2:escalation"></eventPart>
 <map>
 <trigger ref="Task_Overdue"/>
 <outputValue>
 <assignments>
 <assignment leftValue="Task_Escalation/extendedData/BusinessSituationName"
rightValue="'Escalation'"/>
 <assignment leftValue="Task_Escalation/escalation/tns2:processId"
rightValue=".../Process_Instance_ID "/>
 <assignment leftValue="Task_Escalation/escalation/tns2:taskId" rightValue="Task_ID "/>
 <assignment leftValue="Task_Escalation/escalation/tns2:stepName"
rightValue="bmon_Step_Name "/>
 <assignment leftValue="Task_Escalation/escalation/tns2:currentOwner"
rightValue="bmon_PerformerName "/>
 <assignment leftValue="Task_Escalation/escalation/tns2:processName"
rightValue=".../Process_Name "/>
 </assignments>
 </outputValue>
 </map>
</outboundEvent>

```

When an alert is seen notifying someone that a task is overdue, we must now ask ourselves what should that alert contain? Options include:

- The Process Instance ID that contains the task
- The Task ID for the actual task
- The name of the process
- The name of the step of the process
- The current task owner (if one exists)

Of these, the Process Instance ID, the Task ID and the name of the process are not generated automatically by the BPM monitor model generation and need to be explicitly added.

See also:

- Alerts and Actions
- Capturing the Task ID of a task
- Capturing the Process ID of a process
- Capturing the Process Name of a process

## Monitor Notes

The following area is not part of the book but is instead an area used to log the author's thoughts and areas to consider.

Comparing QNames in expressions: <http://ibmforums.ibm.com/forums/thread.jspa?threadID=567118&tstart=0>

see also: fn:resolve-QName

# WebSphere Application Server

IBPM executes on a WebSphere Application Server environment. By and large there isn't much that the users of the product have to know but this section of the book will list some of the more pertinent areas in case the need arises.

## ***Profile Management***

Profiles are configurations of WAS servers. In general, profiles are created with the GUI PMT (Profile Management Tool) utility. This provides a wizard driven configuration of the profile instances. Through PMT WAS servers of a variety of types may be created.

### ***References – Profile Management***

- DeveloperWorks - [The Support Authority: Take the confusion \(and errors\) out of managing profiles for WebSphere Application Server](#) - 2010-07-14

## **Command line profile management**

A command called `manageprofiles` is available to manage profiles. This command has a number of options:

```
manageprofiles -listProfiles
```

Will list the available profiles.

```
manageprofiles -delete -profileName Name
```

Will delete a named profile.

## **Switching a server from production to development mode**

When profiles (including Process Center) are created, the servers are default defined to run in production mode. This is useful (naturally) for a production system but while developing solutions, this can get in your way. Running the server in development mode relaxes some constraints such as the ability to install a new application while there are existing instances of the old application already around.

From within the Admin console, there is a check box that specifies whether or not we run in development mode (default is **not** development).

The screenshot shows the Integrated Solutions Console interface. On the left, there's a navigation sidebar with various categories like Welcome, Guided Activities, Servers, Applications, Services, Resources, Security, Environment, Integration Applications, System administration, Users and Groups, and Monitoring and Tuning. The 'Servers' category is expanded, showing Server Types (WebSphere application servers, WebSphere MQ servers, Web servers). The main panel is titled 'Application servers' and shows 'server1'. It has tabs for Runtime and Configuration, with Runtime selected. Under 'General Properties', there are fields for Name (server1) and Node name (win7-x64Node01), and checkboxes for Run in development mode (checked), Parallel start, and Start components as needed. A red arrow points to the 'Run in development mode' checkbox.

## ***Recovering from a failed adapter install***

If we have a module in ID that is directly deployed to PS that contains a JCA adapter which contains a configuration error and the module is deployed, the module will fail to deploy but will leave artifacts in the run-time that prevents **any** subsequent re-installs of that application. This is believed to be a bug and there is indeed plans for a fix. However, if it happens, here is a workaround:

- Stop the WAS server
- Delete files under

<WAS\_HOME>/profiles/*profilename*/config/cells/*cellname*/cus/<*applicationName*>

<WAS\_HOME>/profiles/*profilename*/config/cells/*cellname*/blas/<*applicationName*>

- Delete all files under:

<*profile\_root*>/wstemp/

<*profile\_root*>/temp/

<*profile\_root*>/config/temp

<*profile\_root*>/logs/dmgr

- Restart the server

## ***Recovering from XA recovery***

In rare circumstances, XA recovery may fail. To clean in-flight transactions and transaction logs, shutdown WAS and in the <Profile>/tranlog/Cell/Node/server/transaction directory, delete the contents of the folders called partnerlog and tranlog.

## ***WAS Security***

WAS Security is the topic of how the WAS server itself manages security artifacts.

See also:

- DeveloperWorks - [WebSphere Application Server V7 advanced security hardening, Part 1: Overview and approach to security hardening](#) – 2010-12-01
- DeveloperWorks - [WebSphere Application Server V7 advanced security hardening, Part 2: Advanced security considerations](#) – 2010-12-01
- DeveloperWorks - [IBM WebSphere Developer Technical Journal: SSL, certificate, and key management enhancements for even stronger security in WebSphere Application Server V6.1](#) – 2006-12-06
- Redbook - [WebSphere Application Server V7.0 Security Guide](#) - 2011-04-25
- Redbook - [IBM WebSphere Application Server V6.1 Security Handbook](#) – 2006-12-28

## **Virtual Member Manager**

The WebSphere Application Server (WAS) has a concept called the Virtual Member Manager. This is a security model that allows federated access to multiple authentication and user management systems. This is also known as the **federated user repository**.

Part of the concept of this is the notion of a *realm*. A realm is a domain of user identities. A VMM instance corresponds to a realm. Associated with a realm is a base entry which forms the root of a hierarchical structure for that realm.

Out of the box, WAS provides support for VMM to include three types of repository for users. These are:

- A file based repository
- One or more LDAP repositories
- A JDBC database repository

There are some restrictions when using VMM:

- The userids found in each repository must be unique
- If a single repository in the realm is down, all access is denied (by design)
- Only a single realm is supported

Using wsadmin, we can create user definitions using VMM. See AdminTask.createUser().

See also:

- DeveloperWorks - [IBM WebSphere Developer Technical Journal: Expand your user registry options with a federated repository in WebSphere Application Server V6.1](#) – 2007-01-24
- DeveloperWorks - [Sample virtual member manager custom adapter for WebSphere Application Server Version 6.1](#) – 2011-06

## ***WebSphere Application Server WEB 2.0 Feature Pack***

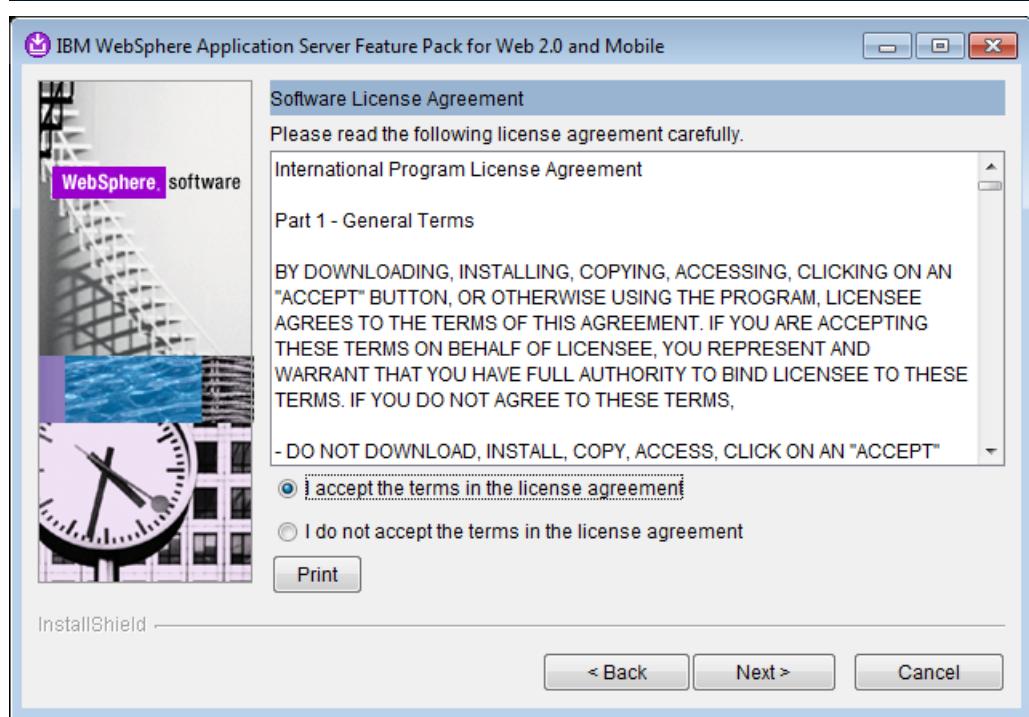
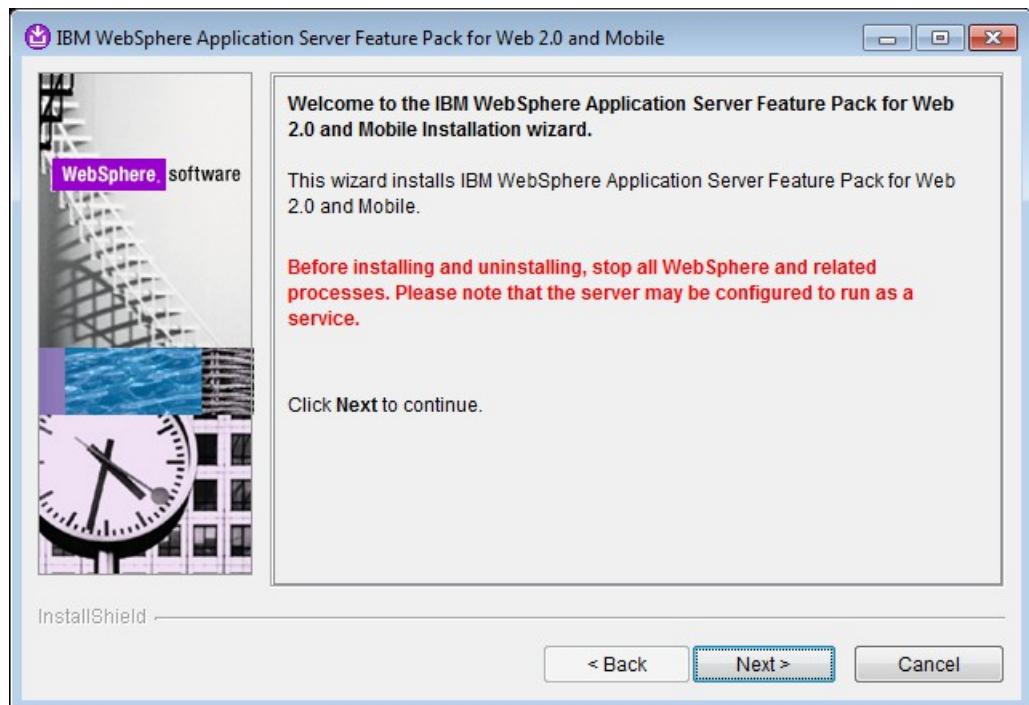
This feature pack is available for download here:

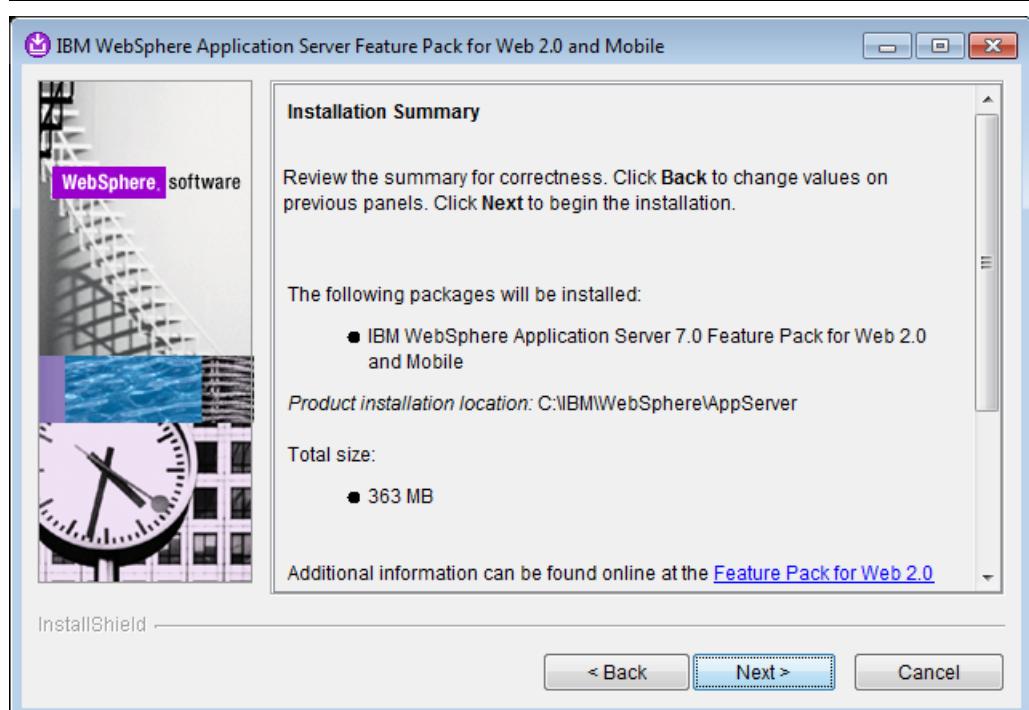
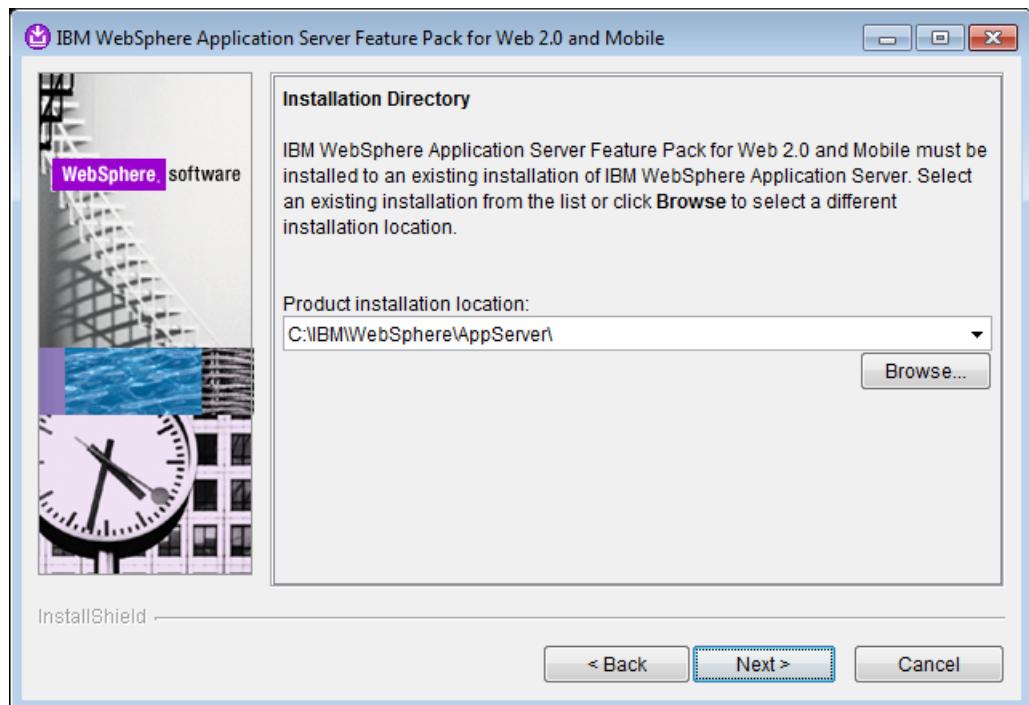
[WebSphere Application Server Feature Pack for Web 2.0 and Mobile](#)

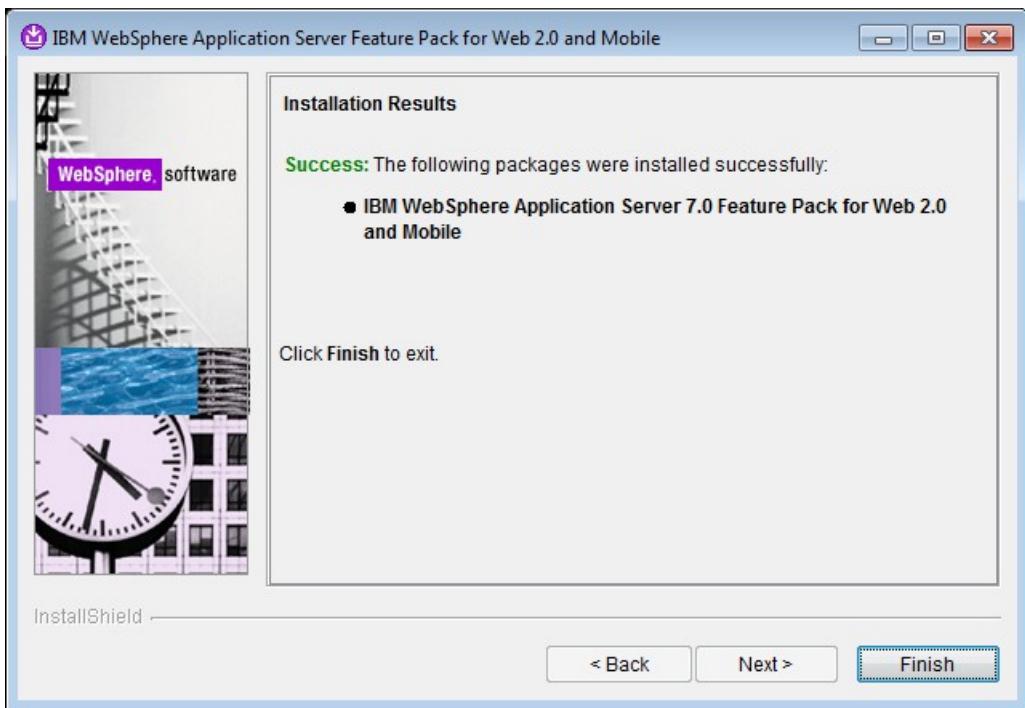
At the time of writing (03-2012) the version available was 1.1 and had a download size of about 264 Mbytes.

To install the feature pack, extract the ZIP and run the command:

```
install -is:javahome <AppServer>\java
```







Following a successful installation, running the WAS versionInfo command will show that the component is installed:

Installed Product	
Name	Web 2.0 and Mobile Feature Pack
Version	1.1.0.0
ID	WEB2MOBILE
Build Level	web21122.15
Build Date	5/25/11
Architecture	Intel (32 bit)

## See Also:

- RedBook - [Building Dynamic Ajax Applications Using WebSphere Feature Pack for Web 2.0](#) - SG24-7635-00

## AJAX Proxy

Another feature provided as part of the Web 2.0 Feature Pack is known as the AJAX Proxy. This is a reverse proxy technology that allows a client to pass a service request through the proxy that will then be forwarded onwards to other servers. This effectively allows a web client to connect out to controlled external servers bypassing the sandbox rule of the browser connections.

For full details of the AJAX proxy, consult the InfoCenter. Here we will describe using the AJAX proxy as a stand-alone deployed application.

The AJAX proxy is governed by a control file with the name the `proxy-config.xml`. Within IBM BPM, this file can be found in:

```
<Root>/BusinessSpace/mm.runtime/config
```

To route through the proxy, target:

```
http://<hostname>:<port>/mum/proxy/<protocol>/<host>:<port>/<url>
```

For example:

```
http://localhost:9081/mum/proxy/http/www.neilkolban.com
```

By default, the proxy configuration is "locked down" to prevent access to the majority of target servers. One can modify the file and update the configuration.

```
AdminTask.updateBlobConfig('[-clusterName "<Cluster Name>" -propertyFileName "<Path>/proxy-config.xml" -prefix "Mashups_"]')
AdminConfig.save()
```

After running this command, it will report:

```
'updateBlobConfig is executed successfully'
```

Here is a concrete example.

Imagine my BPM server is running on `localhost:9081`. Imagine my Business Monitor server is running on `localhost:9085`. My browser, which is served from pages from BPM, wishes to make REST calls to Monitor ... how can this be achieved?

Here is an example REST call:

```
http://localhost:9085/rest/bpm/monitor/models
```

This needs to be made from a web page loaded from `localhost:9081`.

By using the Ajax proxy, we will end up sending the REST request to:

```
http://localhost:9081/mum/proxy/http/localhost:9085/rest/bpm/monitor/models
```

In the `proxy-config.xml`, we added the following:

```
<proxy:policy url="http://localhost:9085/*" acf="none" basic-auth-support="true">
 <proxy:actions>
 <proxy:method>GET</proxy:method>
 </proxy:actions>
</proxy:policy>
```

Then the `updateBlobConfig()` command was run and the nodes synchronized. Now the request to access the monitor models works.

This is an XML file with a number of types of entries.

See also:

- TechNote: [Troubleshooting IBM Business Process Manager: "403 BMWPX0006E: The URL you tried to access through the proxy is not allowed"](#) – 2011-12-12
- [Testing the AJAX Proxy in WebSphere Application Server 8.0](#) - 2013-01-22

`proxy:mapping`

## **JAX-RS – REST Services in Java**

REST based services have become more and more common. The implementation of a REST based service provider in Java was originally built upon the notion of Servlets but as REST became more pervasive, a better solution was proposed. This became known as JAX-RS and is documented in JSR-311.

This section provides a primer and notes on JAX-RS and, where relevant, how they might be leveraged with IBPM.

To start, we will examine some of the core concepts of the JAX-RS standard. Firstly, JAX-RS is based on Java annotations meaning it will only work in the later releases of Java.

We will start with the notion of the Root resources. A Root resource is a class that is associated with the annotation of "`@Path`".

For example:

```
@Path(value="/myPath")
public class MyClass
{
}
```

It appears that an annotation of:

```
@Path(value="/myPath")
```

is equivalent to:

```
@Path("/myPath")
```

Methods contained within such a class can be flagged as the handlers for various HTTP requests. These are termed *resource methods*. For example:

```
@Path("/myPath")
public class MyClass
{
 @GET
 public String myFunc()
 {
 ...
 }
}
```

Signifies that if an HTTP GET request were made to “/myPath” then the Java method called “myFunc ()” would be invoked.

An additional feature known as *sub-resource methods* can be defined that looks as follows:

```
@Path("/myPath")
public class MyClass
{
 @GET
 public String myFunc()
 {
 ...
 }

 @GET
 @Path("/myPart")
 public String my2ndFunc()
 {
 ...
 }
}
```

In this case, an HTTP request to “/myPath/myPart” would cause the function called “my2ndFunc ()” to be invoked.

One further qualifier is provided that looks like:

```
@Path("/myPath")
public class MyClass
{
 @GET
 public String myFunc()
 {
 ...
 }

 @GET
 @Path("/myPart")
 public String my2ndFunc()
 {
 ...
 }

 @Path("/{myParam}/my3rdFunc")
 public String my3rdFunc(@PathParam("myParam") String myParam)
 {
 }
}
```

```
 } ...
}
```

Using this style, we can access the “my3rdFunc ()” function using “/myPath/someParam/my3rdFunc”. This style is known as a Sub-resource locator.

Another style of URL is known as matrix param:

```
@Path("/myPath")
public class MyClass
{
 @GET
 public String myFunc(@MatrixParam("fieldA") String fieldA,
 @MatrixParam("fieldB") int fieldB)
 {
 ...
 }
}
```

would handle requests such as:

```
/myPath;fieldA=xyz;fieldB=123
```

Similar to the above, query parameters can also be supplied:

```
@Path("/myPath")
public class MyClass
{
 @GET
 public String myFunc(
 @QueryParam("fieldA") String fieldA,
 @QueryParam("fieldB") int fieldB)
 {
 ...
 }
}
```

which would handle requests such as:

```
/myPath?fieldA=hello&fieldB=123
```

For method annotations, the following may be supplied corresponding to the HTTP commands:

- @GET
- @POST
- @PUT
- @DELETE
- @HEAD

HTTP Header parameters can be obtained by name in a method signature:

```
public String myFunc(@HeaderParam("User-Agent") String userAgent, ...)
```

When invoked, the value of the "User-Agent" HTTP Header will be passed in as the userAgent parameter.

To access all headers, the following can be used:

```
public String myFunc(@Context HttpHeaders headers, ...)
```

When returning data, a return type of javax.ws.rs.core.Response can be used to set status code as well as HTTP Headers.

A request provides an indication of the data being passed into it. This is passed in with the HTTP header known as “Content-Type”. A matching method can be declared using the @Consumes

annotation:

```
@POST
@Consumes(value="text/xml")
public String doSomething()
{
 ...
}
```

Correspondingly, a method that returns a specific style of data can be declared to produce such with the @Produces annotation. An HTTP request with an HTTP header of “Accept” will be used to match the function.

```
@GET
@Produces(value="text/xml")
public String doSomethingElse()
{
 ...
}
```

Once the classes that are to be exposed as REST resources have been defined, they need to be “packaged” into a group associated with the an application that actually exposes them.

```
public class MyApplication extends javax.ws.rs.core.Application {
 public Set<Class<?>> getClasses()
 {
 Set<Class<?>> classes = new HashSet<Class<?>>();
 classes.add(MyClassA.class);
 classes.add(MyClassB.class);
 return classes;
 }
}
```

What we have here is a Java class that extends the JAX-RS Application class. This class overrides the method called “getClasses ()” which returns a set of classes that are to be exposed as JAX-RS resources. In the example above, two classes are exposed.

The following maps the packages for various annotations:

Annotation	Package
@Path	javax.ws.rs.Path
@PathParam	javax.ws.rsPathParam
@HeaderParam	javax.ws.rs.HeaderParam
@Context	javax.ws.rs.core.Context
@MatrixParam	javax.ws.rs.MatrixParam
@QueryParam	javax.ws.rs.QueryParam
@CookieParam	javax.ws.rs.CookieParam
@FormParam	javax.ws.rs.FormParam

See also:

- WAS 8.5 InfoCenter – [Overview of IBM JAX-RS](#)
- RedBook - [Building Dynamic Ajax Applications Using WebSphere Feature Pack for Web 2.0](#) - SG24-7635-00
- [Wikipedia on JAX-RS](#)
- [JAX-RS: Developing RESTful Web Services in Java](#)
- DeveloperWorks - [Developing JAX-RS 1.1 RESTful Services in Rational Software Architect V8 for deployment to WebSphere Application Server V8](#) – 2011-11-02
- DeveloperWorks - [Develop an Apache HttpClient client for Android to a JAX-RS web service](#) – 2011-10-11
- DeveloperWorks - [Build RESTful web services with Java technology](#) – 2011-10-07
- DeveloperWorks - [Comment lines: Combining Dojo and JAX-RS to create a RESTful service](#) – 2010-07-14

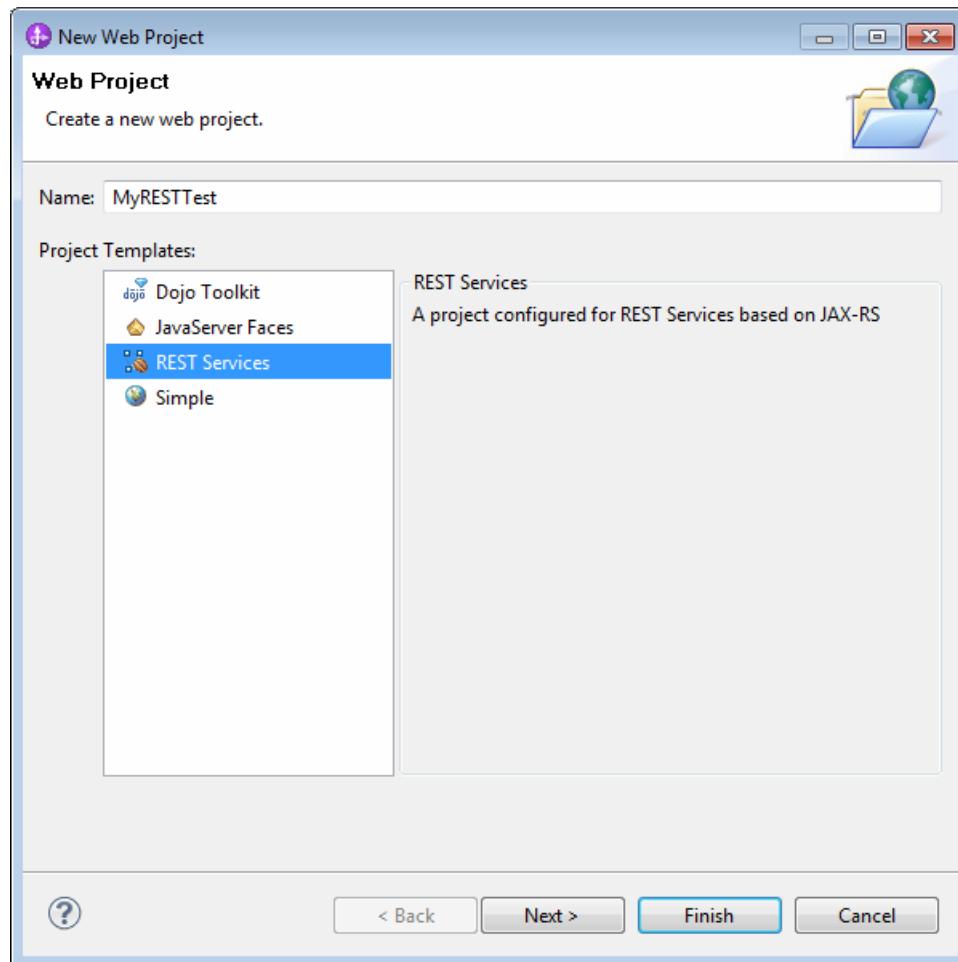
- DeveloperWorks - [Create RESTful Web services with Java technology](#) – 2010-02-23
- DeveloperWorks - [RESTful Web services: The basics](#) – 2008-11-06

## JAX RS in WebSphere Application Server

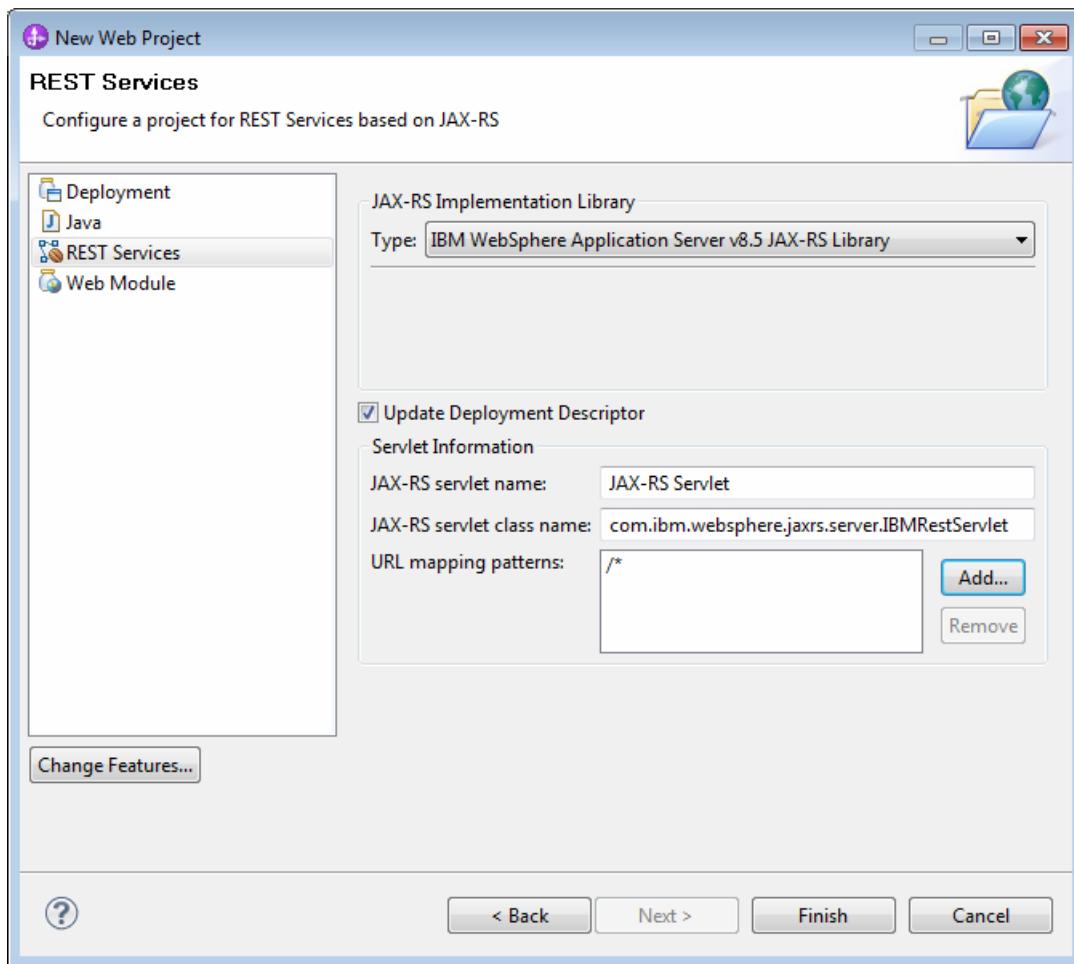
IBM WebSphere Application Server (WAS) provides an implementation of JAX-RS as part of the Web 2.0 Feature Pack and also as part of the base WAS 8.5 environment.

Here is a sample of building a JAX-RS exposed service in IBM Integration Designer.

1. Create a new Web Project in IID. I called my project "MyRESTTest".
2. In the project settings, select "REST Services" as a template for project construction.



3. Change the URL mapping patterns to be "/ /\*"



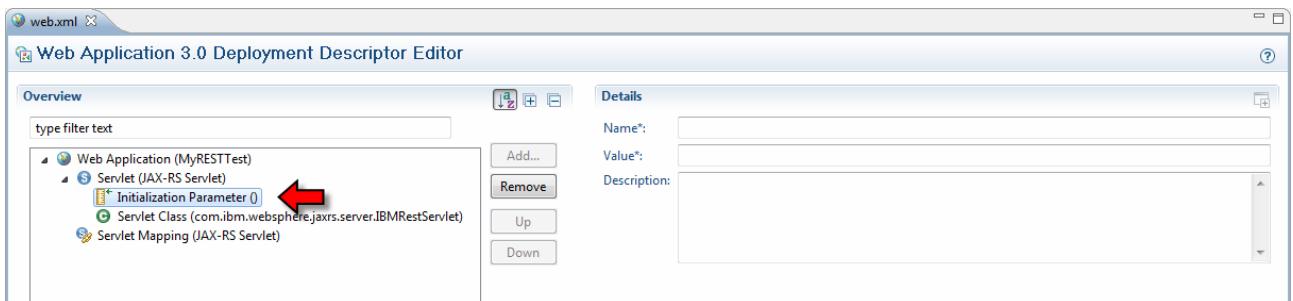
4. Create a Java class that implements a JAX-RS exposed service. This is the Java class that contains the code that will be invoked when the incoming REST request arrive. Notice that the class is annotated with the JAX-RS annotations.

Here is an example:

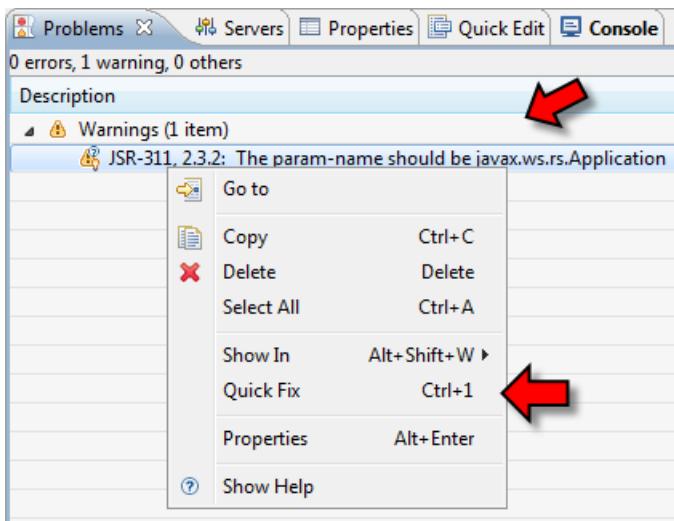
```
package com.ibm.demo;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/rating")
public class Rating {
 @GET
 @Path("/id")
 public String getId() {
 System.out.println("/rating/id has been called!");
 return "Hi from /rating/id";
 }
}
```

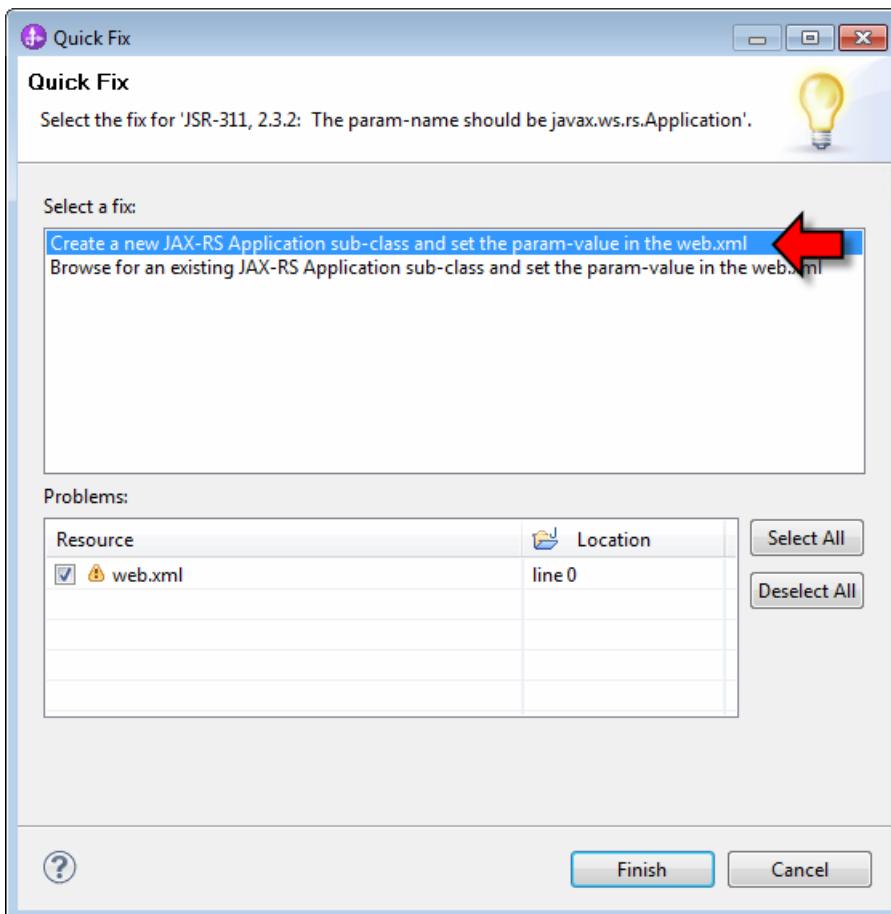
5. Open the Web deployment descriptor file called "web.xml". Add an empty "Initialization Parameter" and save the file. There is no question that this feels like an odd instruction. We could explicitly have added the values we need to the parameters or have had a better way of adding the details, but it "is what it is".



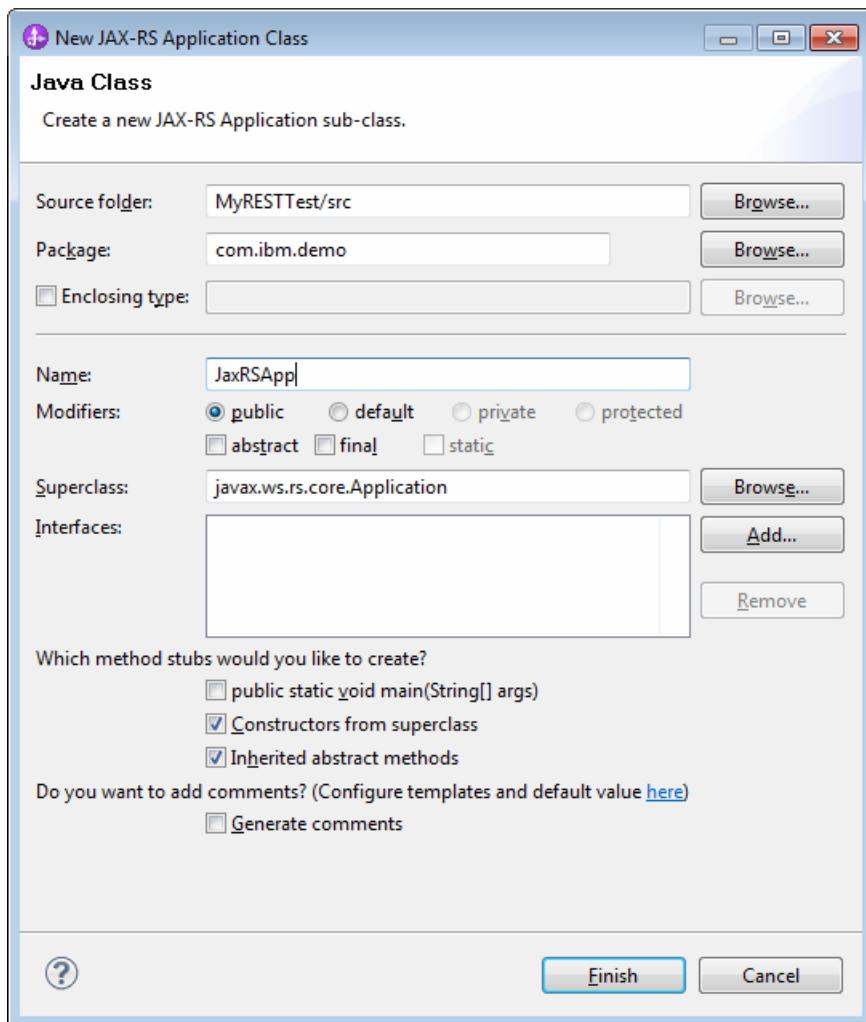
6. Execute the "Quick Fix" wizard with the "JSR-311 2.3.2" warning selected.



7. In the resulting wizard, select the option to create a new JAX-RS Application class.



- Define the new class name and package to be created. In my example I called the Application class "JaxRSApp".



- Implement the body of the newly created class. The Java source file generated is the "bootstrap" for our JAX-RS solution. We need to override a method called "getClasses" to return a Java Class reference to each of the implementations of JAX-RS function.

In my example, I coded:

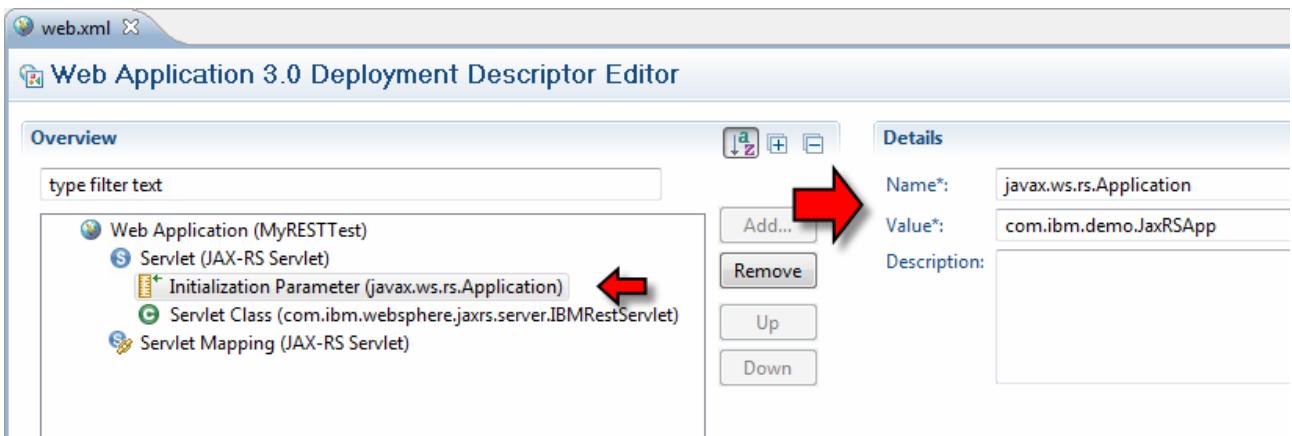
```
package com.ibm.demo;

import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.core.Application;

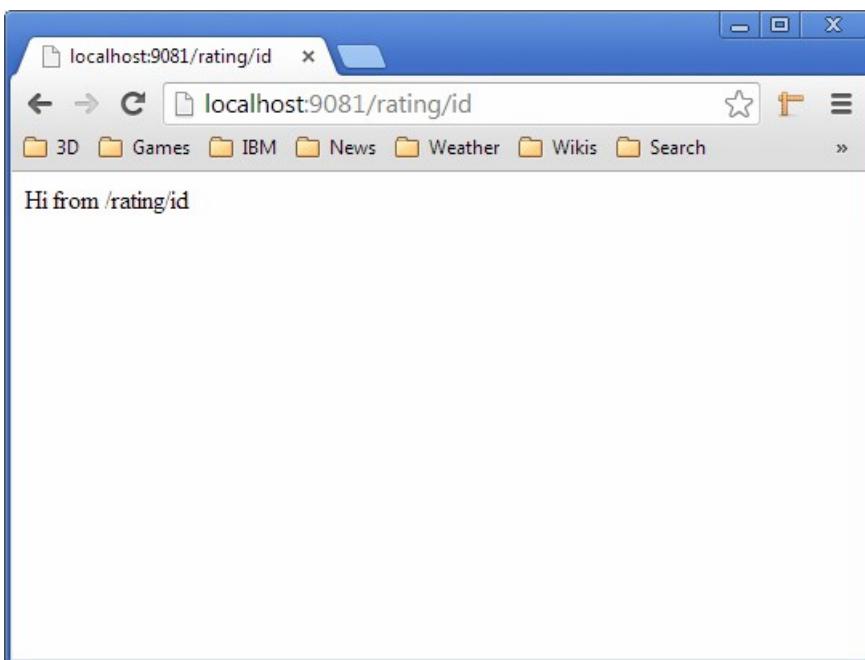
public class JaxRSApp extends Application {
 @Override
 public Set<Class<?>> getClasses() {
 Set<Class<?>> classes = new HashSet<Class<?>>();
 classes.add(Rating.class);
 return classes;
 }
}
```

Note that the class called "Rating.class" is the name of my class which contains the implementation of the exposed JAX-RS REST service.

- (Optional) Examine the web.xml deployment description and see that it now points to our App as the loader for the JAX-RS environment.



11. Deploy the application to a suitable WAS server.
12. Test the REST service. We can use either a browser or a tool such as soapUI.



## ***Deploying Java EE applications to WAS***

There are times when we may wish to deploy Java EE applications to the WAS server hosting Process Server. We can use the WAS admin console to achieve this task but there is an intriguing alternative. WAS provides a monitored directory concept. If a Java EE application is placed in that directory, WAS installs that application for us. By default, this capability is disabled and must be switched on. From the WAS admin console, switch to Applications > Global deployment settings and switch it on:

The screenshot shows the 'Global deployment settings' configuration page. On the left, there's a navigation tree with 'View: All tasks' selected. Under 'Applications', 'Global deployment settings' is chosen. The main panel displays the 'Monitored Directory Deployment' section. It contains a checked checkbox labeled 'Monitor directory to automatically deploy applications'. Below it are fields for 'Monitored directory' set to '\${USER\_INSTALL\_ROOT}/monitoredDeployableApps' and 'Polling interval' set to '5 seconds'. At the bottom are 'Apply' and 'Reset' buttons.

After enabling this property, the server must be restarted.

For stand-alone servers, the default location for the monitored directory is:

```
<profileRoot>/monitoredDeployableApps/servers/server1
```

## ***WAS Messaging***

### **Foreign Connections**

There are times when we wish to setup foreign connections between two SI Buses. Here are some notes.

On Machine B we have a Bus called "MONITOR.win7-x64Node01Cell.Bus". This has a queue defined on it called "TargetQ". We create a JMS Queue Connection Factor called "jms/Bus2QCF" that points to the Bus. We also create a JMS Queue Definition called "jms/TargetQ" that points to the "TargetQ" owned by SI Bus.

Property	Bus 1	Bus 2
BOOTSTRAP_ADDRESS	9810	2814
SIB_ENDPOINT_ADDRESS	7278	7281

### Hermes Settings

	Bus 1	Bus 2
Session Name	Bus1	Bus2
binding	jms/Bus1QCF	jms/Bus2QCF
initialContextFactory	com.ibm.websphere.naming.WsnInitialContextFactory	com.ibm.websphere.naming.WsnInitialContextFactory
providerURL	iiop://localhost:9810	iiop://localhost:2814

localhost:7278:BootstrapBasicMessaging



# **Enterprise Content Management Integration – ECM**

There are many solutions involving BPM that involve working with document and file content such as Microsoft Word, PDFs, Excel spreadsheets, image files and other pre-existing information items. Instances of document types such as these may be needed as part of a process instance. Consider an insurance claim for a car accident. This may involve an initial claim form, a police report, pictures of the accident and faxes from garages with quotes for repair. To process a claim, a clerk may work through a process instance which must make these documents available. Obviously these documents have to be stored somewhere within an IT system and this is where the category of software product called an Enterprise Content Management (ECM) system comes into play. An ECM is a software product that explicitly manages the existence, organization, search and retrieval of instances of documents. A variety of vendors make a variety of such ECMs. These include:

- IBM FileNet
- IBM Content Manager
- Alfresco
- EMC Documentum
- ... others

In addition, the BPM product itself provides an in-built document repository.

In this section we will discuss the support provided by IBM BPM for integrating with ECMs such that content from documents can be managed as part of the process solution.

See also:

- developerWorks - [Integrating BPM and Enterprise Content Management](#) - 2013-12-04
- developerWorks - [Integrating IBM Business Process Manager V8 with an Enterprise Content Management system](#) - 12-2012

## ***Content Management Interoperability Services – CMIS***

If we consider what we want from an arbitrary ECM we will find a common set of required functions. Capabilities that we obviously want are:

- The ability to store documents
- The ability to get lists of documents
- The ability to retrieve documents
- The ability to delete documents

Historically, each provider of an ECM supplied their own sets of APIs to be able to interact with it. Each API was perfect for interacting with a specific brand of ECM however if we wished to change ECM providers, we would be stuck with the job of re-coding our applications to use the different APIs. What we need is an API that is an industry standard that each vendor that implements an ECM can expose. If we can define such an API then we need only write our application that interact with an abstract ECM once and then simply choose an ECM provider to use based on other qualities such as performance, capacity, price or other features.

Content Management Interoperability Services (CMIS) is an open standards protocol from the standards body called OASIS. CMIS provides a vendor neutral API for interacting with ECMs. CMIS is what IBM BPM uses to interact with back-end ECM systems thus insulating BPM from any single provider. A number of ECM's support this standard including:

- IBM FileNet

- Microsoft SharePoint
- EMC Documentum
- Alfresco

CMIS provides the following functions and here we indicate which ones are supported by IBM BPM:

- `getRepositories` – Not exposed
- `getRepositoryChildren` – Not exposed
- `getTypeDescendants` – Get type descendants
- `getTypeDefinition` – Get type definition
- `getChildren` – Get documents in folder
- `getDescendents` – Not exposed
- `getFolderTree` – Get folder tree
- `getFolderParent` – Not exposed
- `getObjectParents` – Not exposed
- `getCheckedOutDocs` – Not exposed
- `createDocument` – Create document
- `createDocumentFromSource` – Copy document
- `createFolder` – Create folder
- `createRelationship` – Not exposed
- `createPolicy` – Not exposed
- `getAllowableActions` – Not exposed
- `getObject` – Get document, Get folder
- `getProperties` – Not exposed
- `getObjectByPath` – Get folder by path
- `getContentStream` – Get document content
- `getRenditions` – Not exposed
- `updateProperties` – Update document properties
- `moveObject` – Move document, Move folder
- `deleteObject` – Delete document, Delete Folder
- `deleteTree` – Not exposed
- `setContentStream` – Set document content

- deleteContentStream – Not exposed
- addObjectToFolder – Add document to folder
- removeObjectFromFolder – Remove document from folder
- query - Search
- getContentChanges – Not exposed
- checkOut – Check-out document
- cancelCheckOut – Cancel check-out document
- checkIn – Check-in document
- getObjectOfLatestVersion – Not exposed
- getPropertiesOfLatestVersion – Not exposed
- getAllVersions – Get all document versions
- getObjectRelationships – Not exposed
- applyPolicy – Not exposed
- removePolicy – Not exposed
- getAppliedPolicies – Not exposed
- getACL – Not exposed
- applyACL – Not exposed

See also:

- [CMIS Specification – Version 1.0](#)
- [WikiPedia – CMIS](#)
- [CMIS at Alfresco](#)

## ***Configuring IBM BPM for an external ECM***

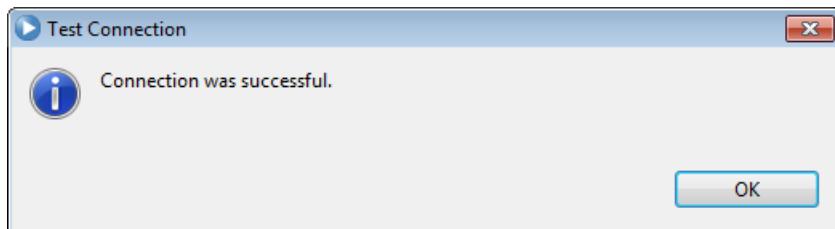
In order to integrate with an external ECM system using CMIS, you must define it within your Process Application. Within Process Designer, within the Process App Settings, you will find a tab called Servers. We create a new definition here ensuring the "Type" is "Enterprise Content Management Server".

A definition of this type has a number of attributes associated with it:

- Hostname – The host-name or TCP/IP address on which the ECM server can be found.
- Port – The TCP/IP port number on the host-name where the ECM server is listening for requests it should perform on behalf of the BPM system.
- Context Path – The relative URL part of the context path on which the ECM server is willing to accept Web Service requests for ECM work.

- Alfresco – alfresco/cmisws
- Secure Server – Whether or not the HTTPS protocol should be used for communications. HTTPS provides transport level security.
- Repository – The name of the repository.
  - Alfresco – Main Repository
- Userid – The userid used to connect to the ECM server.
- Password – The password for the user used to connect to the ECM server.

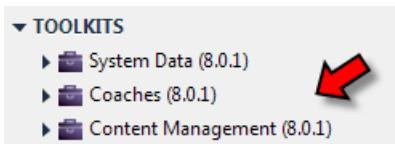
The "Test Connection" button may be used to validate the information entered. If all is well, the following dialog will be shown:



Here is a sample Alfresco definition:

A screenshot of the IBM BPM Process App Settings interface. The top navigation bar shows "Process App Settings" with a checkmark icon, "Overview", "Environment", and "Servers" tabs, where "Servers" is selected. On the left, there's a sidebar with a "Servers" section containing a list box with "Alfresco1 {localhost}" and buttons for "Add" and "Remove". The main area has two expandable sections: "Server Details" and "Server Locations". Under "Server Details", fields include "Name: Alfresco1", "Type: Enterprise Content Management Server", and a "Description" text area with placeholder text. Under "Server Locations", fields include "Environment Type: Default", "Hostname: localhost", "Port: 8080", "Context Path: alfresco/cmisws", "Secure Server: (checkbox)", "Repository: Main Repository", "Userid: admin", and "Password: \*\*\*\*\*". There is also a checked checkbox for "Always Use This Connection Information" and a "Test Connection" button at the bottom.

In order to use ECM functions within a BPM solution, we must add an IBM supplied toolkit as a dependency to our application. Add the content management toolkit which is called "Content Management" which has an acronym ID called "SYSCM":



## The Document List Coach View

After having defined a BPM process app with knowledge of an ECM, we would next turn our attention to thinking about how a user would see managed documents from our solution. IBM has supplied a number of Coach Views that perform such services.

To show documents to the user from a Coach, you can add a "Document List" Coach View to the Coach. From the palette of controls, this looks as follows:



When added to the Coach, it looks like:

Name	Date Created	Creator	Major Version Number	Actions
2012062009 21:29:24.474:037	2012/06/20 21:29:24.474:037	test	1	
2012062009 21:30:32.395:037	2012/06/20 21:30:32.395:037	test	1	
2012062009 21:31:27.968:037	2012/06/20 21:31:27.968:037	test	1	
2012062009 21:33:24.278:037	2012/06/20 21:33:24.278:037	test	1	

Add Document Refresh

It has quite a few configuration parameters:

Properties Validation Errors Where Used

**General**

**Configuration**

**Visibility** Allow create:

**HTML Attributes** Allow update:    
Open in new window:    
Number of results to show:     
Show all content:

Configure for use with:  ECM Documents

▶ BPM document options:

▶ ECM document options:

Search (user implementation required for ECM documents):  Default ECM Search Service

Get all document versions:  ECM Get All Document Versions

Get document:  ECM Get Document Service

Get type definition:  ECM Get Type Definition Service

Get type descendants:  ECM Get Type Descendants Service

- **Allow create** – Allows a user to upload a new document. The document will be added to the folder defined in the "Parent folder path" attribute.

- Allow update – Allows a user to update the content of a document.
- Open in new window – If selected and if a request to show the content of the document is made, then the content is shown in its own browser window. If we don't check this option and wish to show the content of the document then we will have to use a Document Viewer.
- Number of results to show – The number of results for a document search to show in a page. The default is 10.
- Show all content – If not selected (default) then the result will show a fixed number of rows on the page and paging will be enabled. If selected, then the fixed number of rows will still show but paging will **not** be enabled.
- Configure for use with – Select the type of document provider. There are two choices:
  - ECM Documents – An external CMIS compliant document repository
  - BPM Document – The IBM supplied internal document repository
- BPM document options – When the type of repository is the BPM Document Store, these additional options apply:

**BPM document options:**

<b>Display options</b>						
Associated with process instance: <input checked="" type="checkbox"/>						
Display match rule: <input type="button" value="None"/>						
<b>Display properties</b>						
<table border="1"> <thead> <tr> <th></th> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="button" value="+"/></td> <td></td> <td></td> </tr> </tbody> </table>		Name	Value	<input type="button" value="+"/>		
	Name	Value				
<input type="button" value="+"/>						
<b>Upload options</b>						
Default upload name: <input type="text"/>						
User editable: <input type="checkbox"/>						
Add properties: <input type="checkbox"/>						
<b>Upload properties</b>						
<table border="1"> <thead> <tr> <th></th> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td><input type="button" value="+"/></td> <td></td> <td></td> </tr> </tbody> </table>		Name	Value	<input type="button" value="+"/>		
	Name	Value				
<input type="button" value="+"/>						
Hide in portal: <input type="checkbox"/>						

- Display options – This section acts as a filter to be used when displaying items in the list
  - Associated with process instance – By default, all documents are shown. Selecting this option causes only documents created within the current process instance to be shown.
  - Display match rule – A selection of how matches are made. There are three options:
    - None – Display all documents. No filter is applied.
    - Any – Display documents that match any of the filters.
    - All – Display documents that match all of the filters.
  - Display properties
    - Name – The name of a matching property

- `Value` – The value of a matching property
- Upload options
  - `Default upload name` – Default name for an uploaded document.
  - `User editable` – Defines whether or not the user can edit the upload name.
  - `Add properties` – Should additional properties be added during an upload.
  - `Upload properties`
    - `Name` – The name of a property to be added to a document.
    - `Value` – The value of a property to be added to a document.
  - `Hide in Portal` – If selected, the document will not be visible in the document list.

A BPM data type called `BPMDocumentOptions` is defined that models these settings. This data type contains:

- `displayOptions` (`BPMDocumentDisplayOptions`)
  - `associatedWithProcessInstance` (`Boolean`)
  - `displayMatchRule` (`BPMDocumentMatchRule`)
  - `displayProperties` (`NameValuePair`)
- `uploadOptions` (`BPMDocumentUploadOptions`)
  - `defaultUploadName` (`String`)
  - `userEditable` (`Boolean`)
  - `addProperties` (`Boolean`)
  - `uploadProperties` (`NameValuePair`)
  - `hideInPortal` (`Boolean`)
- ECM document options – ???
- Search (user implementation required for ECM documents) – A service which returns the documents to be shown to the user. See Document List – Search service.
- Get all document versions – A script which gets the versions of each document retrieved
- Get document – A script which retrieves an individual document
- Get type definition – A script which retrieves the types of all the documents retrieved.
- Get type descendants – A script which retrieves the types of documents.

But what does this Coach View look like when it is shown within a Coach? The following is an example screen shot:

**My Docs**

Name	Creation Date	Actions
My Name	2012/11/24 20:21:57.451 CST	
My Test doc	2012/11/24 19:48:27.550 CST	

Page : 1

**Create Document** **Refresh**

What is displayed is a table containing details of the documents that are found within the ECM. If there are more documents than can be display in the maximum allowable size, the table will provide paging (if desired). As mentioned, the selection of columns included in the table can be customized.

In the "Actions" column are click-able icons that can provide functions to be performed against the document that a row represents. The magnifying glass icon allows us to view the document's content in an associated content viewer widget.

The pencil icon allows us to edit the properties of the document. When clicked, we can change the properties or upload a new version of the content. This option is only available if the "Allow update" configuration option is checked.

**Update Document**

Document Properties	
* Name	<input type="text" value="My Name"/>
Document Content	
File Name	<input type="button" value="Choose File"/> No file chosen
<b>OK</b> <b>Cancel</b>	

The final icon in the Actions column allows us to view the previous revisions of the document. The revision list is shown in the same view:

**My Docs**

Name	Creation Date	Actions
My Name	2012/11/24 20:21:57.451 CST	
My Test doc	2012/11/24 19:48:27.550 CST	

Page : 1

**Revisions**

Revision	Name	Revision Date	Created By
1.0	ddd	2012/11/24 19:48:27.550 CST	admin

**Close**

**Create Document** **Refresh**

Also in the view are two further buttons associated with the view as a whole. The first is called

"Refresh" and simply refreshes the view from the content managed by the ECM. If new documents have been added, or old ones deleted the view will reflect those changes after a refresh. The second button is called "Create Document".

The screenshot shows a 'Create Document' dialog box. At the top, there's a title bar with the text 'Create Document'. Below it, there are three main sections: 'Document Type', 'Document Properties', and 'Document Content'. In the 'Document Type' section, 'Document' is selected. Under 'Document Properties', the 'Name' field contains 'My Data'. In the 'Document Content' section, the 'File Name' field shows 'dat.xml' and has a 'Choose File' button next to it. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

Here we can provide a number of parts including the type of document, the name we wish to give to the document and a local file whose content will be uploaded and associated with the new document. The newly created document will be created in the ECM folder defined by the "Parent folder path" configuration attribute. The ability to create new documents is only available if the "Allow creates" configuration option is selected.

The Coach View can be bound to a variable of data type "List of ECMDocumentInfo". This list will be populated with information on the retrieved items. The selected item in the list will change when the user selects a document to view.

See also:

- Description of Document Info (ECMDocumentInfo)
- The Document Viewer Coach View

## Document List – Search service

When the Document List Coach View loads, it has to get the list of documents from the ECM. It performs this task by delegating it to a Service defined by the Search configuration parameter. The Search service is used to return the list of documents to be shown to the end user. The Search service can be easily created by clicking the "New" button in the Document List configuration parameters.

Screenshot of the 'Properties' dialog for a service configuration. The 'Configuration' tab is selected. A red arrow points to the 'New...' button in the 'Get type definition' section, which is highlighted with a blue border.

Action	Service	Type	Buttons
Search (user implementation required)	Default ECM Search Service	Content Management	Select... New... Reset
Get all document versions	ECM Get All Document Versions	Content Management	Select... New... Reset
Get document	ECM Get Document Service	Content Management	Select... New... Reset
Get type definition	ECM Get Type Definition Service	Content Management	Select... New... Reset

A new service with the correct signature will be created for you:

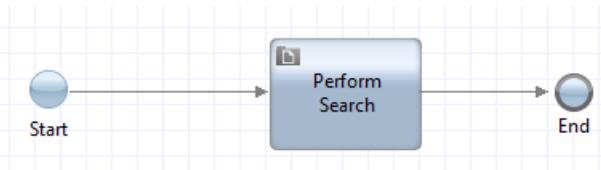
Screenshot of the 'Variables' section of the service configuration. It shows the variable signature:

```

Variables
 Local
 Input
 maxItems (Integer)
 skipCount (Integer)
 searchAllVersions (Boolean)
 cmisQuery (String)
 Output
 searchResult (ECM SearchResult)
 Private
 Exposed Process Variables
 Localization Resources

```

It is your responsibility as the builder of this service to return a `searchResult` object from your execution. To achieve this, add a Content Integration component into the service as shown:

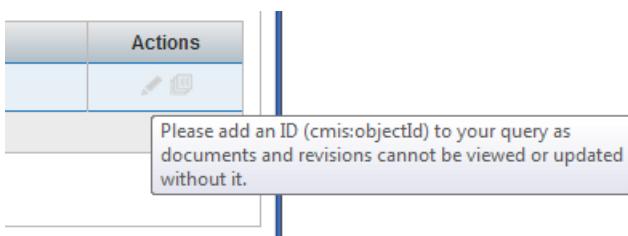


In its implementation settings, select your server and the "Search" operation:

Screenshot of the 'Properties' dialog for the 'Perform Search' step. The 'Implementation' tab is selected. The 'Server' dropdown is set to 'Alf1'. The 'Content Operation' tab is also visible, showing 'Operation name: Search'.

Map its required parameters from the signature of the service you are building:

In order to enable the Actions for the Document List shown rows, the search **must** include the "cmis:objectId" as one of the returned items. This is used as the key to perform the action. This is added by default unless you have chosen to build your own CMIS query. Failure to return this query will result in the actions grayed with a helpful tool-tip hint shown here:



See also:

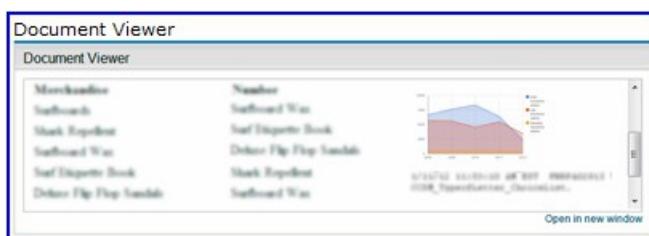
- Content Integration node - Search

## The Document Viewer Coach View

A companion Coach View to the Document List is the Document Viewer. This Coach View's purpose is to act as a container for showing the content of a selected document. From the palette, this Coach View looks like:



When dropped on a Coach canvas, it looks like:



The Document Viewer Coach View has no configuration options. However, it should be bound to a variable of type "ECMDocumentInfo" that should be a variable also bound to a Document List Coach View. The recommended way is to bind to the "listSelected" element in the Document List Coach View bound list. When the user selects a document in the Document List Coach View, the selected entry in the list changes which is noticed by the Document Viewer Coach View and is the trigger to show the new content.

The height of the Document Viewer is not likely to be the size you want. This can be fixed by changing the style:

```
.classDocViewFrame {
```

```
 height: 500px;
}
```

See also:

- The Document List Coach View
- Description of Document Info (ECMDocumentInfo)

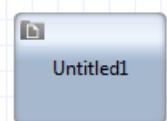
## ***Content Integration node***

We should now have a general idea that interaction with an ECM is achieved using a protocol called CMIS and that there are Coach Views that can be used to present the documents to the user. We have also seen that some of these Coach Views call Services in order for them to obtain data. What has not yet been explained is how these Service arrange for the CMIS calls to the ECM provider to be made. IBM BPM provides a node that can be included in a service called the Content Integration node. This is the core to the story.

The Content Integration node provides CMIS access to the ECM provider. It can be added to a variety of Service implementations. When seen in the palette, it looks as follows:



When added to the service implementation canvas, it looks like:



Once added to the canvas, the implementation details of the node must be defined. There are two primary attributes. The first is the server against which the CMIS command is to be performed. This server definition must be create in the Process Application Server options. The second parameter is called "Operation Name" and defines which CMIS operation is to be performed against the server.



Part of the goal of IBM BPM is to try and shield the developer from the worst details of CMIS usage.

The operations that can be performed on a Content Integration node are:

- Add document to folder
- Cancel check-out document
- Check-in document
- Check-out document
- Copy document
- Create document
- Create folder
- Delete document
- Delete folder
- Get all document versions
- Get documents in folder
- Get document

- Get document content
- **Get folder**
- **Get folder by path**
- **Get folder tree**
- Get type definition
- Get type descendants
- Move document
- Move folder
- Remove document from folder
- Search
- Set document content
- Update document properties

The entries in red are not available to the IBM BPM Document Store.

Each of these commands will be covered in turn.

Depending on which command is selected, the input and output parameters of the Content Integration node will change dramatically.

IBM provides some new Business Object data type definitions for some of those parameters. These new data types are described next

### **Description of a Document (ECMDocument)**

A document is described by the data type called "ECMDocument":

- `objectId` – The unique ID of the document
- `serverName` – The name of the server eg "Alfresco"
- `repositoryId` – The ID of the server hosting this document.
- `objectTypeId` – The type of the document eg. "cmis:document"
- `name` – The name of the document
- `contentURL` – A URL that can be used to extract the content of the document
- `creationDate`
- `createdBy`
- `lastModificationDate`
- `lastModifiedBy`
- `versionLabel`
- `isLatestVersion`
- `isMajorVersion`
- `isLatestMajorVersion`
- `checkinComment`
- `properties`

### **Description of a Search Result (ECM SearchResult)**

When a search is executed, an object of type ECM SearchResult is returned. This object contains:

- `numItems` – The number of records returned

- hasMoreItems – Are there more items to be returned?
- serverName – The name of the server against which the search was executed. For IBM BPM Document Store this is "EMBEDDED\_ECM\_SERVER".
- repositoryId – An identifier for the repository.
- objectTypeIds – The kind of data returned (eg. IBM\_BPM\_Document)
- propertyMetadata – An array of meta data describing the data returned
- resultSet – An array of records, one for each document. The property "row" is an array of records of data type ECMSearchResultRow. This contains a single field called "column" which is a list of ANY. The entries in the data correspond to each of the propertyMetadata items.

## Description of an ECMID

### Description of Document Info (ECMDocumentInfo)

The ECMDocumentInfo contains a single property called "contentURL" which is the Web URL used to retrieve the content of the document. This data type can be seen as a list of variables bound to a Document List Coach View.

Note: It is odd to the author that the ECMDocumentInfo contains only the contentURL. It would seem to have been useful for this data structure to also contain the objectId of the selected documents.

See also:

- The Document List Coach View

### Description of a folder/document property (ECMProperty)

- objectTypeIds (ECMID)
- value (ANY)

### Description of a folder (ECMFolder)

The details of a folder can be retrieved with "Get folder" or "Get folder by path". This data structure represents the details returned.

- objectId
- serverName
- repositoryId
- objectTypeIds
- path
- parentId
- name
- creationDate

- createdBy
- lastModificationDate
- lastModifiedBy
- properties
- subFolders

See also:

- Content Integration node - Get folder
- Content Integration node - Get folder by path

### Description of a Content Stream (ECMContentStream)

When one uses the Get Document Content call, we get back an ECMContentStream. This looks as follows (8.5):

- contentLength (Integer)
- mimeType (String)
- fileName (String)
- content (ANY)

### Description of an object type definition (ECMOObjectTypeDefinition)

- objectType (ECMID)
- localName (String)
- localNamespace (String)
- queryName (String)
- displayName (String)
- description (String)
- serverName (String)
- repositoryId (ECMID)
- baseId (ECMID)
- parentId (ECMID)
- queryable (Boolean)
- propertyTypeDefinitions (List of ECMPropertyTypeDefinition)
- children (List of ECMObjectTypeDefinition)

For IBM BPM, the property type definitions are:

Name	Type	Updateable	Queryable
IsReserved	boolean	No	Yes
IsVersioningEnabled	boolean	No	Yes

MajorVersionNumber	Integer	No	Yes
MinorVersionNumber	Integer	No	Yes
VersionStatus	Integer	No	Yes
ReservationType	Integer	No	Yes
DateCheckedIn	Date	No	Yes
CmIsMarkerForDeletion	Boolean	No	No
DateContentLastAccessed	Date	No	Yes
ContentRetentionDate	Date	No	Yes
CurrentState	String	No	Yes
IsInExceptionState	Boolean	No	Yes
ClassificationStatus	Integer	No	Yes
IndexationId	id	No	Yes
CmlIndexingFailureCode	Integer	No	Yes
DocumentTitle	String	Yes	Yes
IBM_BPM_Document_DocumentId	Integer	When Checked Out	Yes
IBM_BPM_Document_ParentDocumentId	Integer	On Create	Yes
IBM_BPM_Document_UserId	Integer	When Checked Out	Yes
IBM_BPM_Document_FileType	String	When Checked Out	Yes
IBM_BPM_Document_HideInPortal	Boolean	On Create	Yes
IBM_BPM_Document_ProcessInstanceId	Integer	On Create	Yes
IBM_BPM_Document_Properties	String[]	When Checked Out	Yes
IBM_BPM_Document_FileNameURL	String	When Checked Out	Yes
IBM_BPM_Document_ContentMigrated	Boolean	When Checked Out	Yes
cmis:name			
cmis:objectId			
cmis:baseTypeId			
cmis:objectTypeId			
cmis:createdBy			
cmis:creationDate			
cmis:lastModifiedBy			
cmis:lastModificationDate			
cmis:changeToken			
cmis:isImmutable			
cmis:isLatestVersion			
cmis:isMajorVersion			
cmis:isLatestMajorVersion			
cmis:versionLabel			
cmis:versionSeriesId			
cmis:isVersionSeriesCheckedOut			
cmis:versionSeriesCheckedOutBy			
cmis:versionSeriesCheckedOutId			
cmis:checkinComment			
cmis:contentStreamLength			
cmis:contentStreamMimeType			
cmis:contentStreamFileName			
cmis:contentStreamId			

## **Content Integration node - Add document to folder**

This function adds a Document already uploaded to an existing folder. The input parameters are the DocumentId of the document to be added to the folder plus the FolderId into which the document will be added.

## **Content Integration node - Cancel check-out document**

Cancel a check-out of a previously checked-out document.

See also:

- Content Integration node - Check-in document
- Content Integration node - Check-out document

## **Content Integration node - Check-in document**

Check in a checked-out document.

See also:

- Content Integration node - Cancel check-out document
- Content Integration node - Check-out document

## **Content Integration node - Check-out document**

Check out a document. The input parameter is the DocumentId of the document to be checked-out.

When we check out a document, we are given a temporary DocumentId that we can then use to work with it. It is possible to "lose" that DocumentId before we check the document back in. For example as the result of a coding error or system crash. In this case, the original document will remain locked and we will be prevented further access to it. To resolve that issue, we need a mechanism that will allow us to retrieve the temporary document id so that we can unlock it again.

See also:

- Content Integration node - Cancel check-out document
- Content Integration node - Check-in document

## **Content Integration node - Copy document**

Make a copy of a document.

## **Content Integration node - Create document**

Create a new instance of a document. The input data for the document can also be supplied.

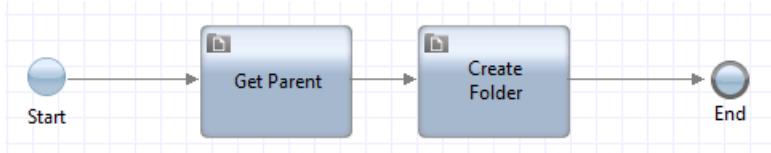
## **Content Integration node - Create folder**

Create a new folder as a child of an existing folder. The parameters to this command are

- Object Type ID – The type of the new folder to be created (eg. "cmis:folder")
- Parent Folder ID – The ID of the parent folder which this new folder will be a child
- Name – The name of the new folder to be created
- Properties (Optional) – Properties of the new folder. An list of ECMProperty objects

It returns a "Folder ID" that uniquely represents the new folder.

The Parent Folder ID can be retrieved by a prior call using the "Get folder by path" operation. This will return an ECMFolder object. Contained within it is a property called "objectId" which is the ID of the parent folder.



An error will be thrown if the folder already exists.

See also:

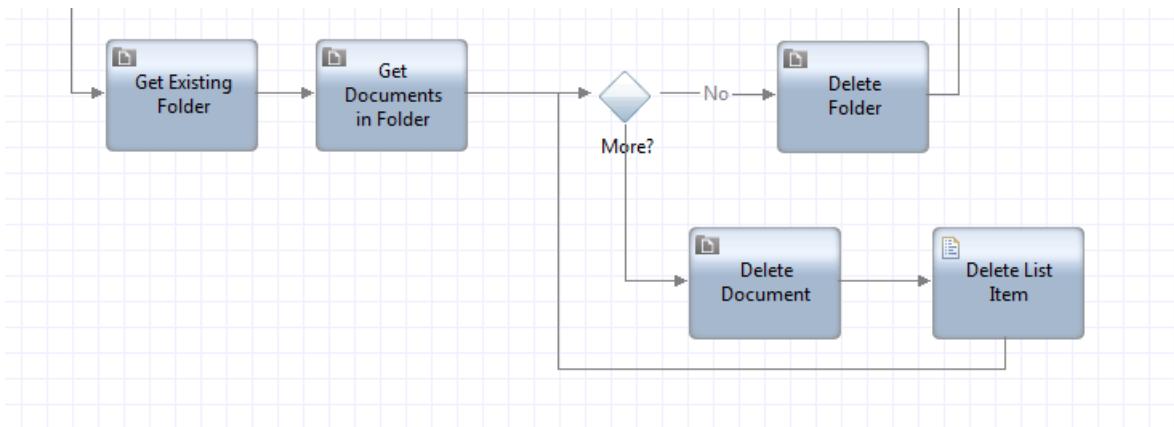
- Content Integration node - Get folder by path
- Description of a folder (ECMFolder)

## Content Integration node - Delete document

Delete a document. The document to be deleted is supplied by a "Document ID". A Boolean called "All Versions" can be used to delete all versions or just some.

## Content Integration node - Delete folder

Delete a folder. This command will delete a folder. The folder is identified with its own folder ID. If the folder contains content, it will not be allowed to be deleted. One solution to deleting folders which contain content is to first delete the content and then delete the folder. This can be achieved with the already existing primitives supplied as part of the Content Integration node suite of operations. The algorithm is to get a list of all documents in the folder and then, looping over each item, delete it. Once all the items have been deleted, we can then delete the folder.



CMIS provides a useful function called "deleteTree" which deletes the folder and all descendant objects. Unfortunately, that operation is not mapped to the Content Integration node and can't be used.

## Content Integration node - Get all document versions

Retrieve all the versions of a document.

## Content Integration node - Get documents in folder

This method takes a folder ID and returns a list of ECMDocument objects where each document

corresponds to a document found in the folder defined by the ID.

### **Content Integration node - Get document**

Retrieve a specific document.

### **Content Integration node - Get document content**

Retrieve the content of a specific document.

### **Content Integration node - Get folder**

Retrieve the details of a specific folder.

See also:

- Description of a folder (ECMFolder)

### **Content Integration node - Get folder by path**

This method takes the path to a folder as a String and returns an ECMFolder object which is populated with the details of that folder. Importantly, this includes the ObjectID that uniquely identifies the folder.

See also:

- Description of a folder (ECMFolder)

### **Content Integration node - Get folder tree**

### **Content Integration node - Get type definition**

### **Content Integration node - Get type descendants**

### **Content Integration node - Move document**

### **Content Integration node - Move folder**

### **Content Integration node - Remove document from folder**

### **Content Integration node - Search**

The search mechanism is based on the CMIS Query language. Process Designer provides guidance on creating such a search. If a service contains a Content Integration node, a new tab can be found at the top of the definition. This tab is called "Content Filters" and is used to build a CMIS query against the ECM. The query to be built has a number of properties and provides a number of ways in which it can be constructed. Here is a screen shot of a basic setup.

The screenshot shows the 'Content Filters' tab selected in the top navigation bar. On the left, there's a sidebar with icons for 'Perform Search' and 'Untitled2'. The main area is titled 'Build Search Filter' and contains the following sections:

- Method of creating search:** A dropdown menu set to 'Content Filters'.
- Object Type:** A section where 'Document' is selected from a dropdown. There's also a 'Select...' button.
- Properties:** A table listing four items with columns: Name, Creation Date, and Object Id. The data is as follows:

	Name	Creation Date	Object Id
1	35b8be80-170f-40af-a173-513758b83165	Tue Feb 15 16:30:39 CST 2011	workspace://SpacesStore/8d4429e7-804f-43cf-bd81-288e56
2	42fcbae6-b1fe-4028-9f85-9ad7f81a8e3b	Tue Feb 15 16:18:03 CST 2011	workspace://SpacesStore/db31dce5-2469-4c68-8641-9beca
3	50046cc0-9034-420f-925b-0530836488c4	Tue Feb 15 16:23:00 CST 2011	workspace://SpacesStore/e57195d3-aeda-432d-bfc4-0a556
4	567ee439-4ebc-40cf-a783-3e561ad5a605	Tue Feb 15 16:31:26 CST 2011	workspace://SpacesStore/7778cf88-836f-4833-a0df-3056d2

- Search Criteria:** A section with a checkbox for 'Result set sort order specified by process variable' and a 'Match Criteria' dropdown set to 'All'. There's also a 'Add Search Criterion...' button.

The first thing we will discuss is the entry titled "Method of creating search". This has two possible values:

- **Content Filters** – The search will be built by the developer visually from settings contained within this page
- **Data Mapping** – The search will be built by hand using explicit CMIS query language and defined in the data mapping parameters of the Content Integration node

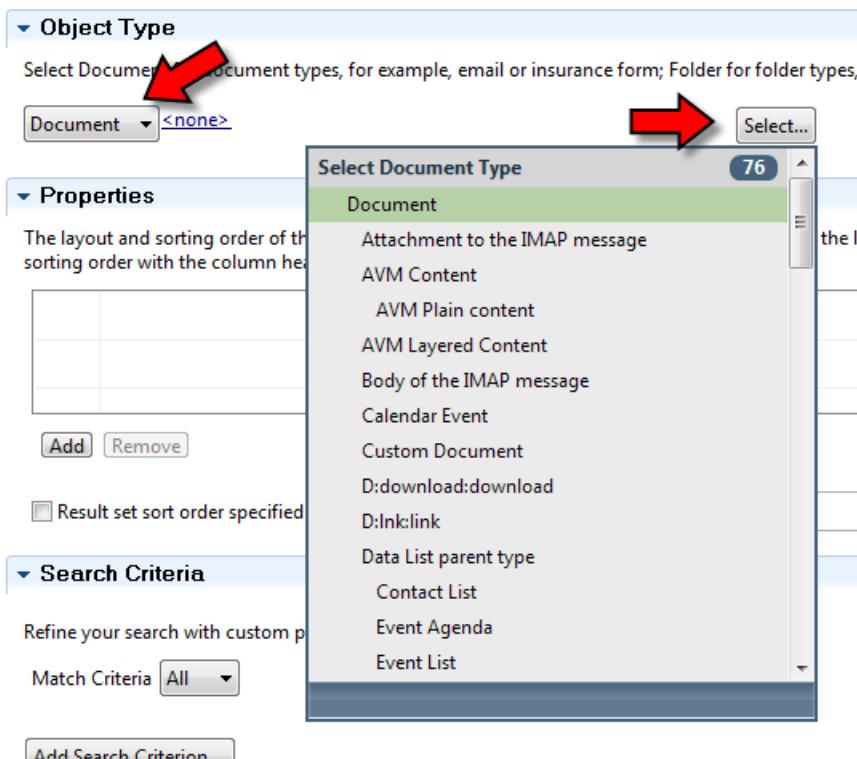
By using Data Mapping, you have full low-level control over the query executed but in order to use that, you will have to understand and test CMIS queries. The Content Filters setting allows you to build the query "visually". Let us concentrate on how to build queries visually.

The Object Type area allows us to define which types of objects we wish to search upon. At the highest level, there are two types of object we can list. These are:

- **Document** – Return a list of documents
- **Folder** – Return a list of folders

We can think of both Documents and Folders as base "types" of more specific types of document. For example, consider a business which has multiple types of documents. Maybe it has "receipts", "orders", "tax returns" and more. Each of these is indeed a "Document" but the ECM system can define specializations of such documents against which constraints and other rules may be applied. The types of documents (all based on the base "Document" type) is defined at the ECM server but can be dynamically retrieved by IBM BPM.

Once the choice is made in the visual wizard builder as to whether we want to return Documents or Folders, we then have the option to further refine the type of Objects to return by selecting a specialization using the Select button. In the following screen shot we have selected that we want Documents (one of the two allowable types) and from there we clicked Select. A query was then made to the ECM provider to retrieve a list of the other types of Documents available. From here we can select the root Document type or a more specialized variant. When the query is finally made, only documents of that specific type (or which inherit from that type) are returned:



Once we have defined the types of Object to be returned by the search, the next aspect is to define which properties of the Object we wish to return and/or display. In the middle of the editor a table is shown. This table should be used as a mental aid to consider the data returned. Each column in the table is an indication of the property of data from an object that will be shown. The Add and Remove buttons can be used to add additional columns or remove existing columns.

Finally, we can define filtering information in the "Search Criteria" area. Here we specify expressions which are applied to the data and only those expressions which evaluate to true are returned to the user.

The Content Integration searching can be filtered by a variety of parameters including:

- Base Type Id
- Content Stream File Name – The underlying file name of the document
- Content Stream Length – The length in bytes of the data of the document
- Content Stream MIME Type – The MIME type of the document
- Created by
- Creation Date – The date the document was added to the ECM
- Last Modified By

- Last Modified Date
- Name – The name of the document as known to the ECM
- Object Id
- Object Type Id – The type of object eg, "cmis:document"

The return from a Content Integration search is an object of type `ECMSearchResult` which contains:

- `numItems`
- `hasMoreItems`
- `serverName`
- `repositoryId`
- `objectTypeId`
- `propertyMetadata`
- `resultSet`

The result of building this query visually will be Data Mappings that are bound for us:

The screenshot shows the Data Mapping configuration interface. On the left, there are tabs for Properties, Validation Errors, and Where Used. The main area has tabs for Step, Implementation, Data Mapping (which is selected), and Pre & Post. Under Data Mapping, there are two sections: Input Mapping and Output Mapping.

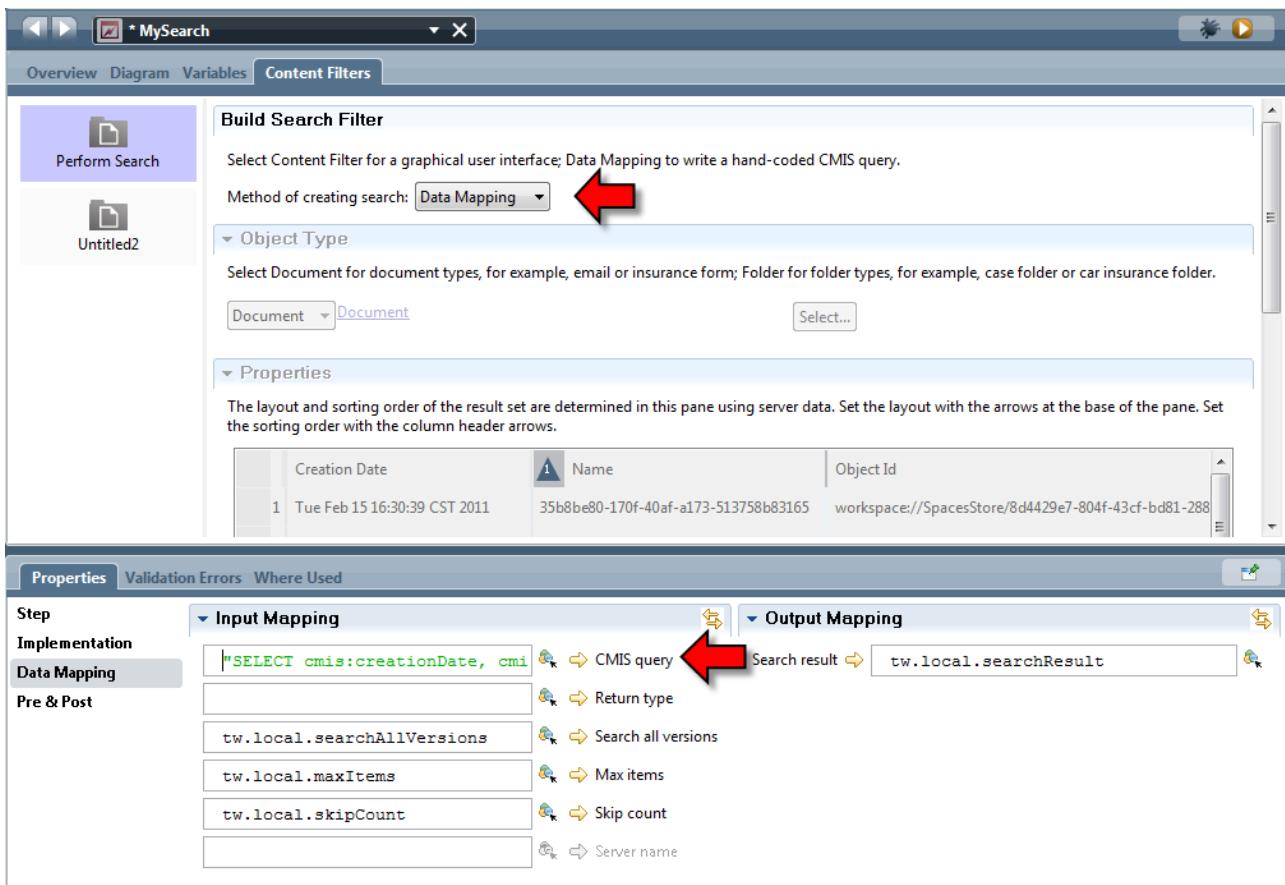
**Input Mapping:**

- "SELECT cmis:creationDate, cmi..." → CMIS query
- tw.local.searchAllVersions → Return type
- tw.local.maxItems → Search all versions
- tw.local.skipCount → Max items
- tw.local.skipCount → Skip count
- tw.local.skipCount → Server name

**Output Mapping:**

- Search result → tw.local.searchResult

By changing the method of search from "Content Filters" to "Data Mapping", we now have full control over the searching:



The main part of the Content Filters editor is now grayed to indicate that it is now editable while the CMIS query data mapping is now modifiable. Here we can supply a valid CMIS query, the result of which will be used as the query against the ECM provider.

The properties available for Documents are:

Property	Queryable
cmis:name	●
cmis:objectId	●
cmis:baseTypeId	●
cmis:objectTypeId	●
cmis:createdBy	●
cmis:creationDate	●
cmis:lastModifiedBy	●
cmis:lastModificationDate	●
cmis:changeToken	
cmis:isImmutable	
cmis:isLatestVersion	
cmis:isMajorVersion	
cmis:isLatestMajorVersion	
cmis:versionLabel	
cmis:versionSeriesId	
cmis:isVersionSeriesCheckedOut	

cmis:versionSeriesCheckedOutBy	
cmis:versionsSeriesCheckedOutId	
cmis:checkinComment	
cmis:contentStreamLength	•
cmis:contentStreamMimeType	•
cmis:contentStreamFileName	•
cmis:contentStreamId	

The properties available for Folders are:

Property	Queryable
cmis:name	•
cmis:objectId	•
cmis:baseTypeId	
cmis:objectTypeId	•
cmis:createdBy	•
cmis:creationDate	•
cmis:lastModifiedBy	•
cmis:lastModificationDate	•
cmis:changeToken	
cmis:parentId	•
cmis:path	
cmis:allowedChildObjectIds	

See also:

- Document List – Search service
- Description of a Search Result (ECM SearchResult)
- [CMIS Query Language \(Alfresco Wiki\)](#)

## Content Integration node - Set document content

Set the content of an existing document.

## Content Integration node - Update document properties

This operation is used to update the properties of an existing document. The parameters to it are:

- Document ID (ECMID) – The id of the document to be updated
- Name (String) – The optional new name of the document
- Properties (List of ECMProperty) – An optional list of the new properties.

See also:

- Description of a folder/document property (ECMProperty)

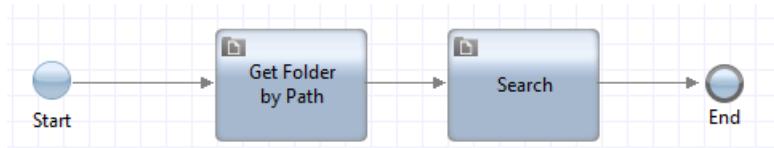
## Examples of Content Integration nodes

Here are some examples of using Content Integration nodes

### Finding documents in a specific folder

Imagine we wish to find documents located in a specific folder. We know the path name to the folder.

We create a service that looks as follows:



The first Content Integration node performs a "Get folder by path" command using a path variable as input. This returns an ECMFolder object. Contained within that object is a property called `objectId` which is the ID of the folder corresponding to the path. In the Search Content Integration node, we build a specific CMIS query that looks as follows:

```
"SELECT cmis:name, cmis:creationDate, cmis:objectId FROM cmis:document WHERE IN_FOLDER('" + tw.local.folder.objectId + "') ORDER BY cmis:name ASC"
```

This uses the CMIS function called "`IN_FOLDER`" which constrains the search to be documents within a folder given by its ID. We now have that ID from the previous step and use it.

### Inbound ECM events

In addition to being able to cause work to be performed in an external ECM system, IBPM can also be informed when external work happens. This means that changes in the ECM can result in events being sent to IBPM that inform it of interesting updates. An indication of a change in an ECM is termed a "content event". It is the responsibility of the external ECM to generate and publish such content events. This means that logic/code needs to be built and injected into the ECM. An instance of such code is called an ECM event handler.

The content event story has some key concepts associated with it. The first is the "Event source ID".

The following lists the events supported by BPM and what types of ECM object they apply to:

Event Type	Allowable Object
CheckedIn	Document
CheckedOut	Document
CheckOutCanceled	Document
ClassChanged	Folder or Document
ClassifyCompleted	Document
Created	Folder or Document
Deleted	Folder or Document
Filed	Folder
Frozen	Document
Locked	Folder or Document
PublishCompleted	Document

PublishRequested	Document
SecurityUpdated	Folder or Document
StateChanged	Document
Unfiled	Folder
Unlocked	Folder or Document
Updated	Folder or Document
VersionDemoted	Document
VersionPromoted	Document

IBM supplies an event handler for the IBM FileNet product. This is supplied in source form in a file found at:

C:\IBM\WebSphere\AppServer\BPM\EventHandlers\ECM\FileNet\filenet-bpm-event-handler-51-src.jar

## ECM event REST requests

An Event Handler interacts with the BPM run-time by sending REST requests to the server.

The format of the request is:

```
POST /rest/bpm/wle/v1/event?eventSourceId=?&objectTypeId=?&eventType=?&objectId=?
```

The possibilities for eventType are:

CheckOutCanceled
CheckedIn
CheckedOut
ClassChanged
ClassifyComplete
Created
Deleted
Filed
Frozen
Locked
PublishCompleted
PublishRequested
SecurityUpdated
StateChanged
Unfiled
Unlocked
Updated
VersionDemoted
VersionPromoted

## *Debugging ECM calls*

Since IBM BPM is making CMIS calls using the Web Services protocol, we can intercept those calls and view the requests and the responses. To achieve this, Apache TCPMon can be used. In the ECM Server definitions, specify the port number for TCPMon and have TCPMon forward those requests to the final destination for the ECM provider.

**Note:** 2012-11-25 – It seems that the introduction of TCPMon is causing more problems than it solves. TCPMon has been seen to truncate data and only pass a subset onwards... ouch!!

As an alternative to TCPMon, the IID TCP/IP Monitor view may be used. It seems unable to parse the content as XML but a copy/paste of text into an XML editor seems to do the trick.

## CMIS Tools

When learning or debugging interactions with ECMs through CMIS we can use a rich selection of existing CMIS based tools. This section provides some notes on using these.

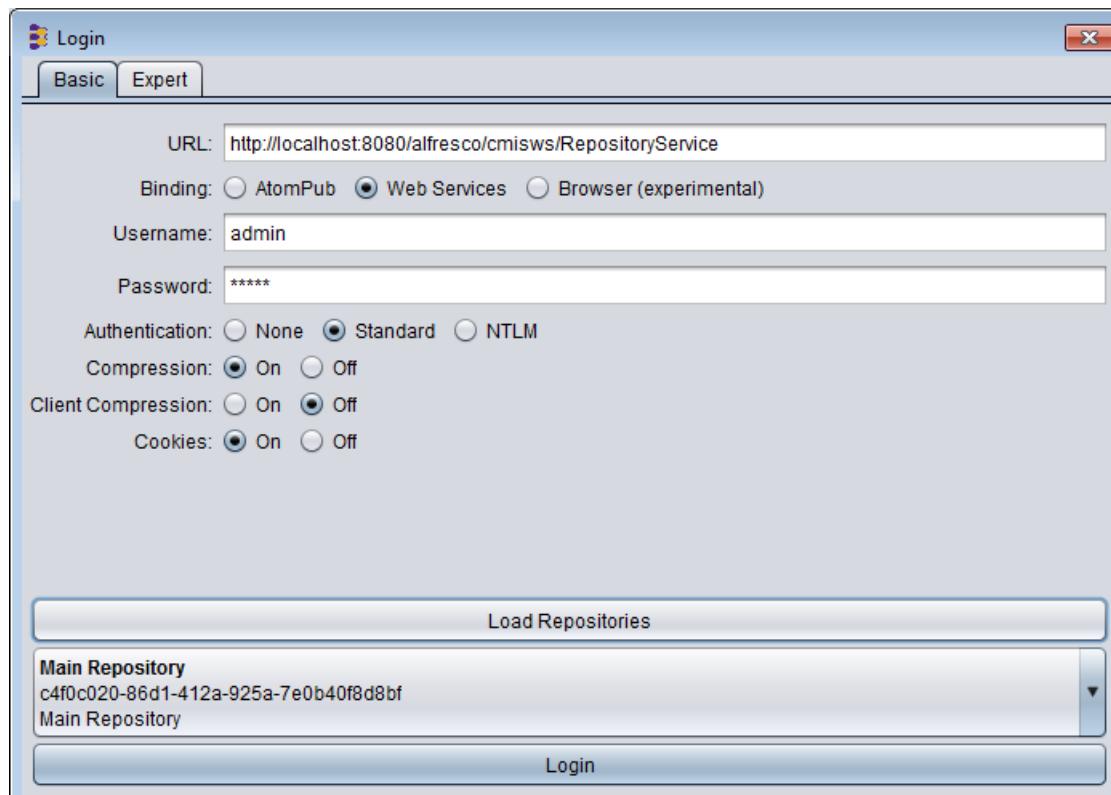
### CMIS Workbench

The CMIS Workbench tool can be used to interact with an ECM Server using CMIS directly using the CMIS protocol. CMIS Workbench is part of the Apache Chemistry project and can be found here:

<http://chemistry.apache.org/java/developing/tools/dev-tools-workbench.html>

If you are going to be working extensively with IBM BPM and ECM, it is suggested that you become familiar with this tool as it will reveal a wealth of information.

To access Alfresco, use:



The URLs are:

- Alfresco

- Web Services - <http://localhost:8080/alfresco/cmisws/RepositoryService>
- Atom - <http://localhost:8080/alfresco/cmisatom>

## ***API Programming for ECM***

The primary way for interacting with the ECM is via CMIS. However, there are other opportunities to work with the ECM outside of this area.

### ***Uploading a new document***

The only legal way to upload a new document is through the Document List Coach View. However, there appears to be a way to programatically achieve this but it is neither documented nor supported. It has been found by reverse engineering how the Document List achieves the task.

The core appears to be an HTTP request to:

```
/portal/jsp/ecmDocument?operation=ajax_createDocument
```

This needs important HTTP header data including:

- Content-Type: multipart/form-data

The properties sent by the browser seem to include:

- objectTypeId – (Required) IBM\_BPM\_Document
- cmis:name – The CMIS name of the document in the CMIS store. This is shown in the "Name" column of the document list.
- fileContent; filename
- snapshotId – (Required) eg. "2064.xxx"
- ecmServerConfigurationName – EMBEDDED\_ECM\_SERVER
- folderPath – eg. "/"
- versioningState – eg. "major"
- snapshotId – 2064.xxx
- IBM\_BPM\_Document\_Properties
- IBM\_BPM\_Document\_ProcessInstanceId – eg. -1
- IBM\_BPM\_Document\_HideInPortal – eg. "false"
- IBM\_BPM\_Document\_UserId – eg. 1002
- IBM\_BPM\_Document\_FileNameURL – Name of the file as shown in the "File name or URL" column in the document list.
- IBM\_BPM\_Document\_FileType – FILE

## ***ECM Providers***

## IBM BPM Document Store

Starting with version 8.5 of IBM BPM, a component called the "BPM Document Store" was supplied with the IBPM product. This is an implementation of a CMIS aware document repository that simply "exists" within the BPM product. For some purposes, this will provide all that is needed to work with documents without having to have an external document management system available to you.

It is important to note that the BPM Document Store does **not** support folder based operations. To leverage those functions, a full function CMIS compliant ECM provider should be used.

A document managed by the BPM Document Store has a data structure associated with it that is called "IBM\_BPM\_Document". Its properties are shown in the following table:

Property	Type	Description
IBM_BPM_Document_DocumentId	Integer	A unique Id for this document instance
IBM_BPM_Document_ParentDocumentId	Integer	A unique Id for the parent of this document instance
IBM_BPM_Document_FileType	String	What kind of document is this? Choices are: <ul style="list-style-type: none"><li>• FILE</li><li>• URL</li></ul>
IBM_BPM_Document_FileNameURL	String	The original file name of the document
IBM_BPM_Document_HideInPortal	Boolean	??
IBM_BPM_Document_ProcessInstanceId	Integer	??
IBM_BPM_Document_Properties	String[]	Additional properties of the document. In a search result, the data will contain an array of properties. Each item will be of the format " <code>&lt;name&gt;, &lt;value&gt;</code> ". For example, if a SearchResult is returned, the the following can be used to access a list of Strings for each property:  <code>searchResult.resultSet.row[i].column[j].value[k]</code>  Experience also seems to show that if there is only ONE property, instead of an array of Strings being returned, only a single string is returned. We need to accommodate this in the logic.
IBM_BPM_Document_UserId	Integer	Creator or last modifier of the document

When building a CMIS query by hand, it seems that the query must execute against "IBM\_BPM\_Document" as opposed to "cmis:document" which would be the normal CMIS target. For example, the following is a valid CMIS query against the IBM BPM Document Store:

```
SELECT cmis:name, cmis:objectId, cmis:versionSeriesId, IBM_BPM_Document_FileNameURL,
IBM_BPM_Document_FileType, IBM_BPM_Document_Properties FROM IBM_BPM_Document ORDER BY cmis:name ASC
```

## FileNet

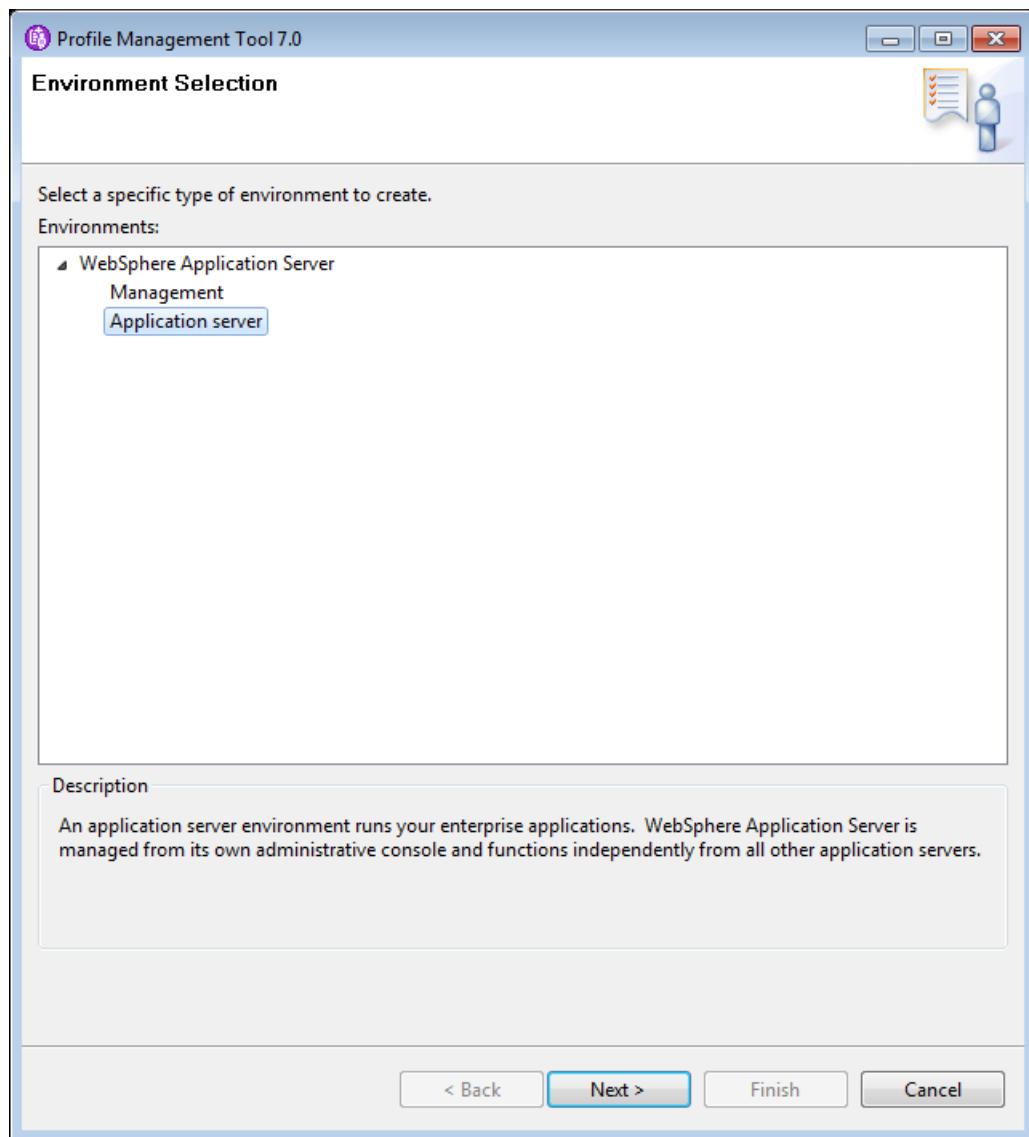
### FileNet Information Sources

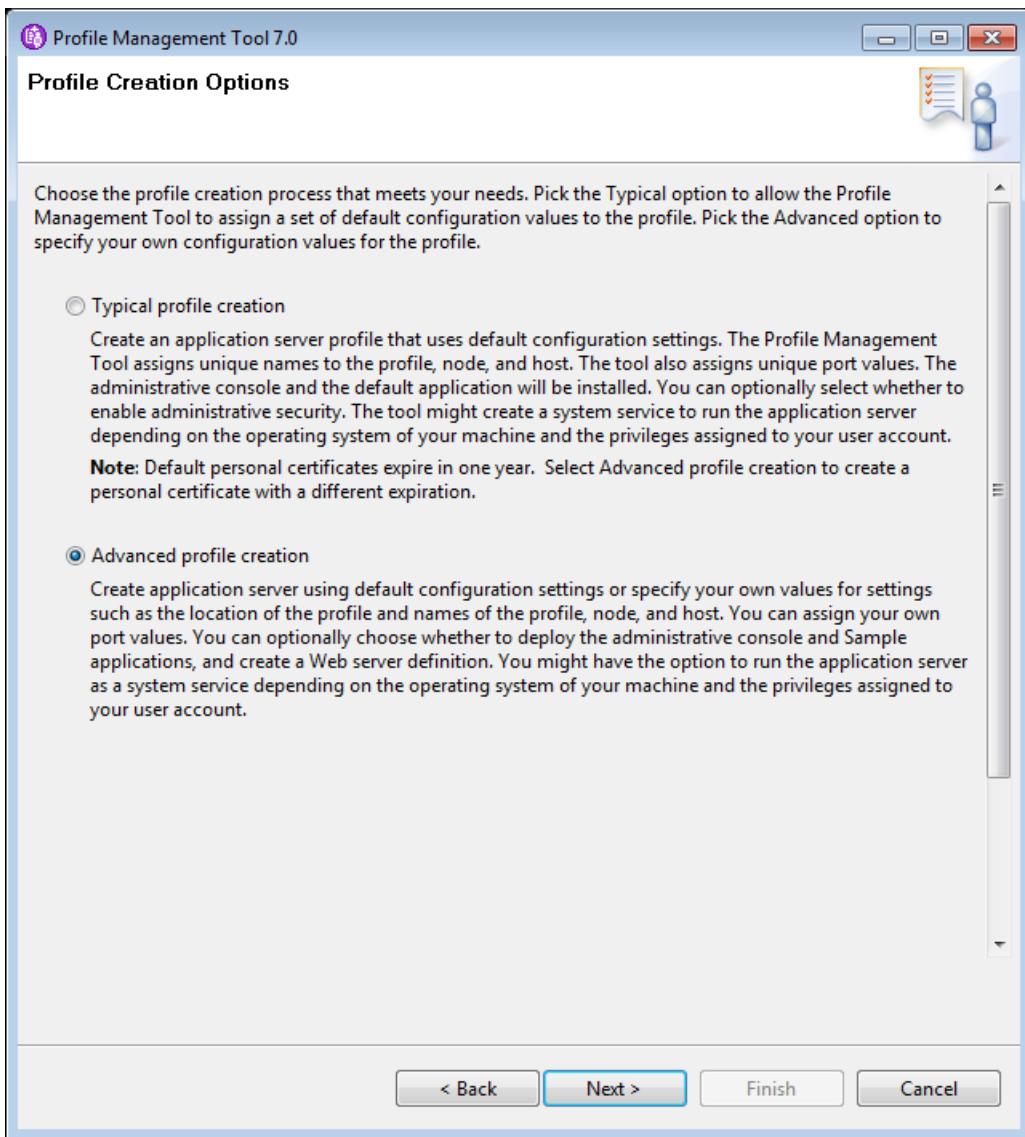
- [InfoCenter - FileNet v5.1](#)

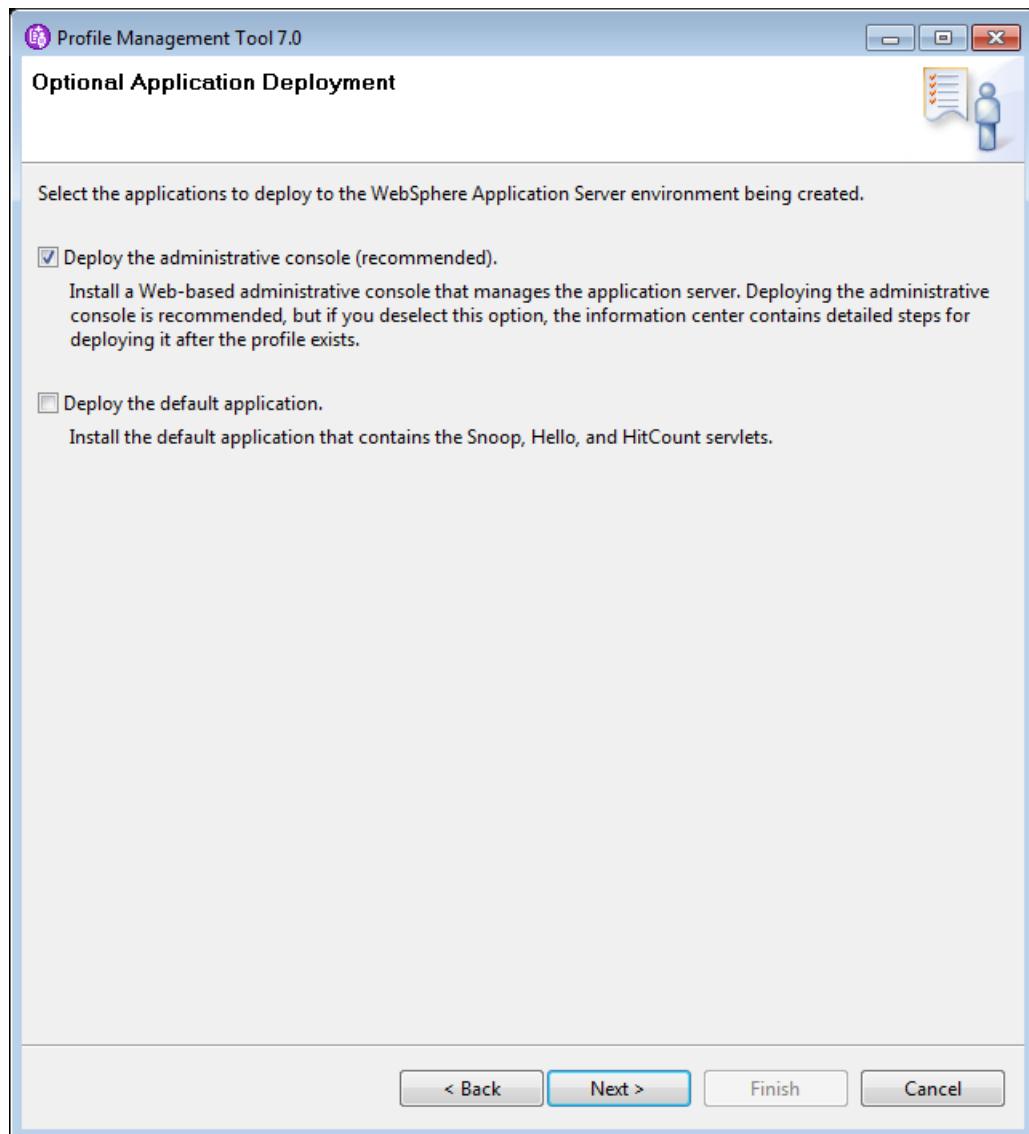
## **FileNet Installation**

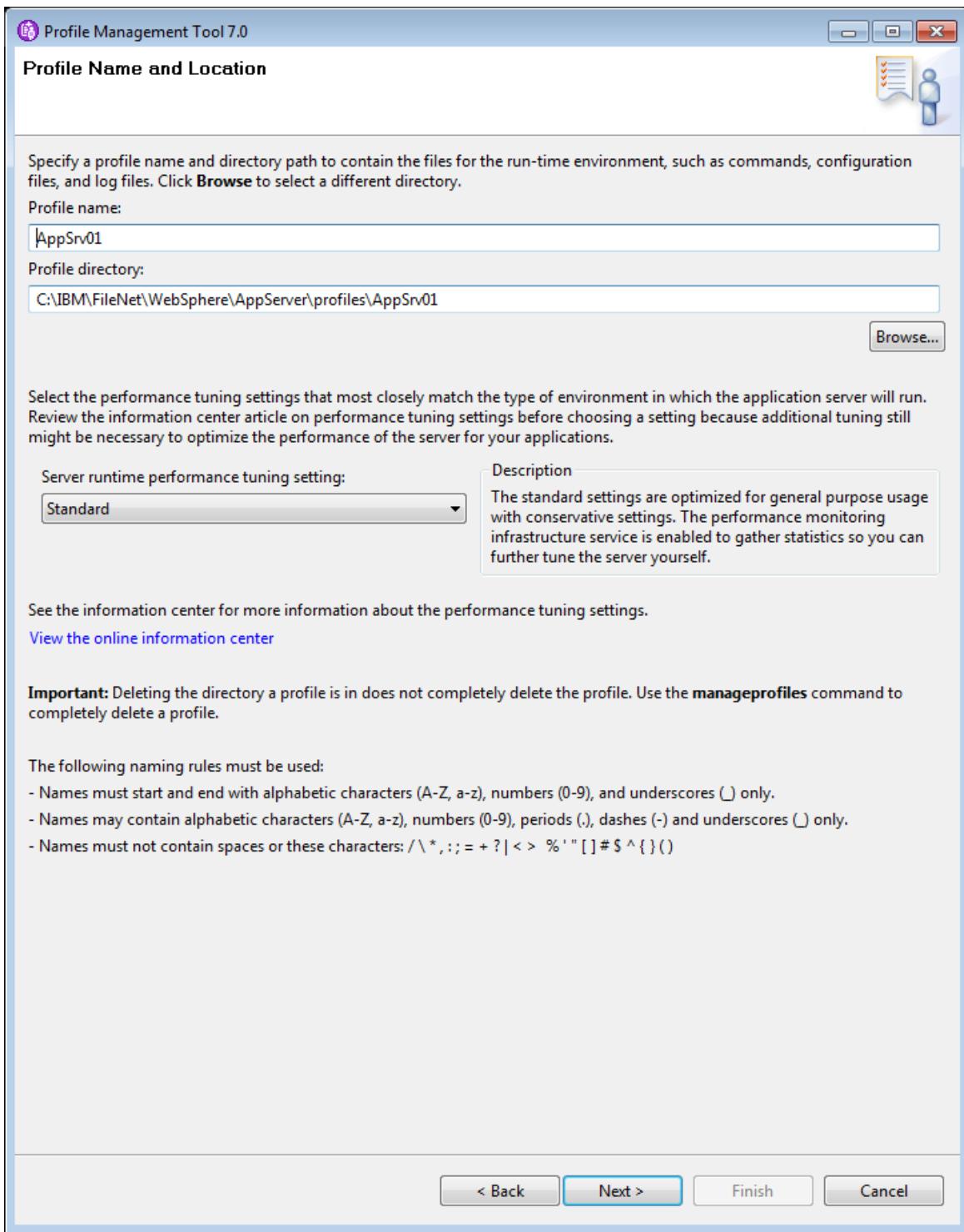
Microsoft WSE 3.0 was required to be installed on my box.

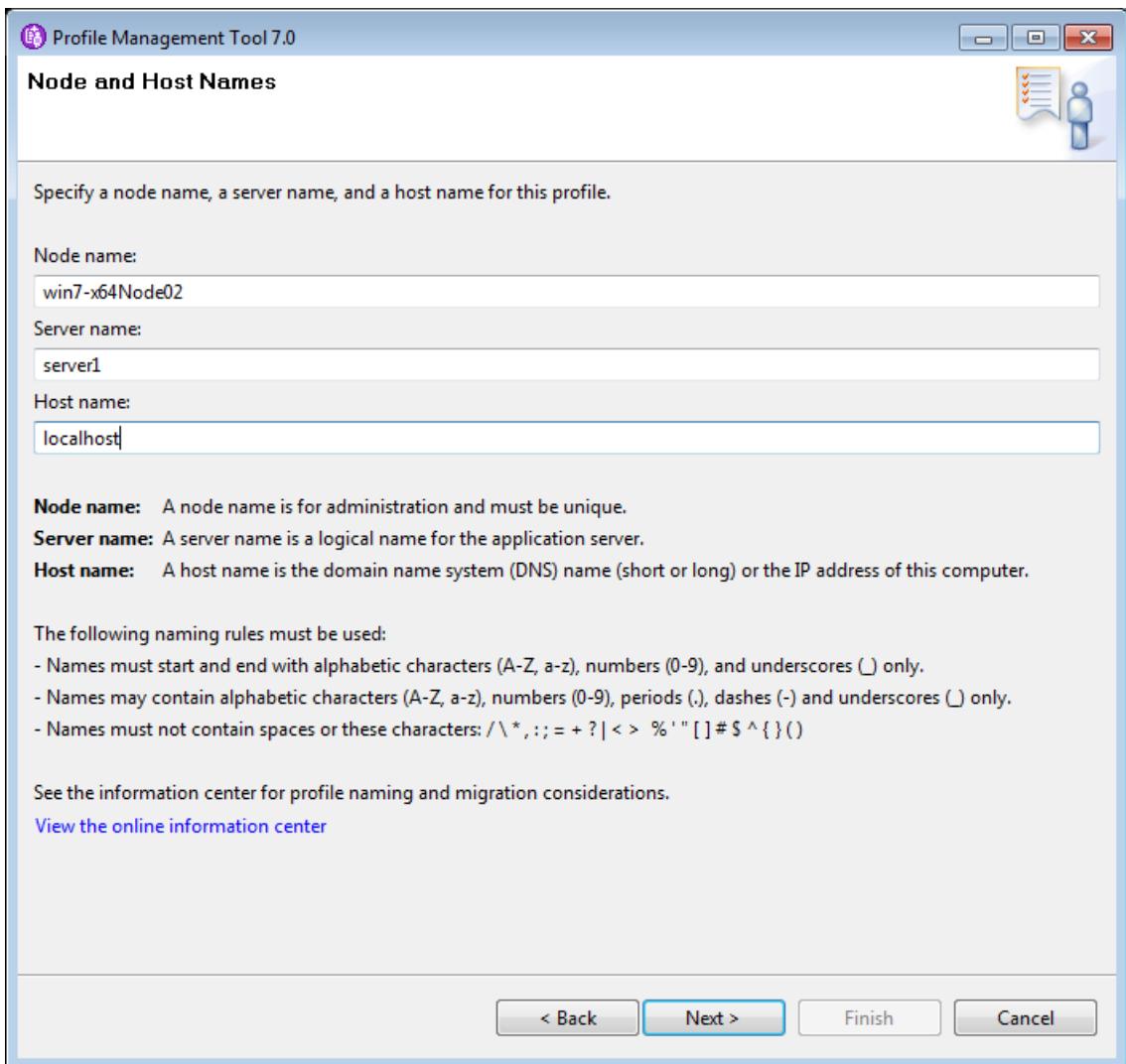
We start by using the WAS Profile Management Tool to create a basic WAS profile:

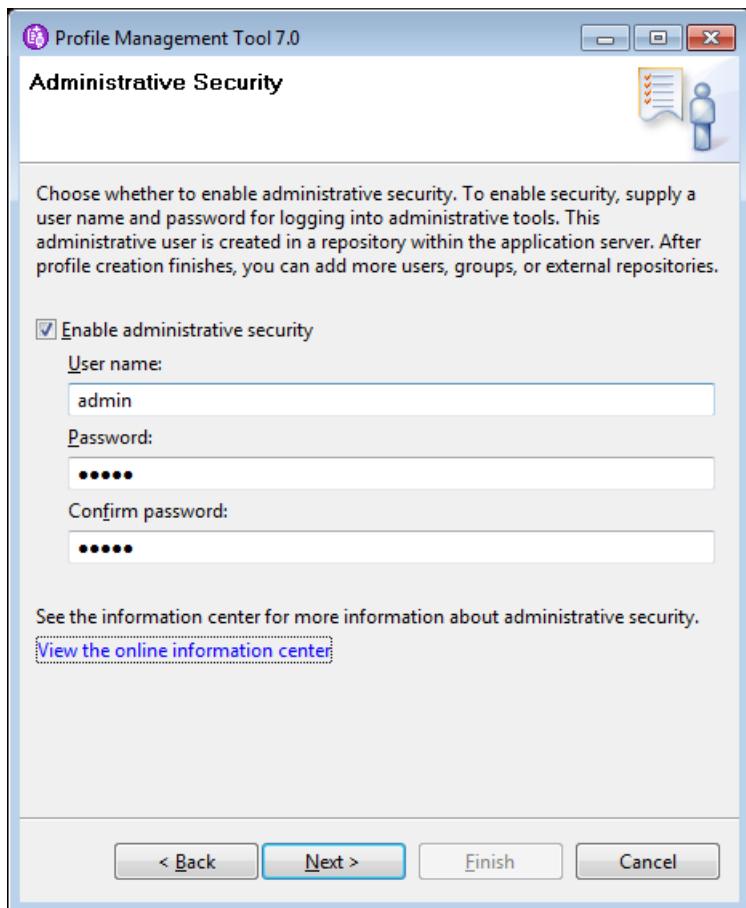


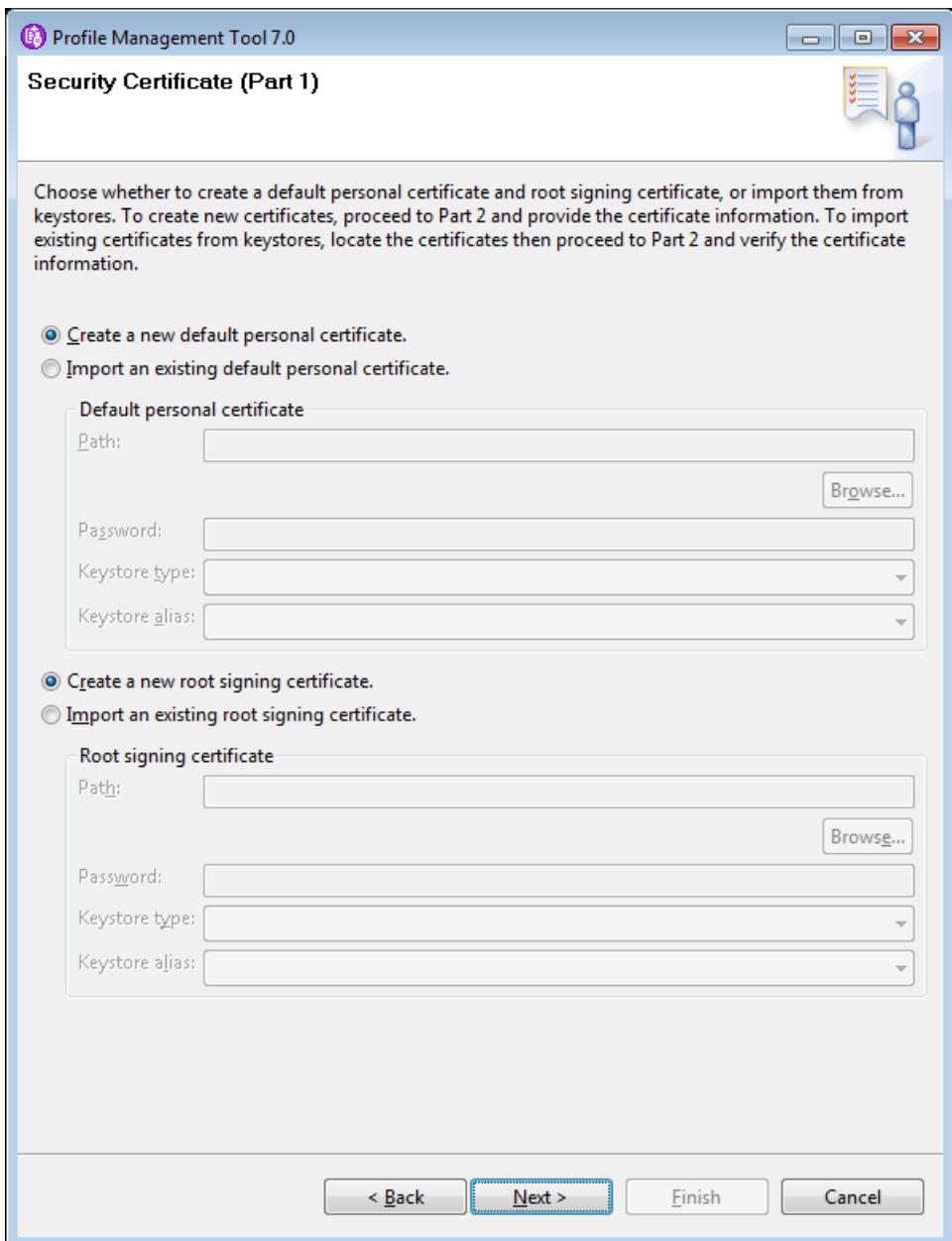


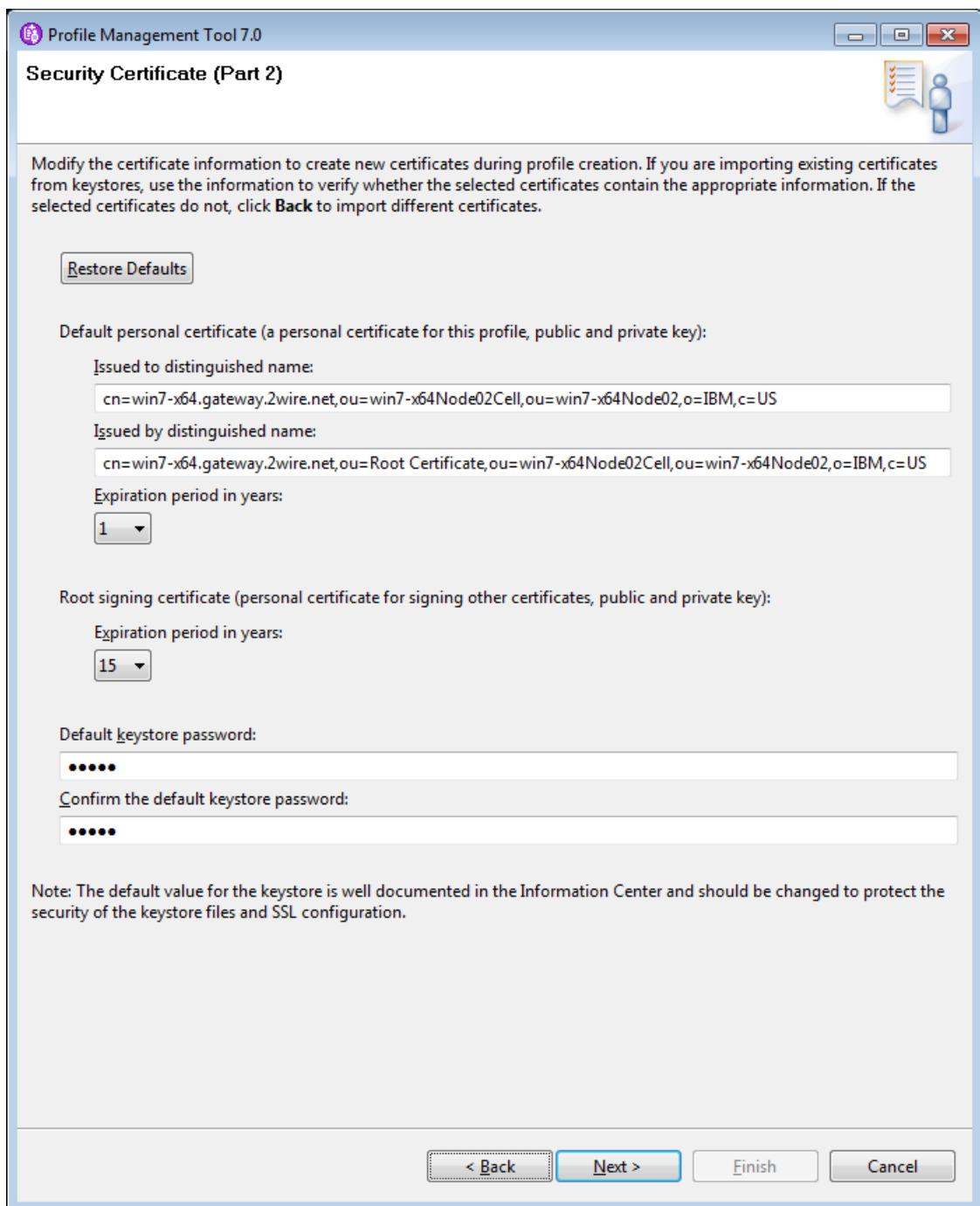


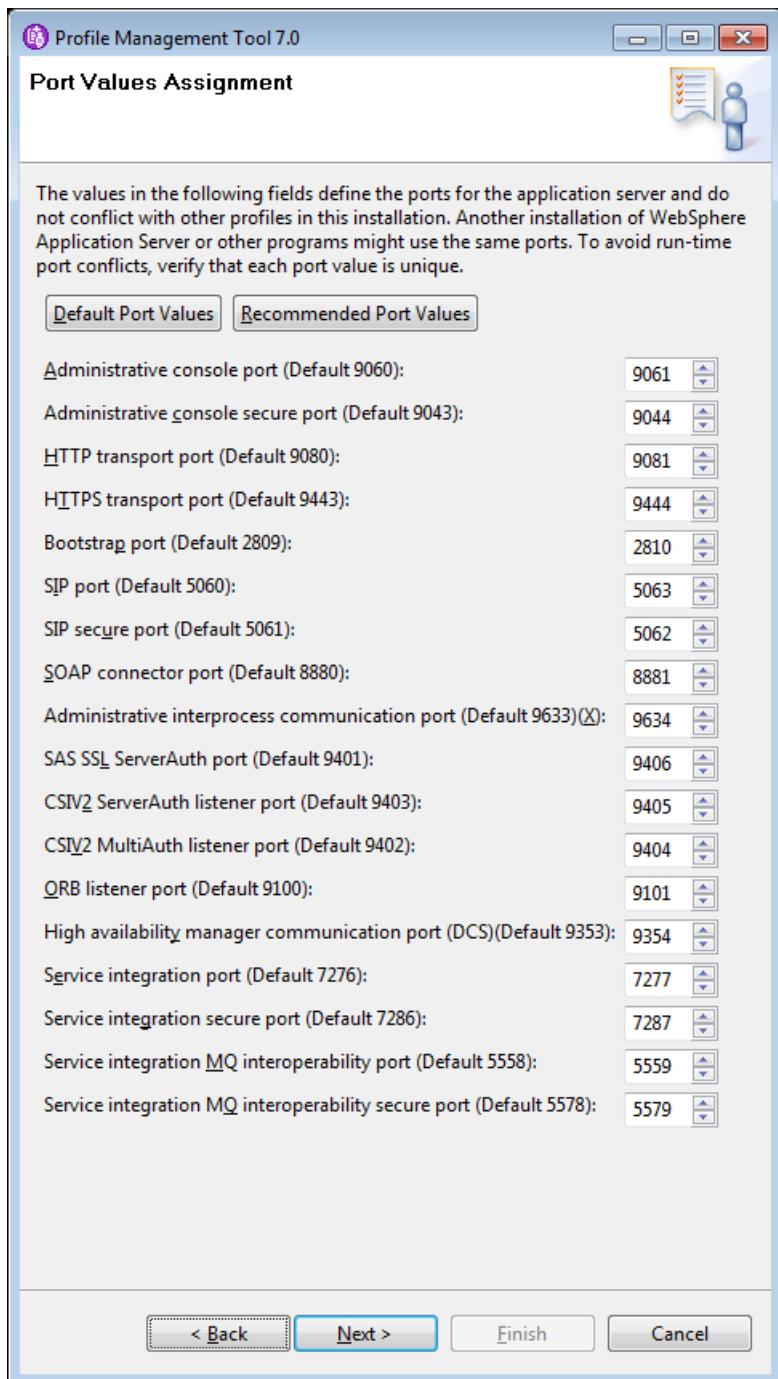


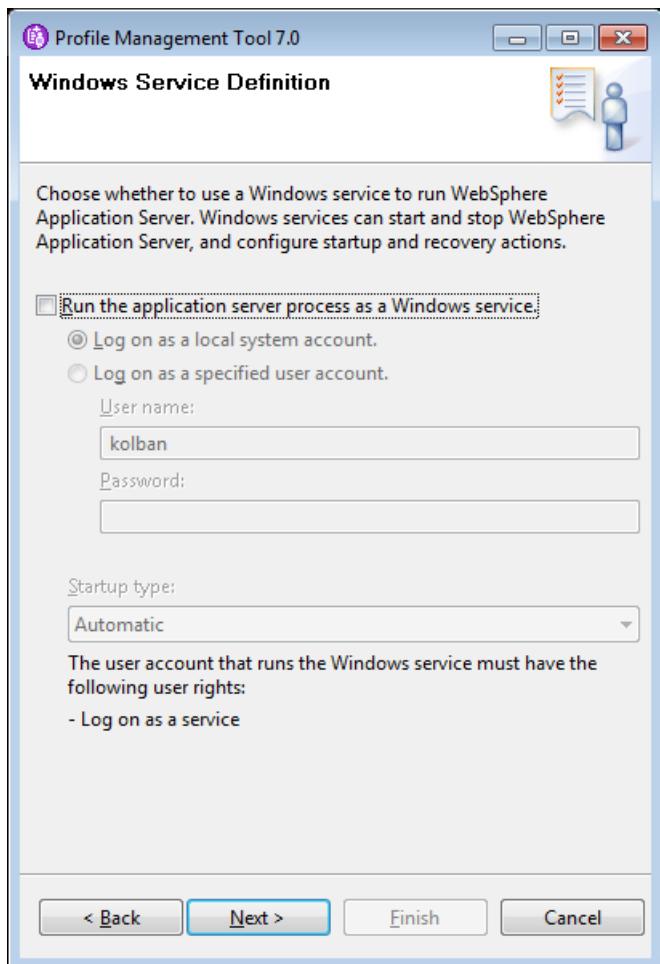


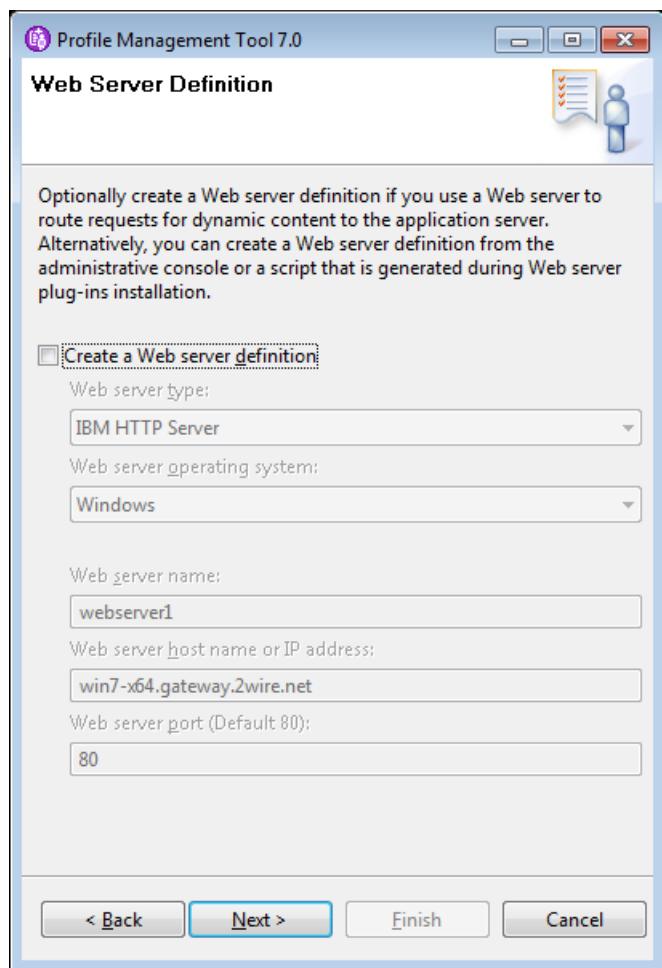


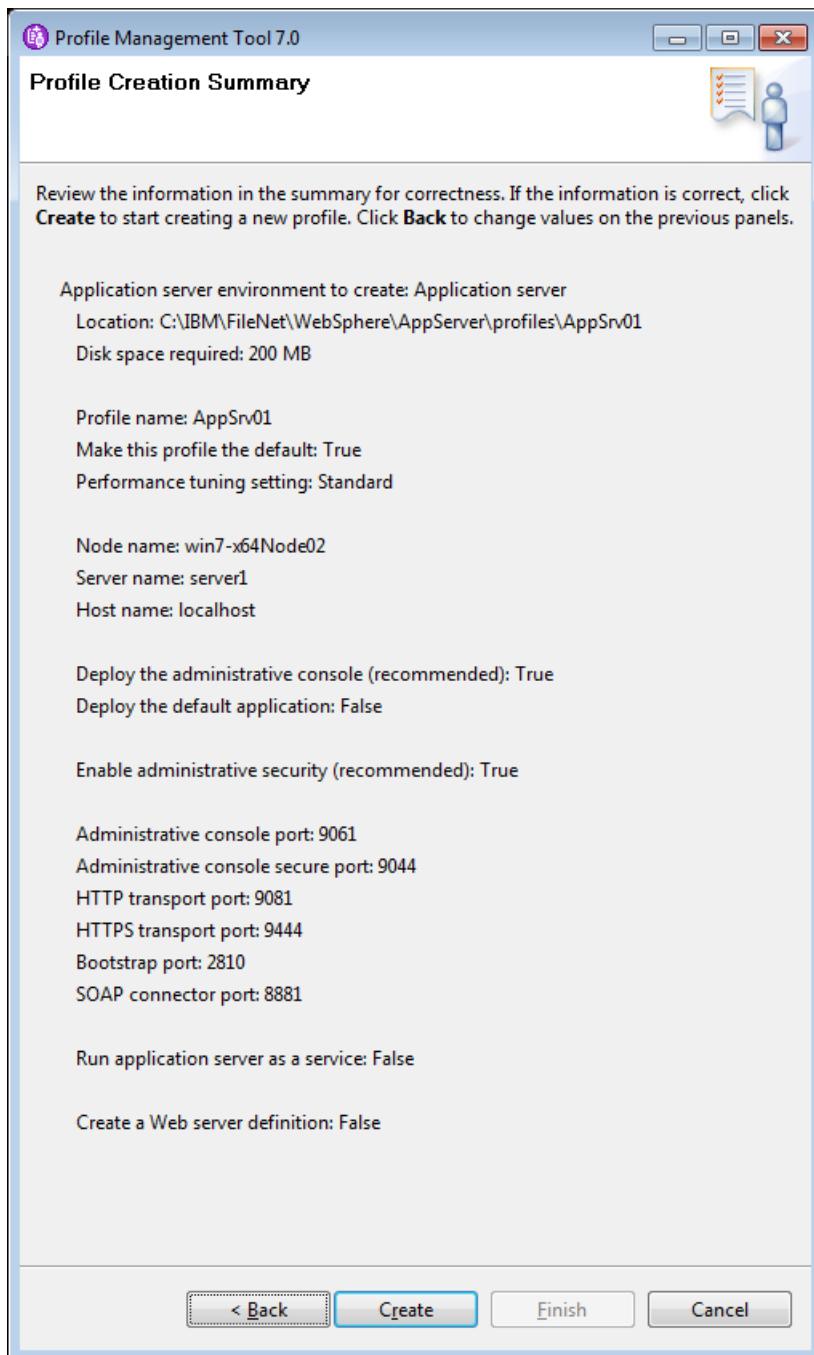




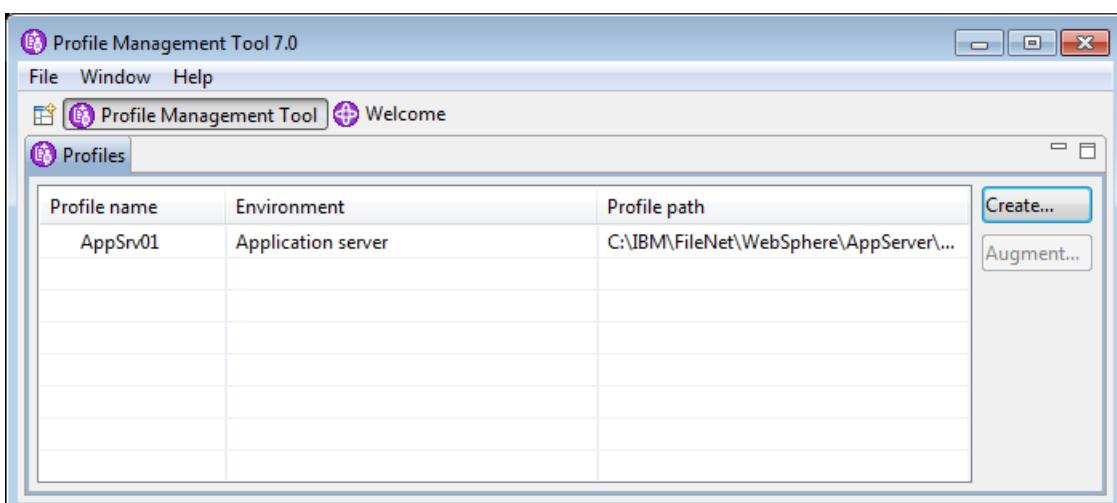
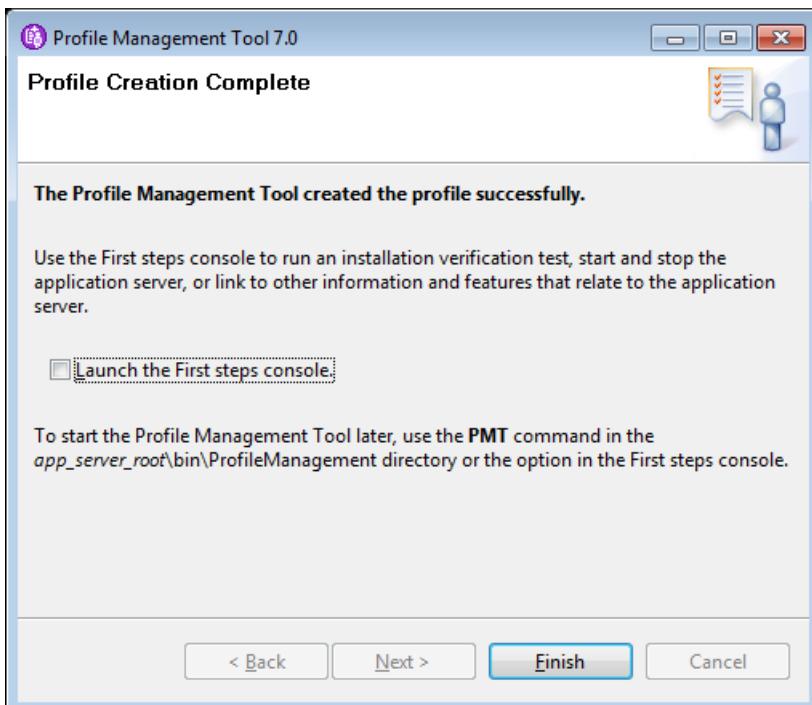






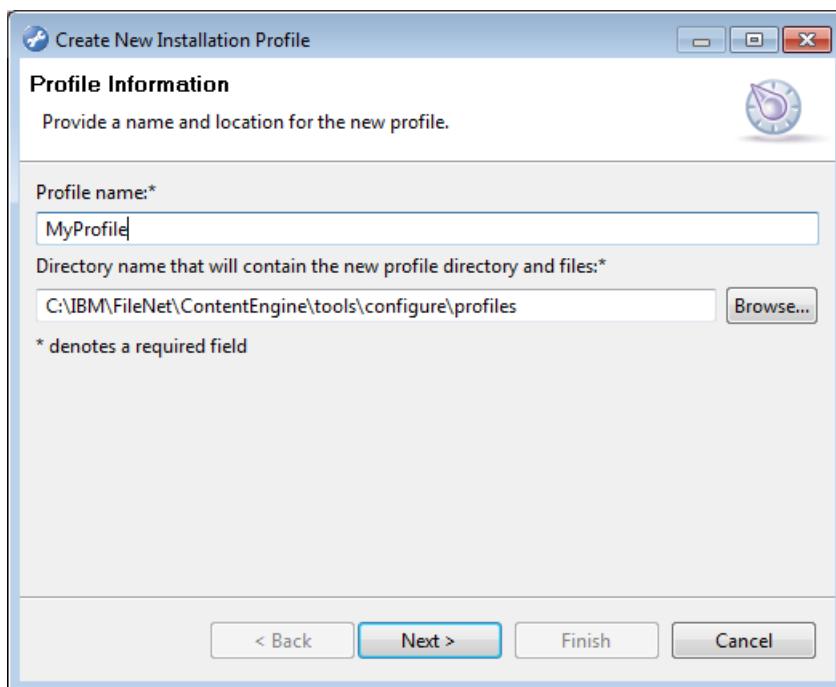
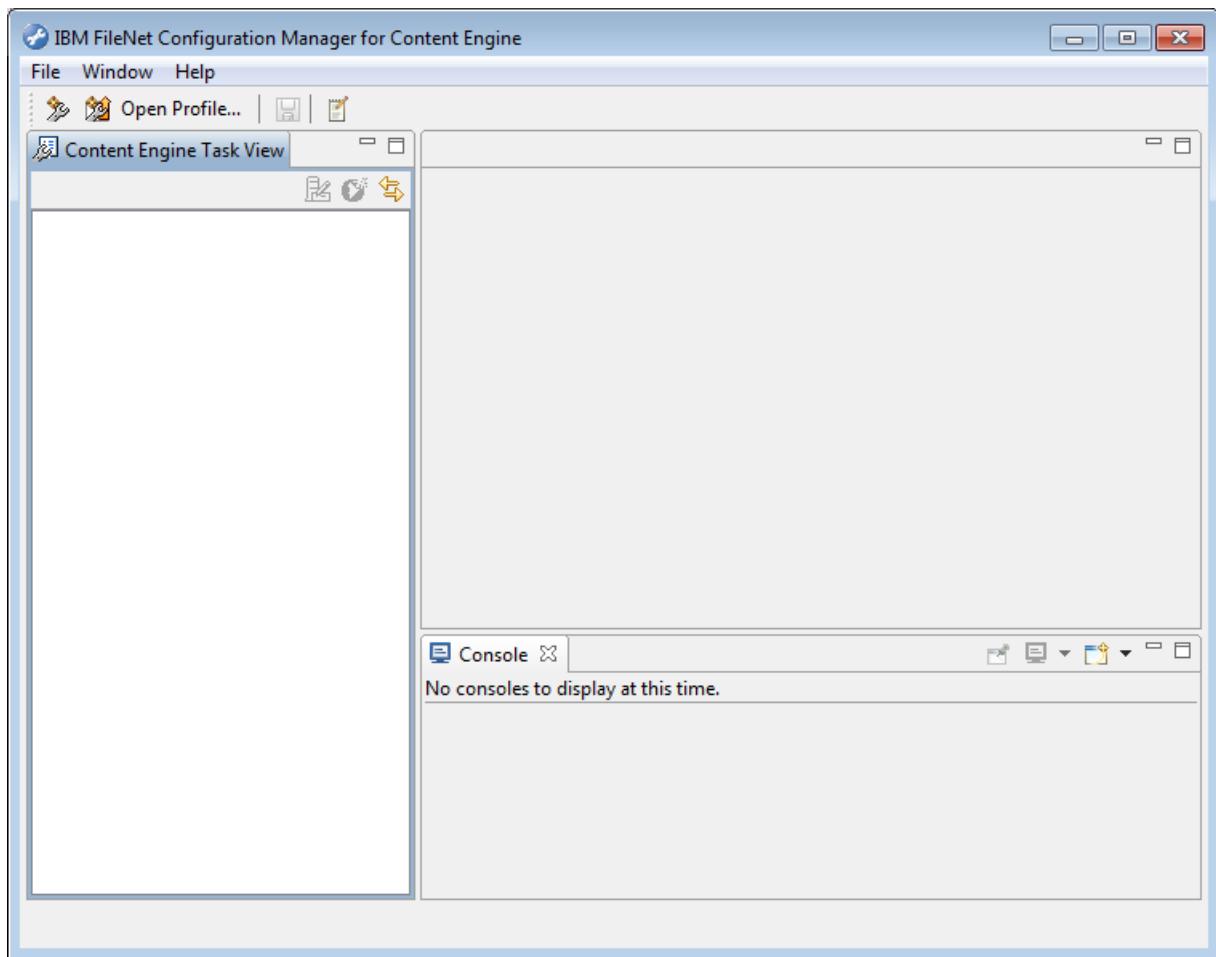


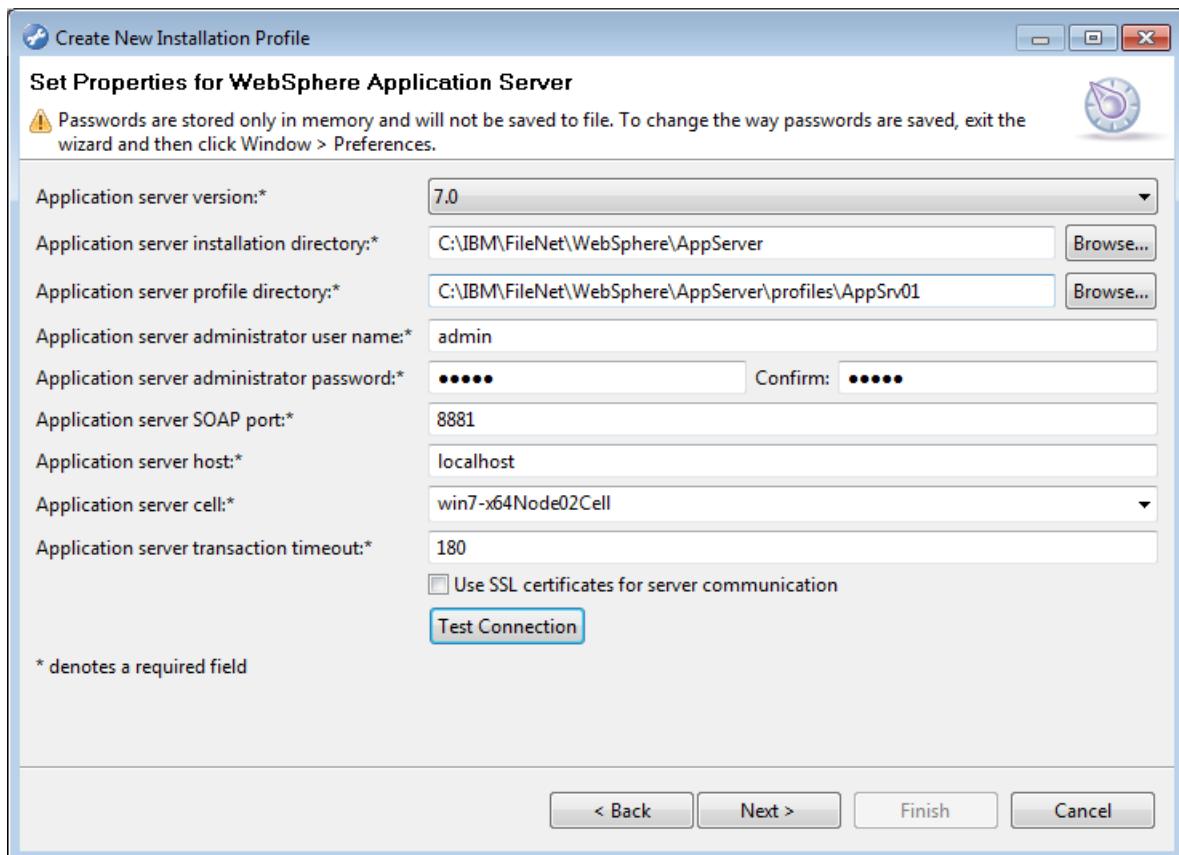
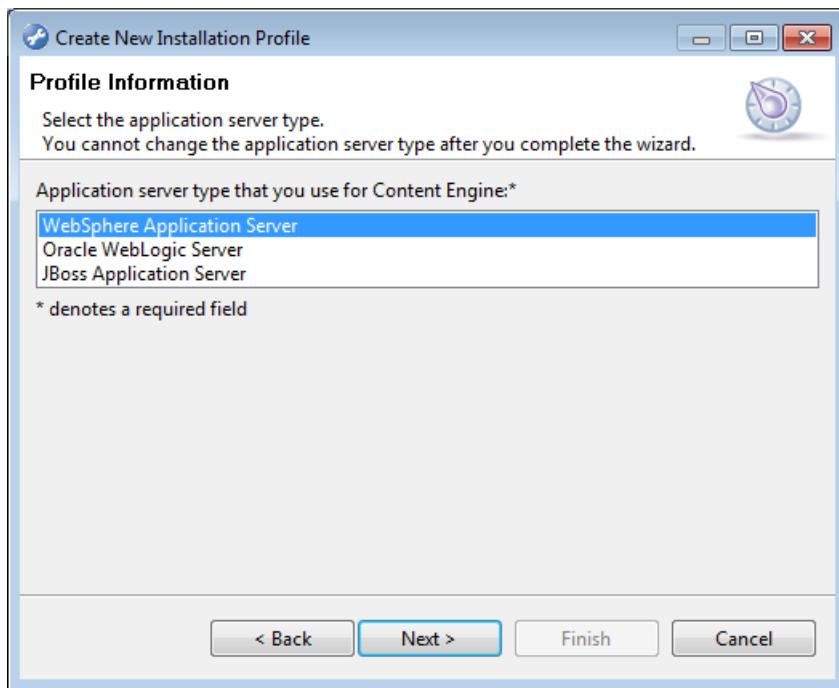
After this we select Create and the process of creating a WAS server profile begins.

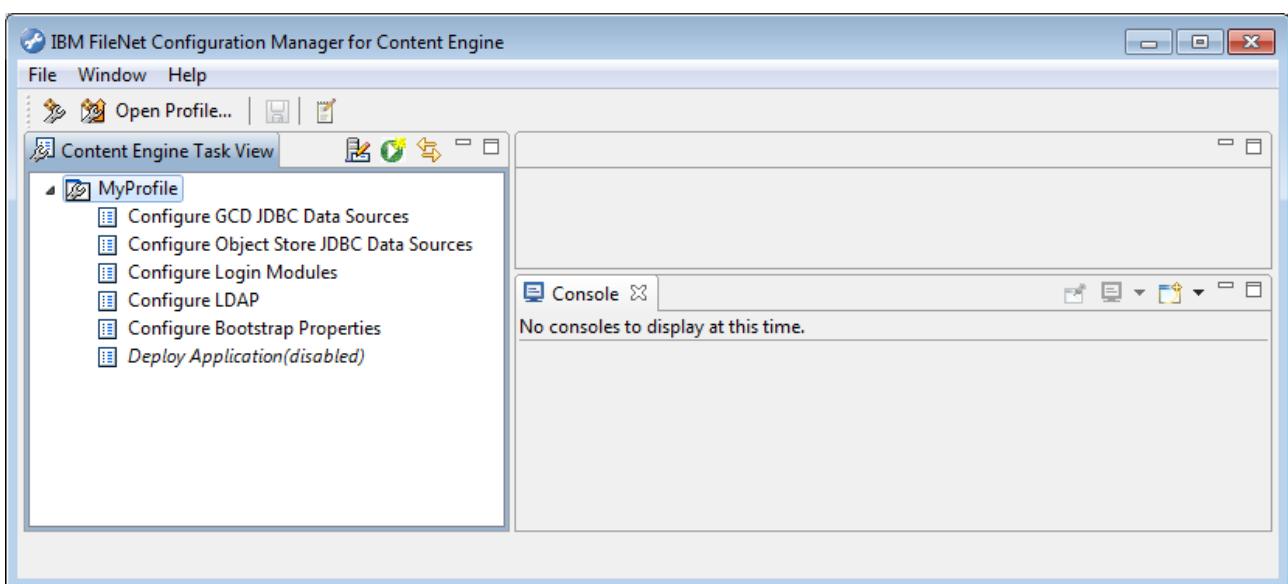
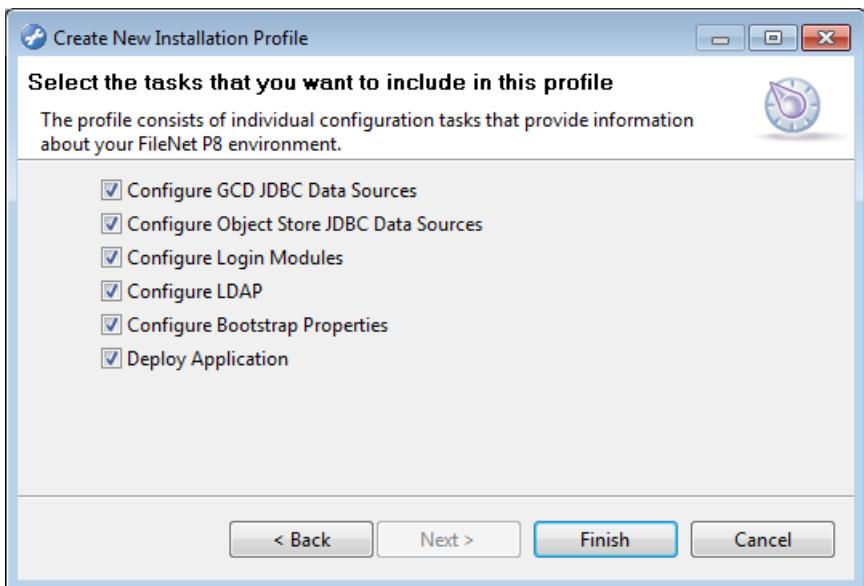


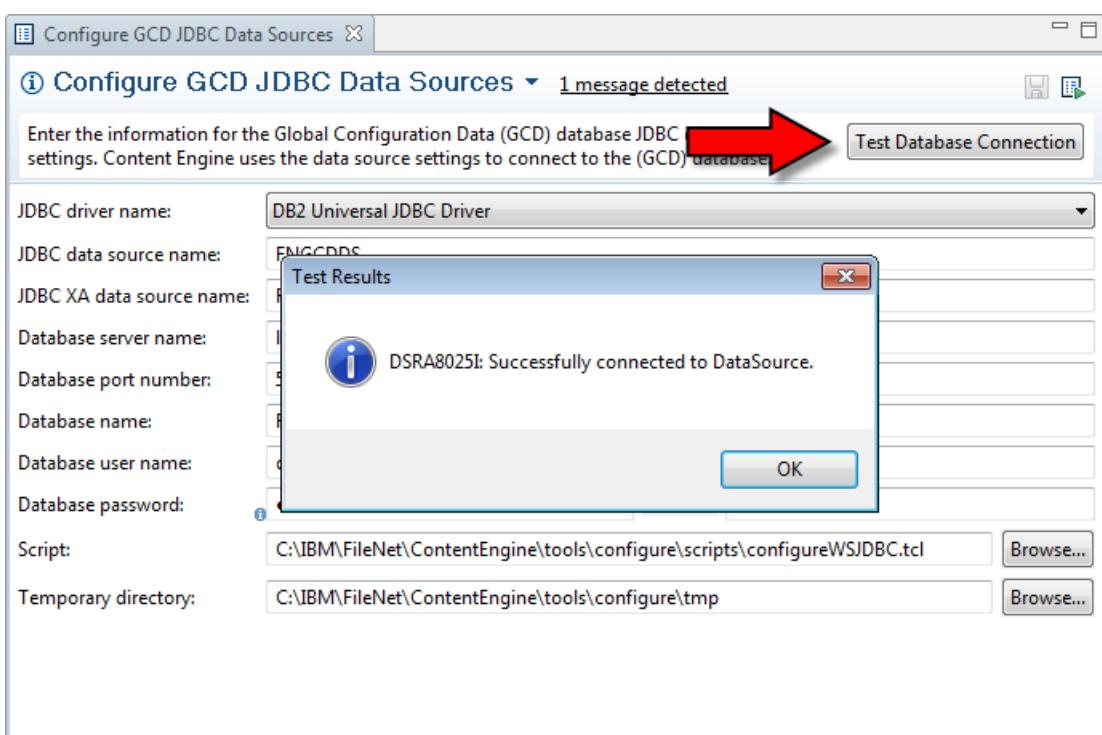
Now that WAS is installed, we can open up FileNet Configuration Manager. Make sure that the AppServer is already running. Also ensure that the following WAS environment variables are set correctly:

<input type="checkbox"/>	<a href="#">DB2UNIVERSAL JDBC DRIVER_NATIVEPATH</a>	C:\Program Files (x86)\IBM\SQLLIB\BIN	Node=win7-x64Node02
<input type="checkbox"/>	<a href="#">DB2UNIVERSAL JDBC DRIVER_PATH</a>	C:\Program Files (x86)\IBM\SQLLIB\java	Node=win7-x64Node02









Running this task produces the following output:

```
Starting to run Configure GCD JDBC Data Sources
```

```
Configure GCD JDBC Data Sources ****
Finished running Configure GCD JDBC Data Sources
```

Configure Object Store JDBC Data Sources

Enter the information for the object store JDBC data source settings. Content Engine uses the data source settings to connect to the object store database.

Test Database Connection

JDBC driver name:	DB2 Universal JDBC Driver
JDBC data source name:	FNOSDS
JDBC XA data source name:	FNOSDSXA
Database server name:	localhost
Database port number:	50000
Database name:	FNDB
Database user name:	db2admin
Database password:	*****
Script:	C:\IBM\FileNet\ContentEngine\tools\configure\scripts\configureWSJDBC.tcl
Temporary directory:	C:\IBM\FileNet\ContentEngine\tools\configure\tmp

Configure Object Store JDBC Data Sources

Enter the information for the object store JDBC data source settings. Content Engine uses the data source settings to connect to the object store database.

Test Database Connection

Test Results

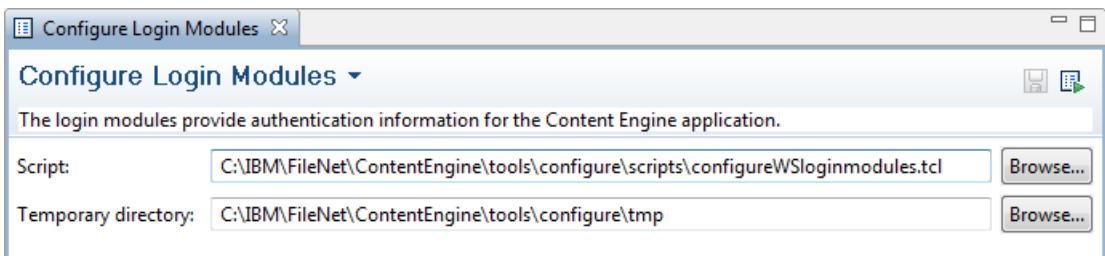
DSRA8025I: Successfully connected to DataSource.

OK

JDBC driver name:	DB2 Universal JDBC Driver
JDBC data source name:	FNOSDS
JDBC XA data source name:	FNOSDSXA
Database server name:	localhost
Database port number:	50000
Database name:	FNDB
Database user name:	db2admin
Database password:	*****
Script:	C:\IBM\FileNet\ContentEngine\tools\configure\scripts\configureWSJDBC.tcl
Temporary directory:	C:\IBM\FileNet\ContentEngine\tools\configure\tmp

Starting to run Configure Object Store JDBC Data Sources

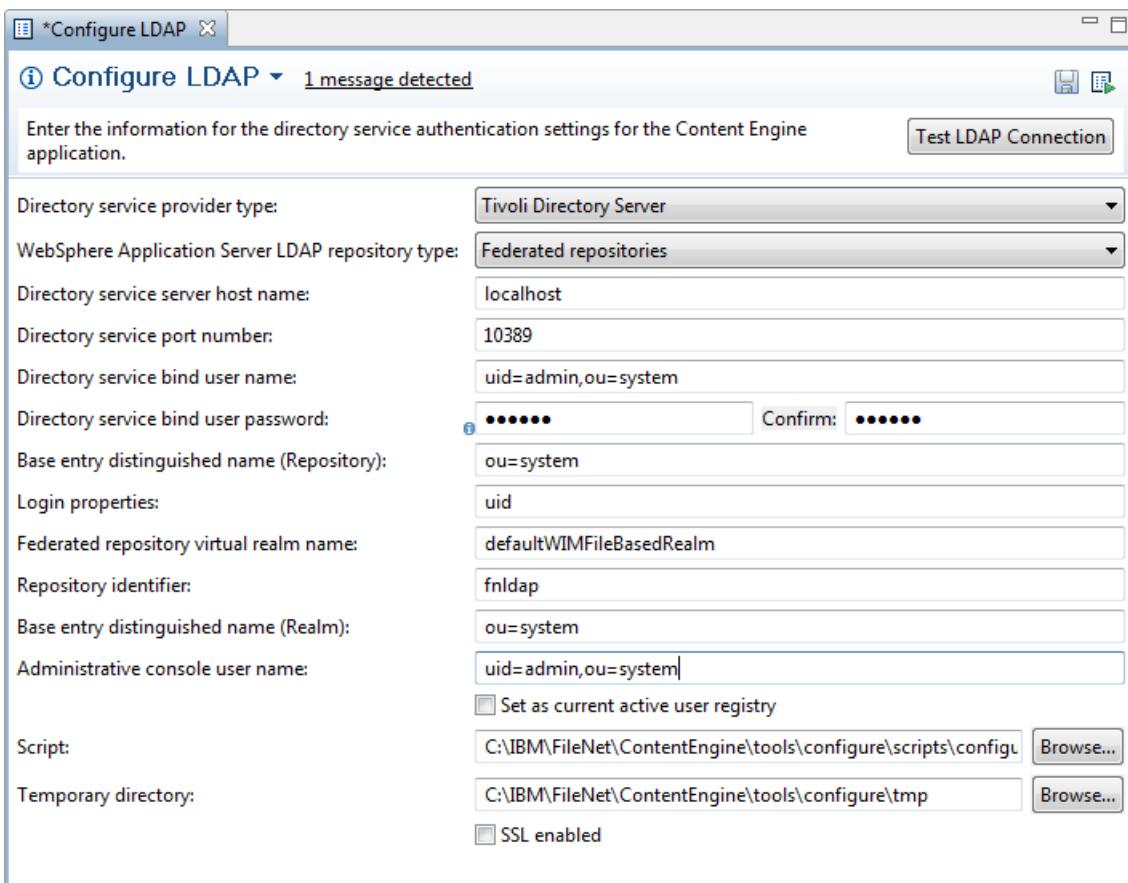
Configure Object Store JDBC Data Sources \*\*\*\*\*  
Finished running Configure Object Store JDBC Data Sources



Starting to run Configure Login Modules

Configure Login Modules \*\*\*\*\*

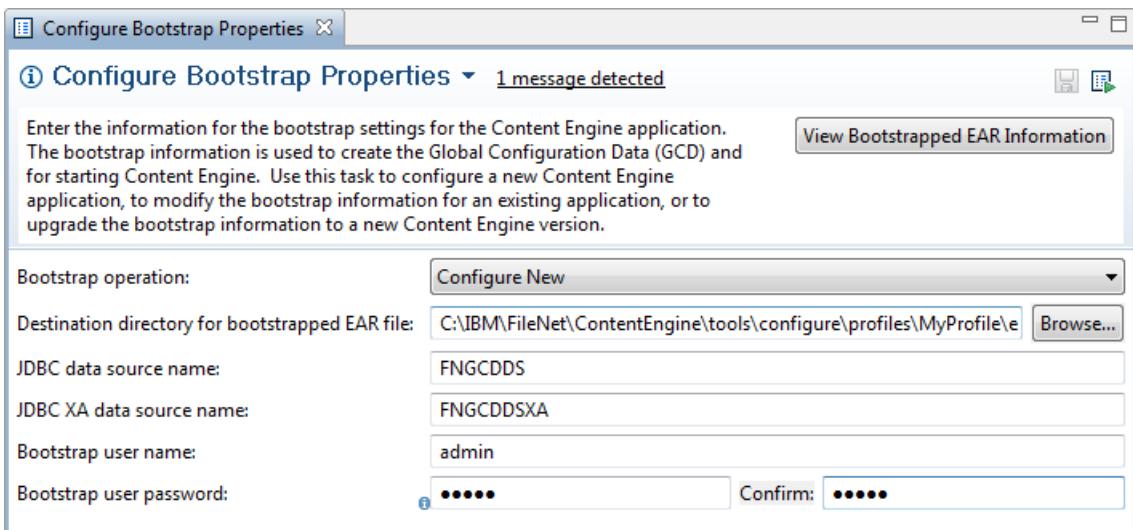
Finished running Configure Login Modules



Starting to run Configure LDAP

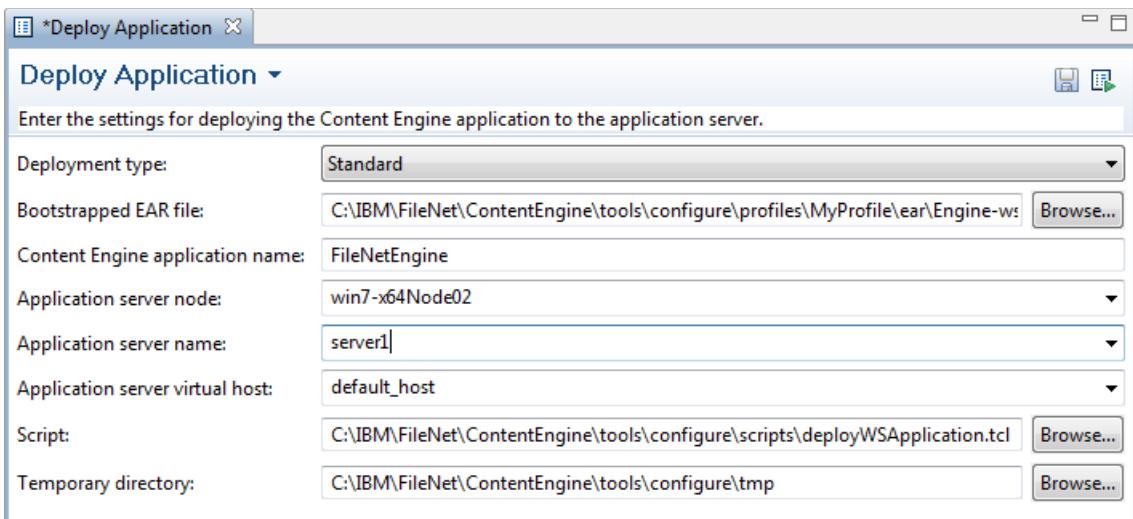
Configure LDAP \*\*\*\*\*

Finished running Configure LDAP



Starting to run Configure Bootstrap Properties

```
Configure Bootstrap Properties *****
Finished running Configure Bootstrap Properties
```



Starting to run Deploy Application

```
Deploy Application *****

Finished running Deploy Application
```

## Alfresco

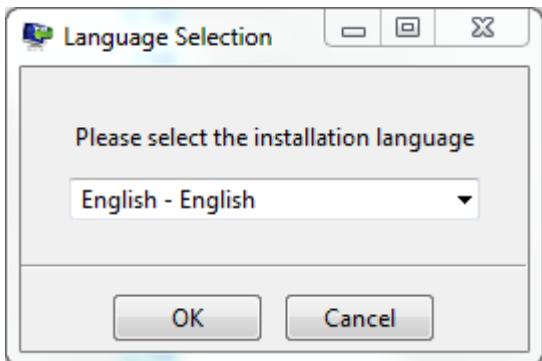
Alfresco is an open source document management system. Details about this project can be found:  
<http://www.alfresco.com/>

## Alfresco Installation

Here is an installation walk-through for Alfresco. Ensure PostgreSQL is installed as that will be used as the database for Alfresco's own operations. Next we will perform a walk-through of an

Alfresco installation.

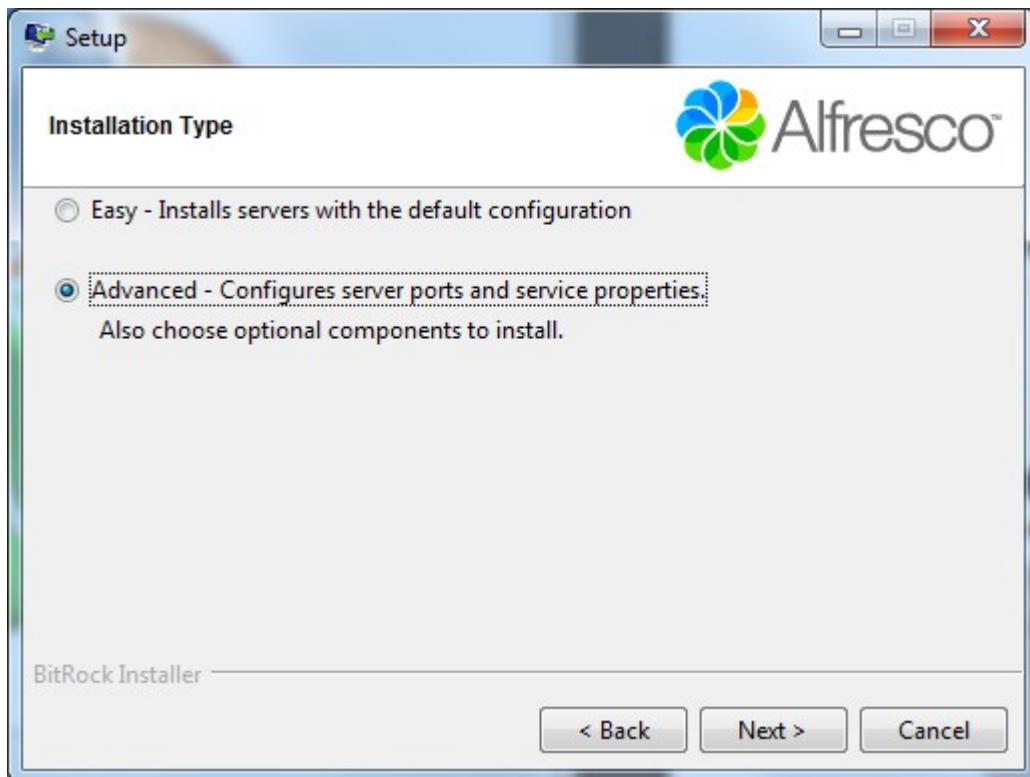
Once launched, we are asked which language we wish to use during installation:



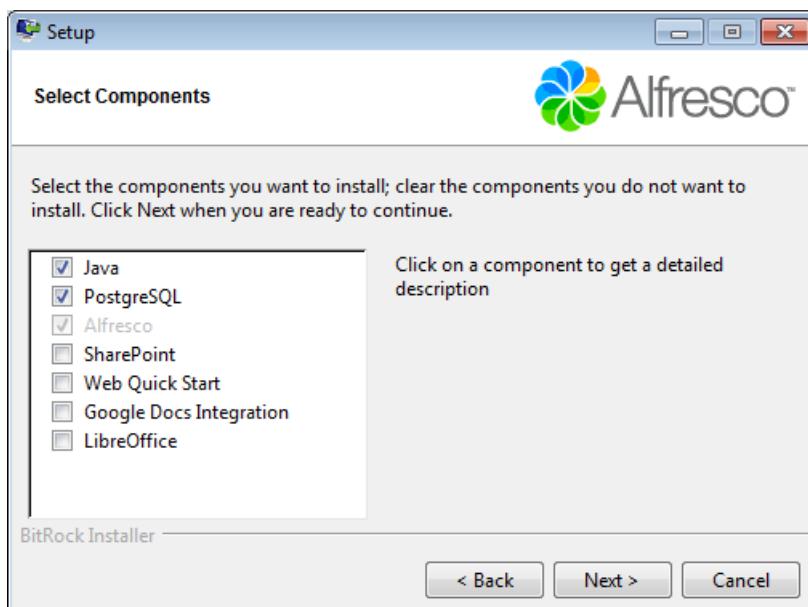
We are shown a welcome screen:



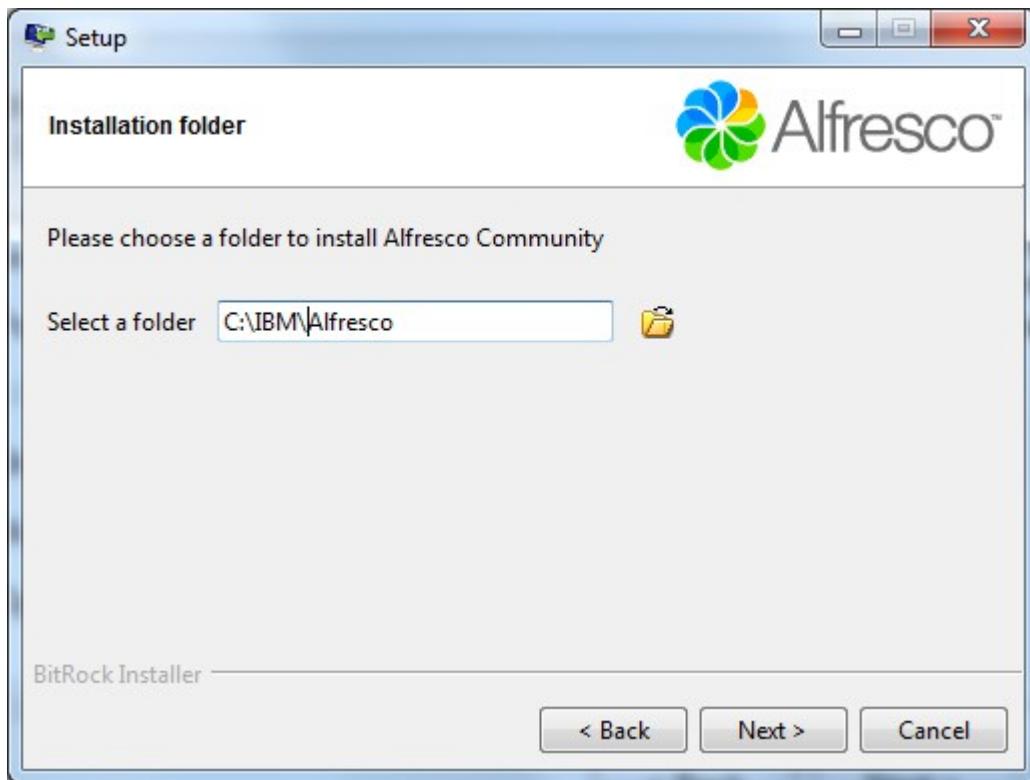
We are given the choice of what type of installation we wish to perform. The choices are Easy or Advanced. I chose Advanced:



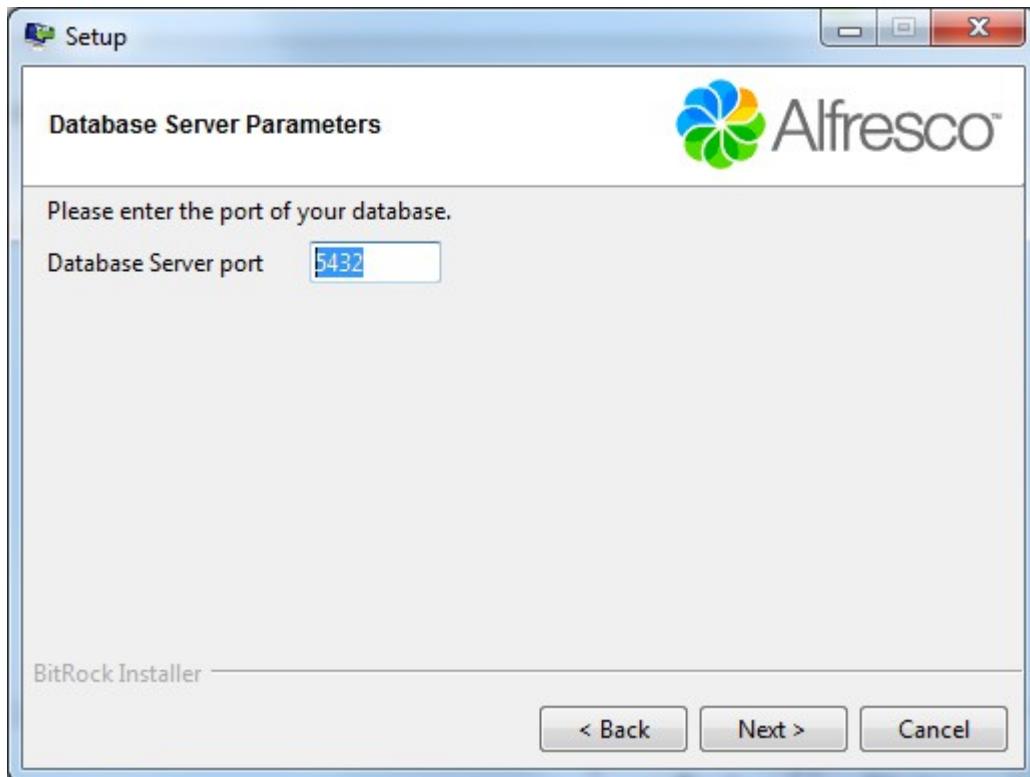
We select which components we wish to add:



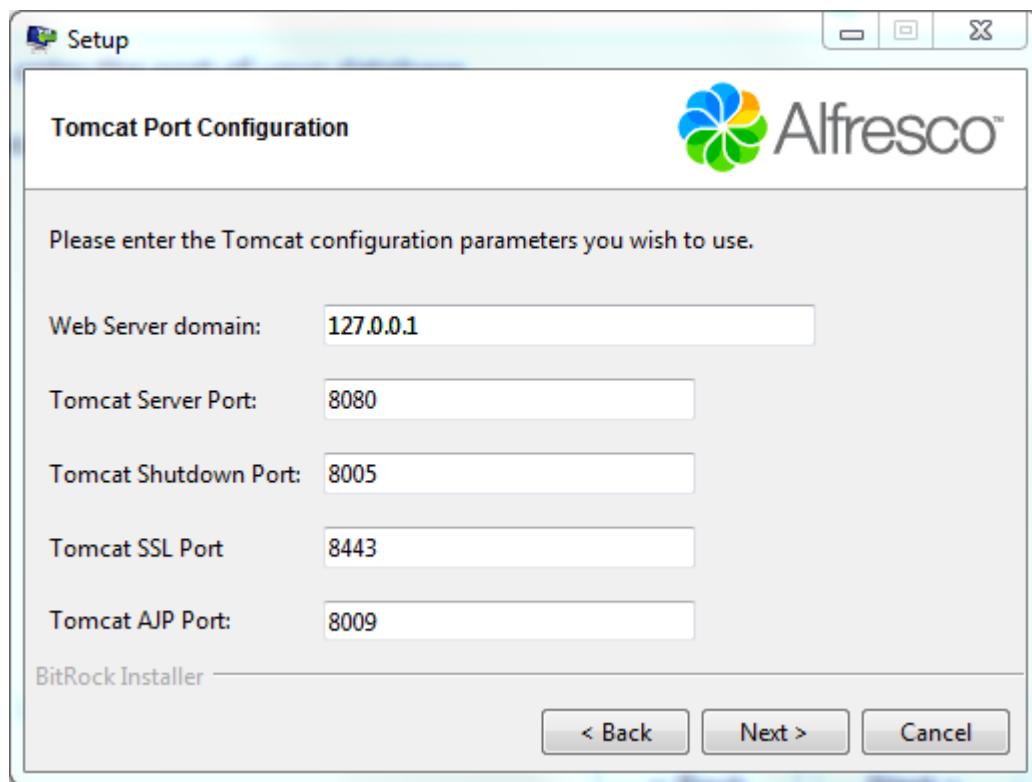
We are asked where on the local file system we wish the product to be installed:



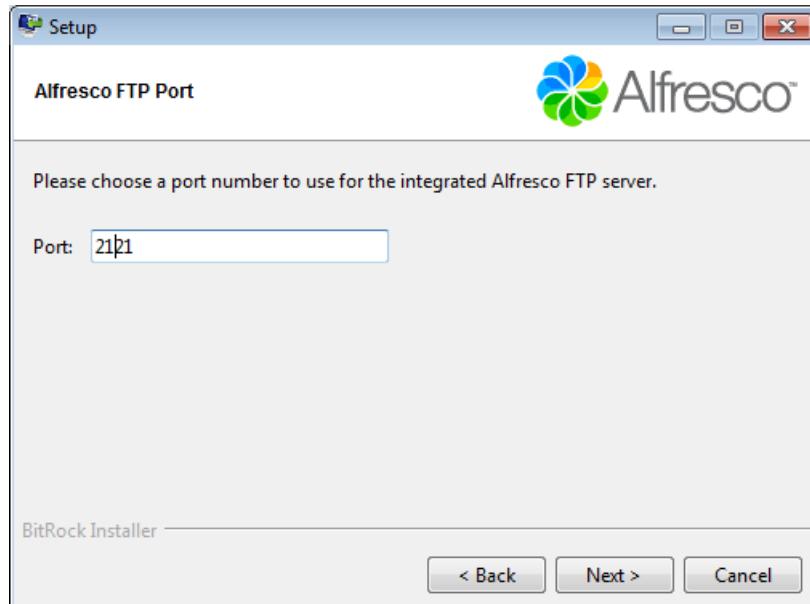
Next we are asked what port number should be used for the database:



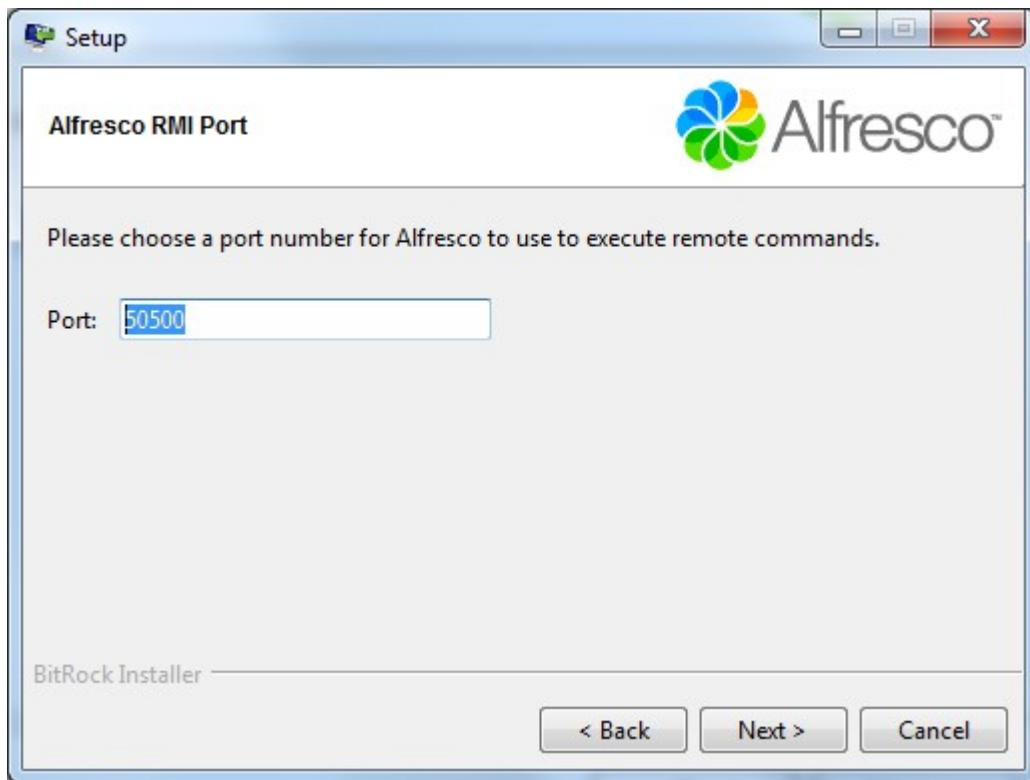
Since Alfresco runs on Tomcat, we are asked to provide properties of the Tomcat environment:



We are asked which port for FTP should be used:



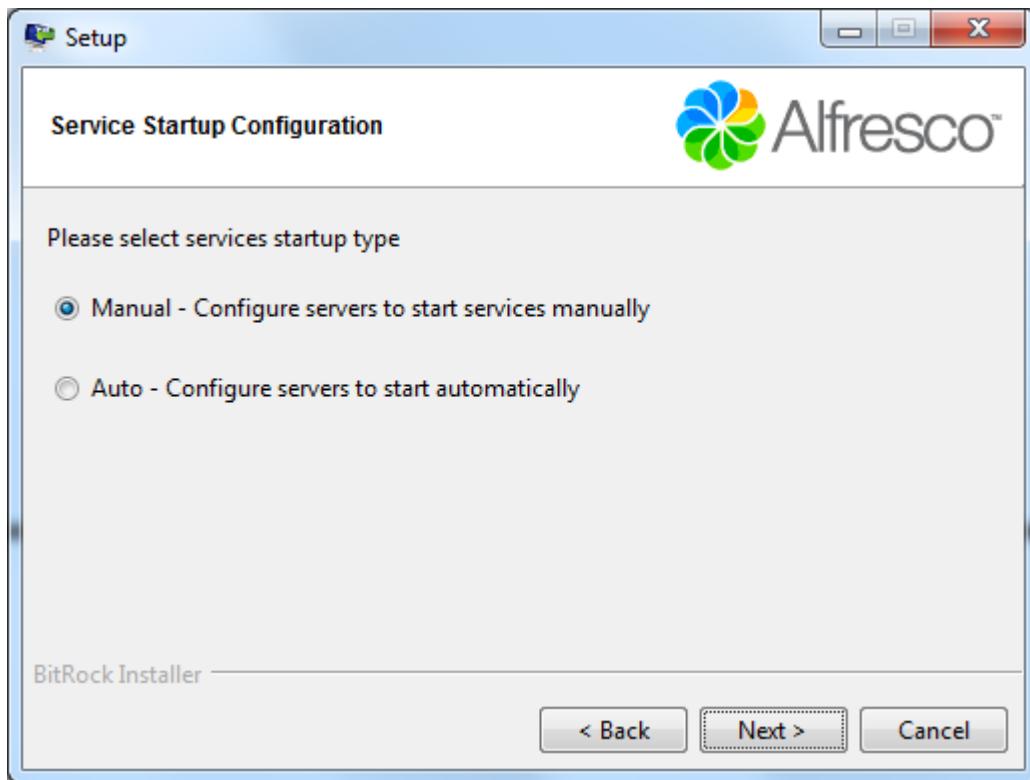
Now we are asked for the port for remote commands:



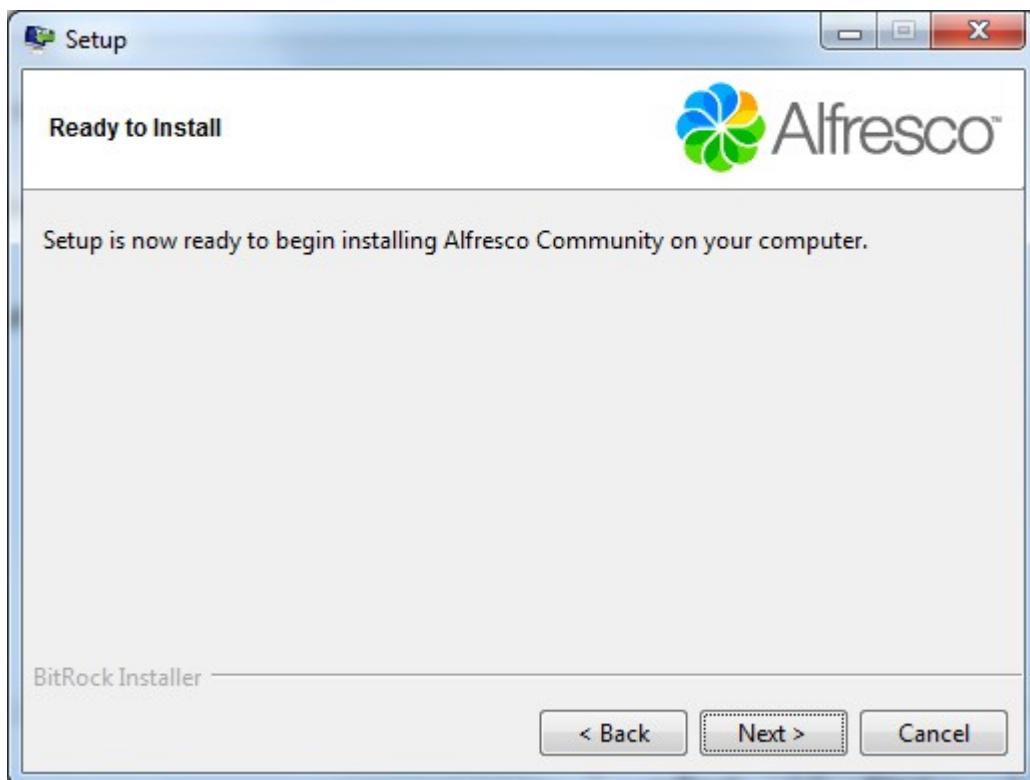
Now we supply an administrator password. I recommend admin/admin for testing purposes.



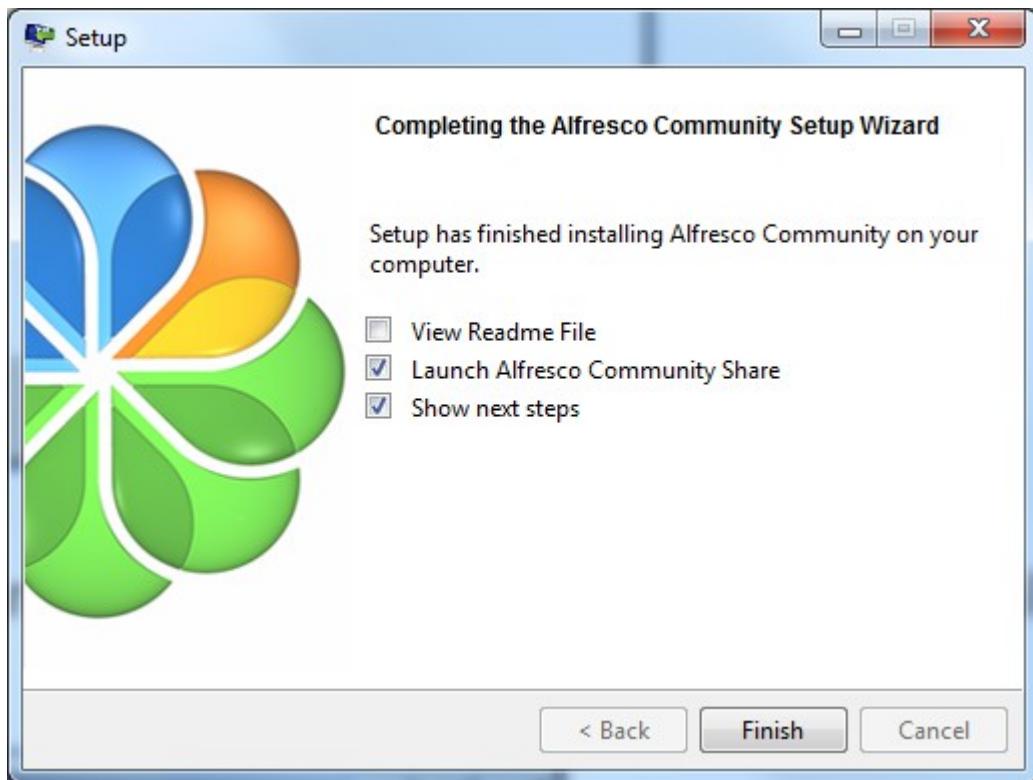
Next we are asked whether Alfresco should start at system start-up time:



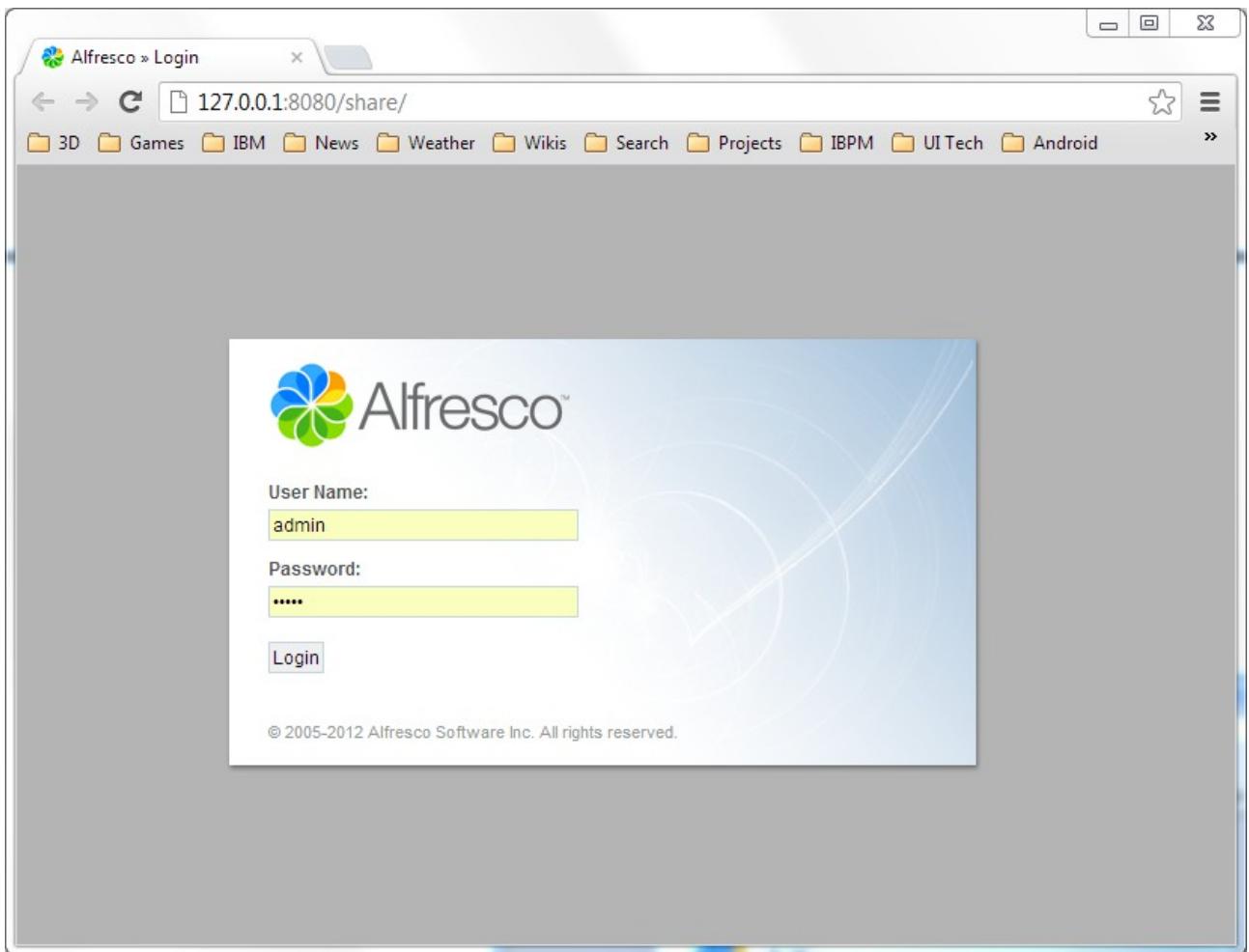
We have now supplied all the information necessary for the installation and configuration to proceed:



When the installation completes, we are presented with a next-steps screen:



We can now login to the Alfresco system dashboard. This can be accessed from a browser at:  
<http://localhost:8080/share/>



The first start of an Alfresco server may take a little while as the databases appears to be created and populated.

When documents are added to Alfresco, those documents are **not** immediately seen in a search. This means that if we create a document and then perform a query, the new document **may** not show up. This has serious implications for IBM BPM. The Document List Coach View allows us to upload documents and then immediately refreshes its view by performing a search. If the search does not show the document, the end user may not realize that the document has already been uploaded.

Digging into this story, we find that the search engine used by *default* in Alfresco is called "solr". This search engine periodically indices its data (by default once every 15 seconds). This means there is a latency between a new document addition and it showing in a search. If we can not accept this, we can ask Alfresco to utilize an alternative search engine called "lucene" that maintains a constant index and hence searches for newly uploaded documents show immediately. Here is a recipe to enable the "lucene" engine.

1. Stop Alfresco
2. Edit the `alfresco-global.properties` file found in `/tomcat/shared/classes`
3. Change the line to

```
index.subsystem.name=lucene
```

4. Add the line to

```
index.recovery.mode=FULL
```

5. Remove the following lines (if present)

```
/alfresco/alf_data/solr
/tomcat/conf/Catalina/localhost/solr.xml
/tomcat/webapps/solr
```

6. Start Alfresco. It may take a while because the indices are building.

7. Stop Alfresco

8. Change the line to

```
index.recovery.mode=AUTO
```

9. Start Alfresco

### ***Alfresco debugging***

When calls are made to Alfresco and those calls fail, additional log information may be found in

<AlfrescoRoot>/tomcat/logs

specifically, the file called:

alfrescotomcat-stdout.<date>

seems to contain a wealth of goodness.

---

# WebSphere Operational Decision Management - WODM

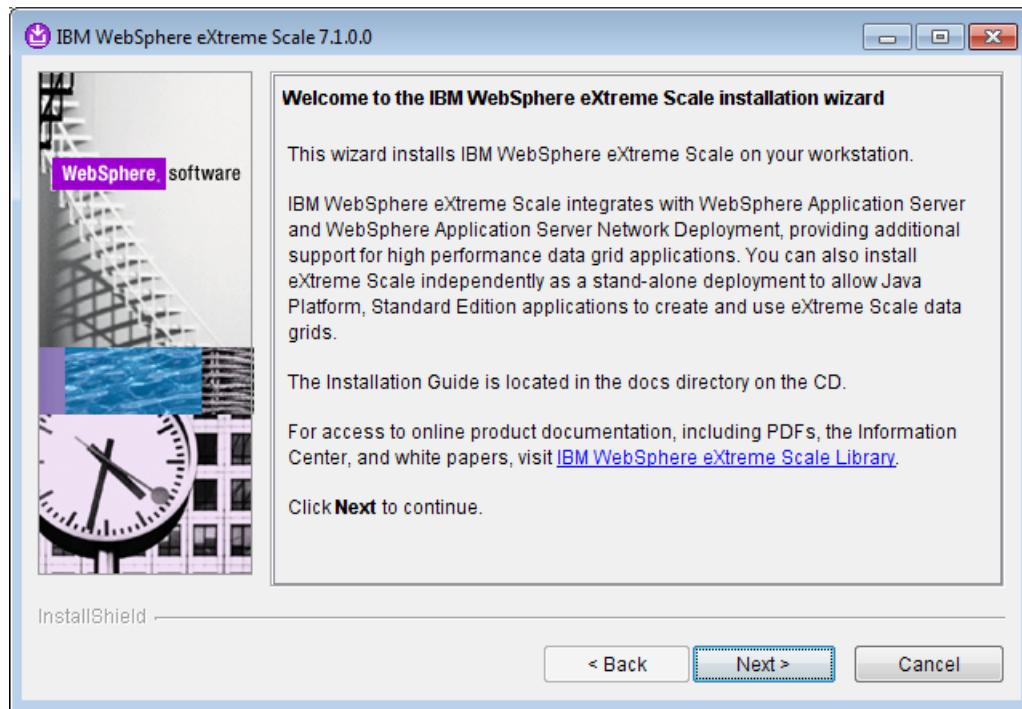
The IBM product called WebSphere Operational Decision Management (WODM) comes from the heritage of two preceding products called JRules and WebSphere Business Events (WBE).

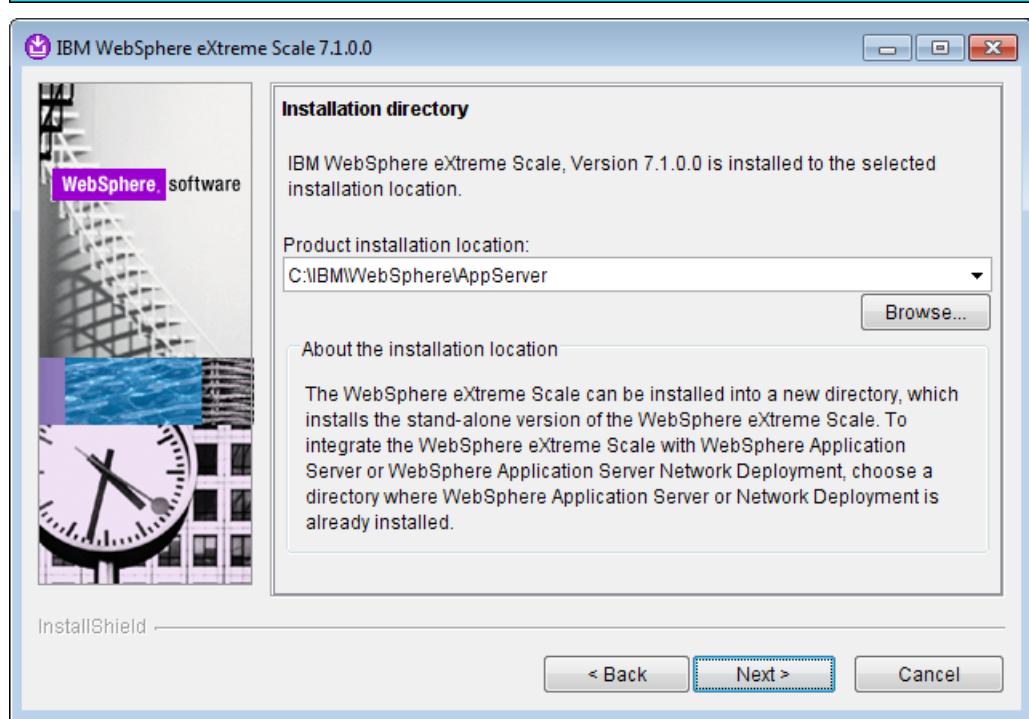
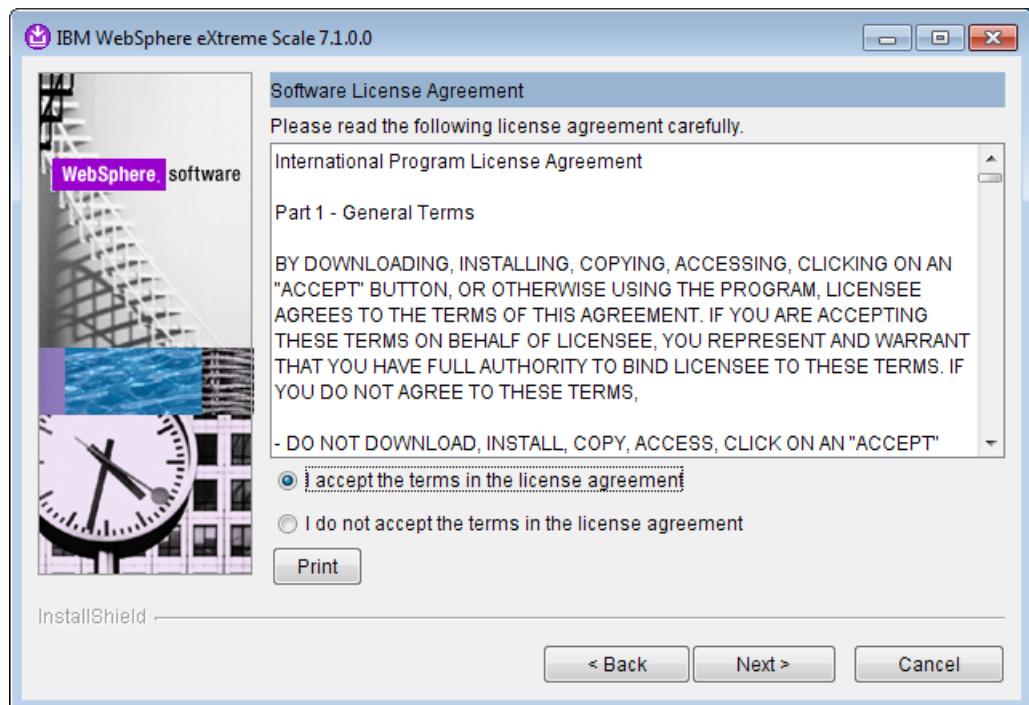
## WODM Installation

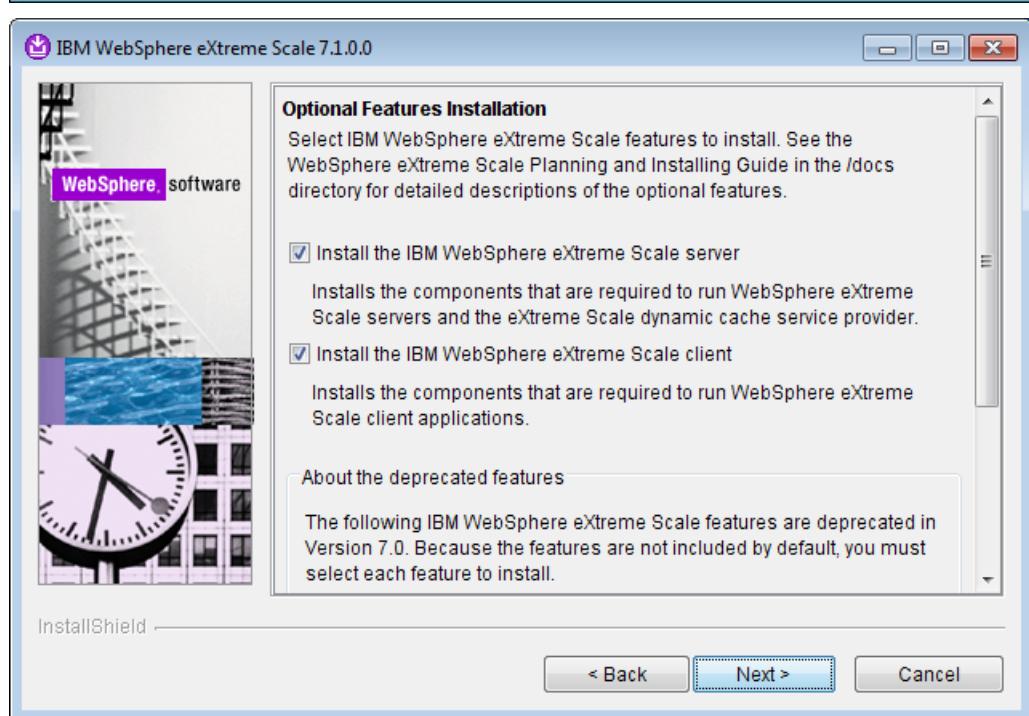
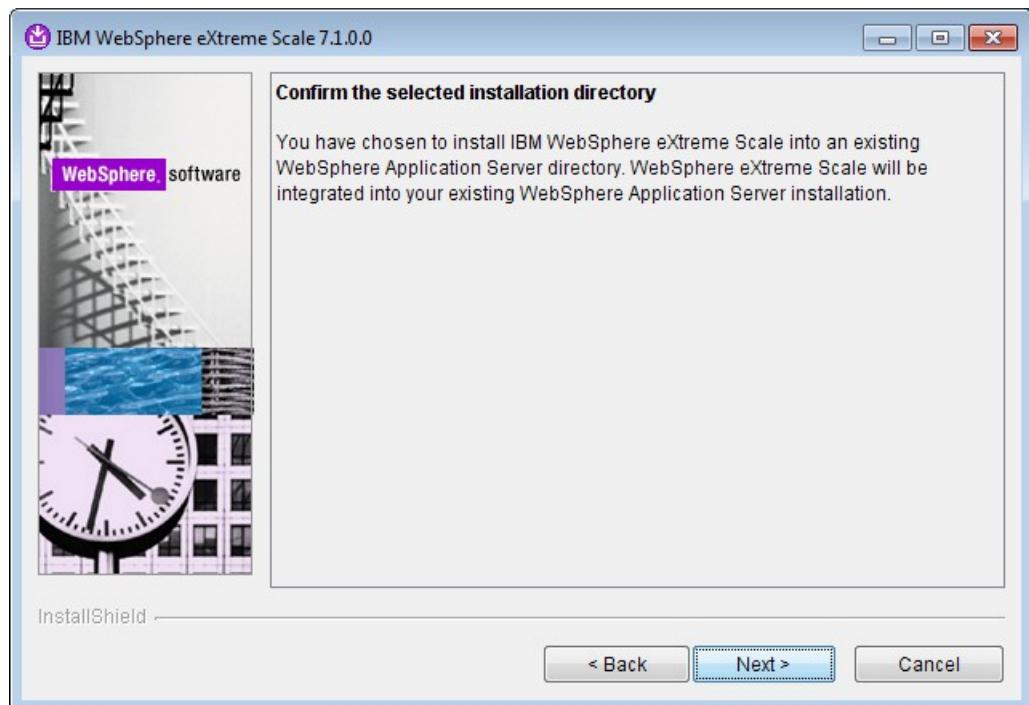
The part numbers for WODM are:

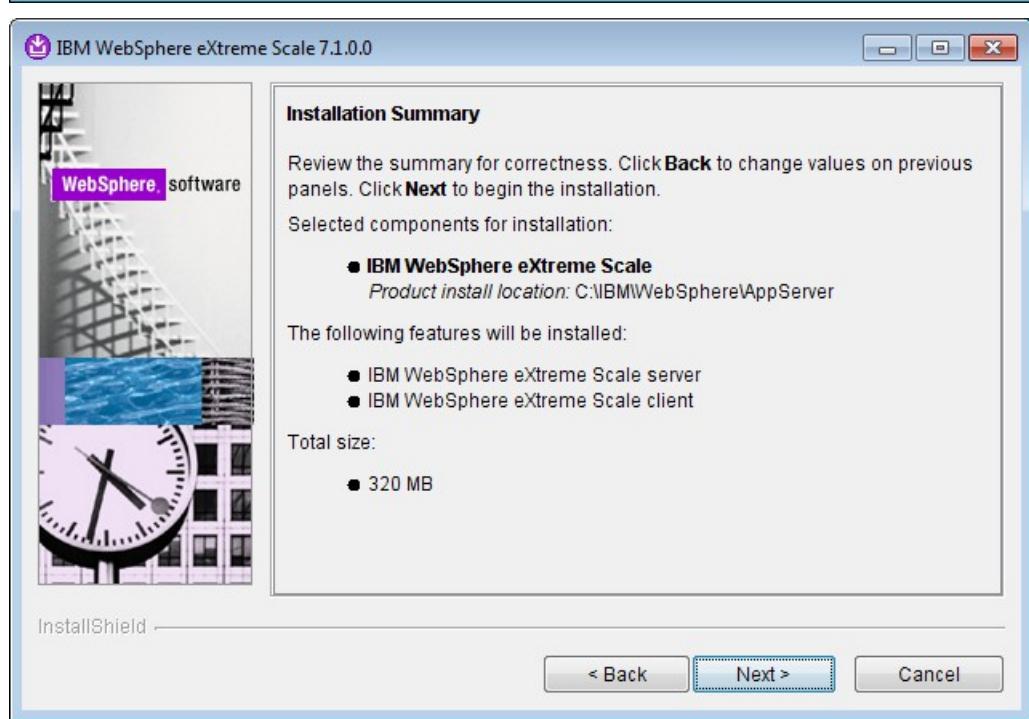
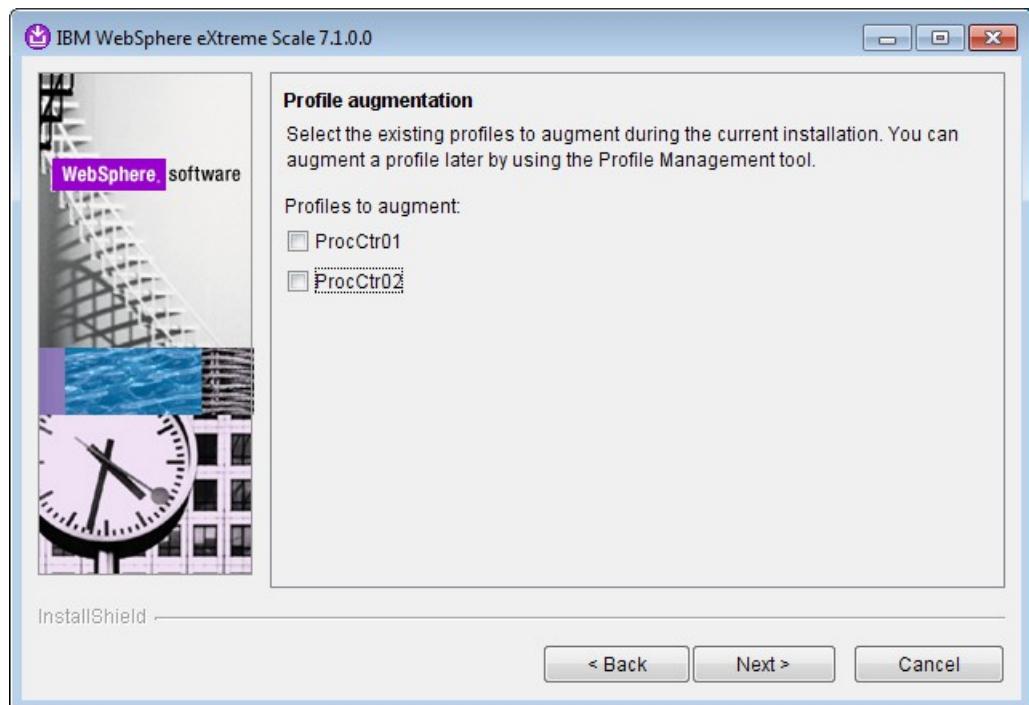
Windows – 64 Bit – 8.0.1	
Description	Part Number
eAssembly	CRKX8ML
Operational Decision Server	CID9GML
Operational Decision Center	CID9HML
Operational Decision Manager Disk 1	CID6UML
Operational Decision Manager Disk 2	CID6VML
Operational Decision Manager Disk 3	CID6WML

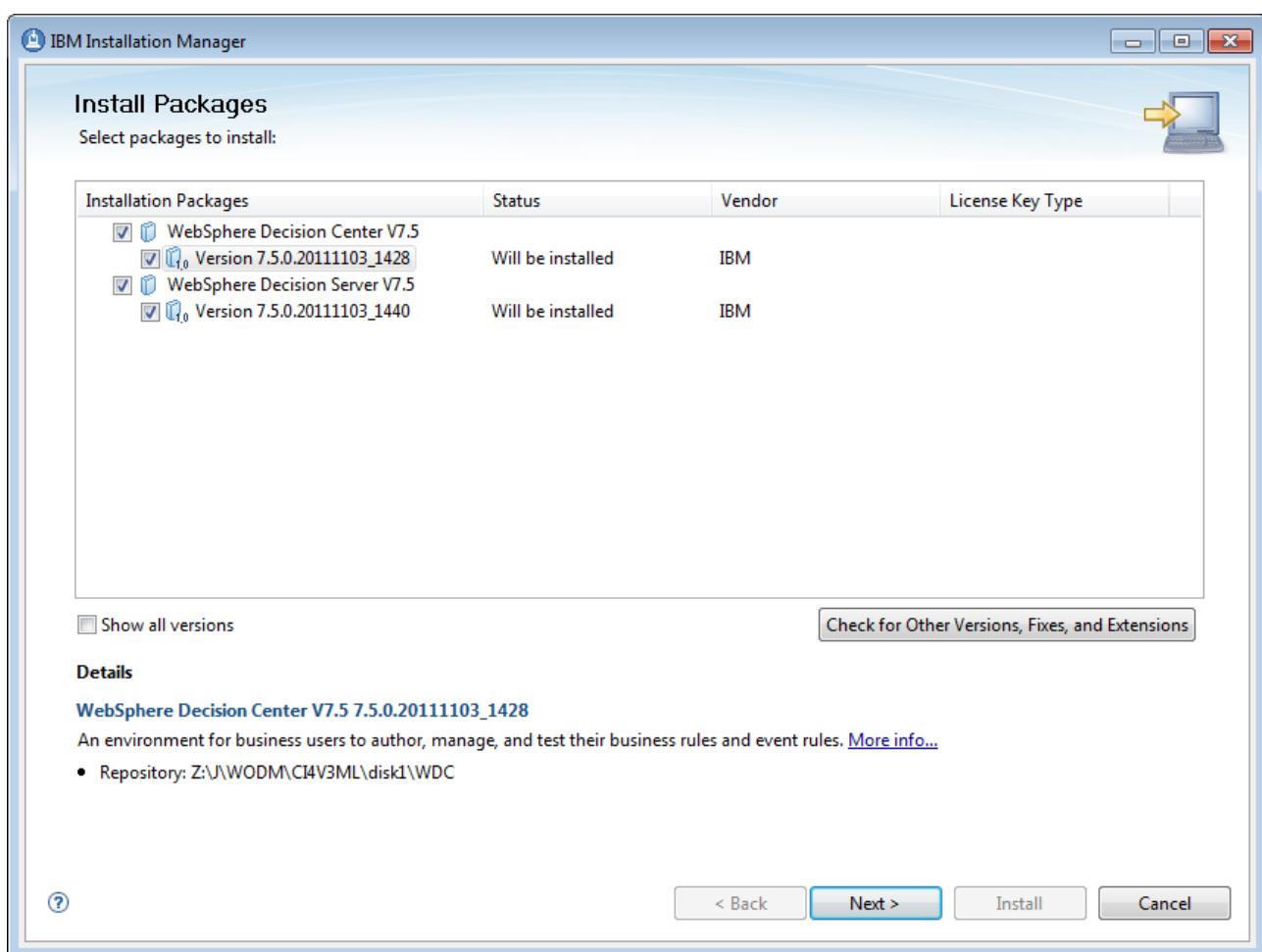
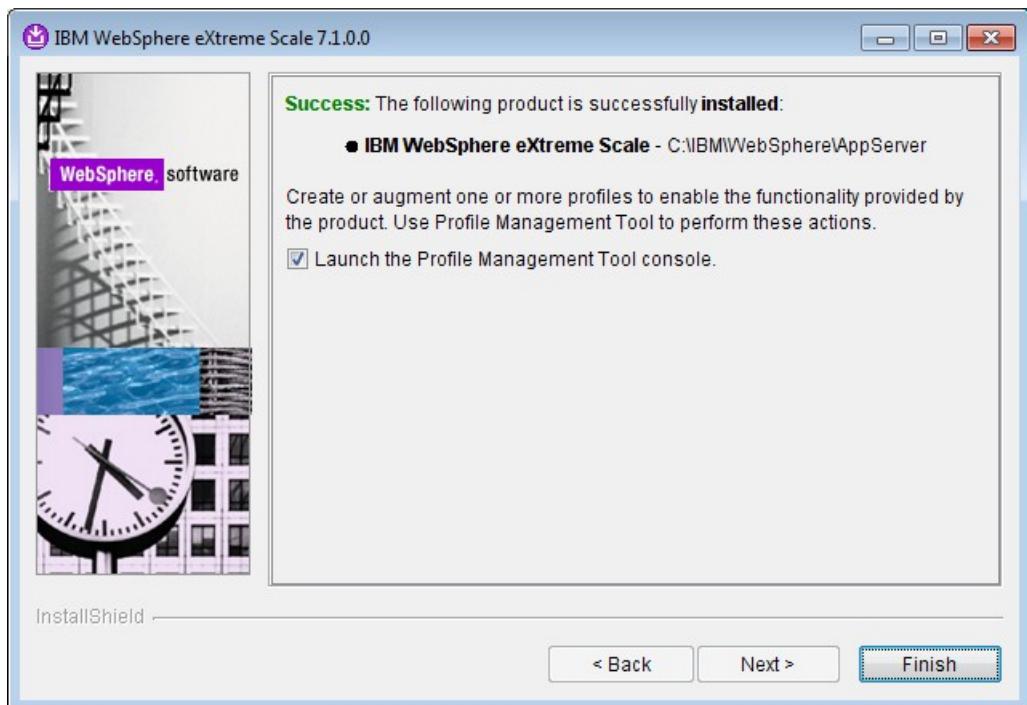
The Eventing engine requires IBM WebSphere eXtreme Scale to be installed.











Note: When installing WODM onto a IBPM Business Space, a problem was found (2012-01-01) which said that Business Space was at the wrong level. A PMR was raised and a circumvention was to edit the file called:

<WebSphere App Server>/BusinessSpace/bspace.versions

and change the line which read:

RTC.build.release=7.5.1

to

RTC.build.release=7.5.1.0

### ***Creating an instance of a Decision Server***

Decision Server is distributed as a Java EE EAR application that is installed on a Java EE server. For our purposes, we will assume that this is WAS. There are a few instructions for performing this task which are quite well documented in the InfoCenter.

A number of security resources are recommended:

#### Groups

- resAdministrators
- resDeployers
- resMonitors

#### Users

- resAdmin
- resDeployer
- resMonitor

### ***Creating an instance of Decision Server Events***

The Decision Server Events component requires a database to be configured for its use. This database must be configured prior to the creation or augmentation of profiles. For example, to create a DB2 database, first we create a DB called DSEVENTS.

With the database created, I open a DOS/DB2 command window. Next I change directory to <WODM Install>/config/db

I execute the command:

DB2 CONNECT TO DSEVENTS USING db2admin

and then

db2 -tvf db2.sql

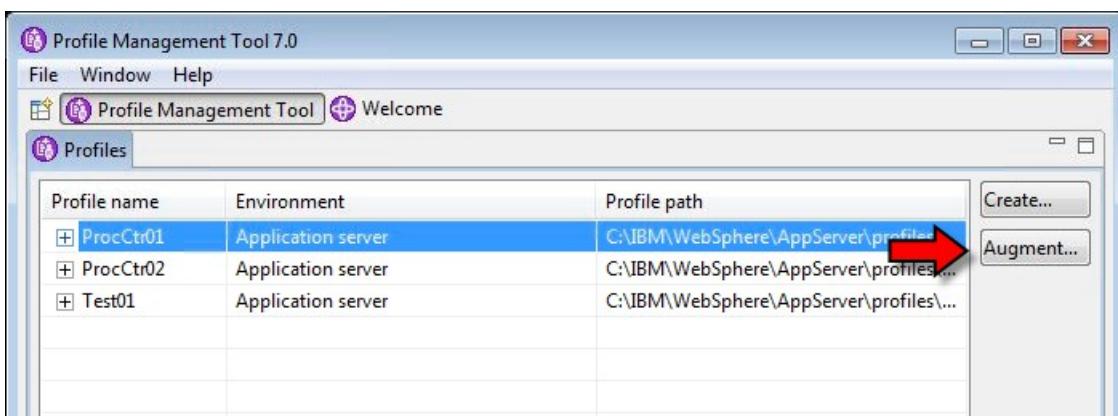
This creates the tables and other artifacts. No errors should be shown/listed. After completion a query of tables should show similar to the following:

Name	Schema	Table space	C
ACTIONS	DB2ADMIN	USERSPACE1	
ACTIONS_PREFS	DB2ADMIN	USERSPACE1	
AUTH_GROUPS	DB2ADMIN	USERSPACE1	
AUTH_USERS	DB2ADMIN	USERSPACE1	
BIZCAL_DAYS	DB2ADMIN	USERSPACE1	
BIZCAL_FACILITIES	DB2ADMIN	USERSPACE1	
CAPTURE_SETS	DB2ADMIN	USERSPACE1	
CAPTURED_EVENTS	DB2ADMIN	USERSPACE1	
CONN_PROPS	DB2ADMIN	USERSPACE1	
CSIO_ENTRY	DB2ADMIN	USERSPACE1	
HISTORY_ACTION	DB2ADMIN	USERSPACE1	
HISTORY_ENT_OBJ	DB2ADMIN	USERSPACE1	
HISTORY_ENT_OBJ_VALUES	DB2ADMIN	USERSPACE1	
HISTORY_EVENT	DB2ADMIN	USERSPACE1	
HISTORY_FILTER	DB2ADMIN	USERSPACE1	
HISTORY_RULE	DB2ADMIN	USERSPACE1	
HISTORY_WATCH_TIME	DB2ADMIN	USERSPACE1	
LABELS	DB2ADMIN	USERSPACE1	
MAESTRO_ASSET	DB2ADMIN	USERSPACE1	
PROJECT_ASSET	DB2ADMIN	USERSPACE1	
PROJECT_LABELS	DB2ADMIN	USERSPACE1	
PROPERTIES	DB2ADMIN	USERSPACE1	
STEPS	DB2ADMIN	USERSPACE1	
TIME_BASED_ASSET	DB2ADMIN	USERSPACE1	
USER_PROJECT	DB2ADMIN	USERSPACE1	

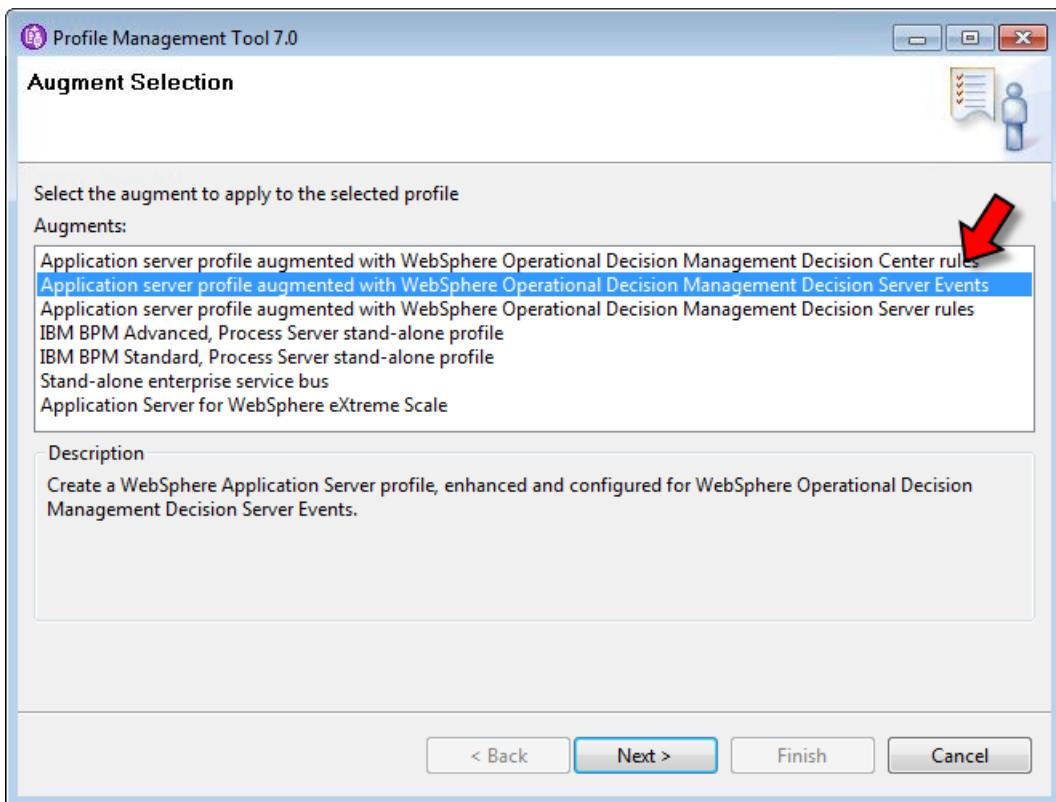
## Augmenting an existing App Server with Decision Server Events

If a WAS App Server is already defined and we wish to augment that server with Decision Server Events, we can do that through either the PMT tool or through the `manageprofiles` command. Here is a walk-through of using the PMT tool.

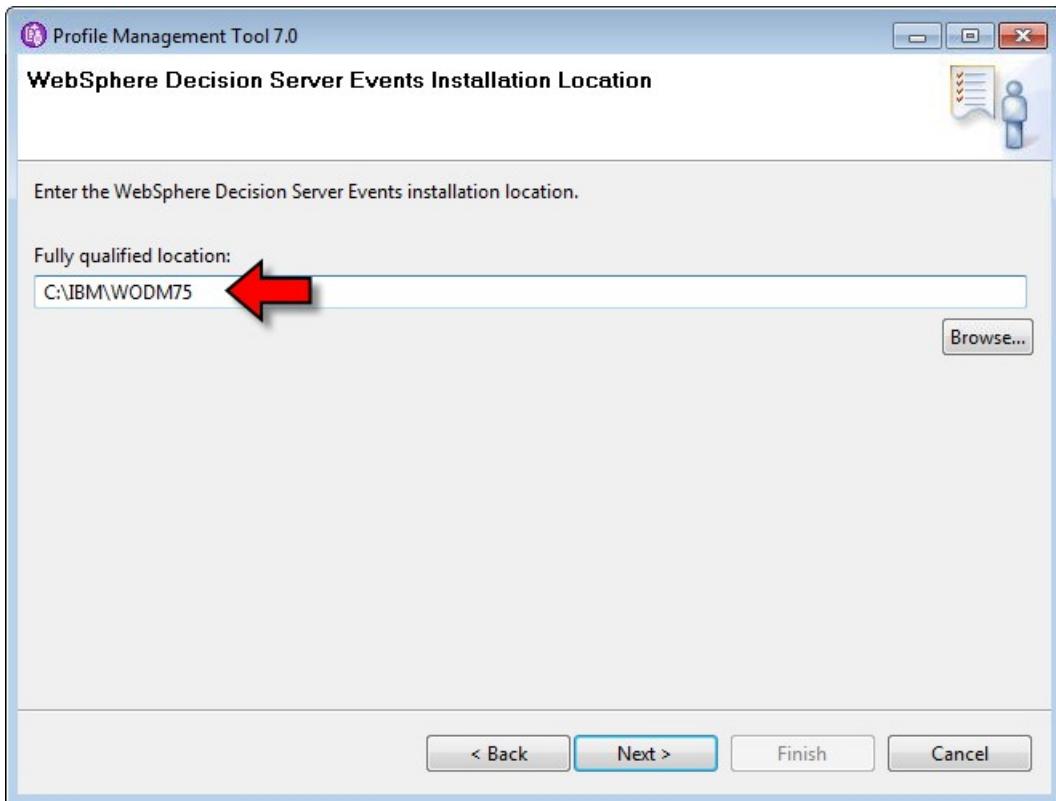
After launching the PMT tool, we select the profile we wish to augment. In this example it happened to be a Process Center server. Once the profile is selected, we click the Augment button to start the activity.



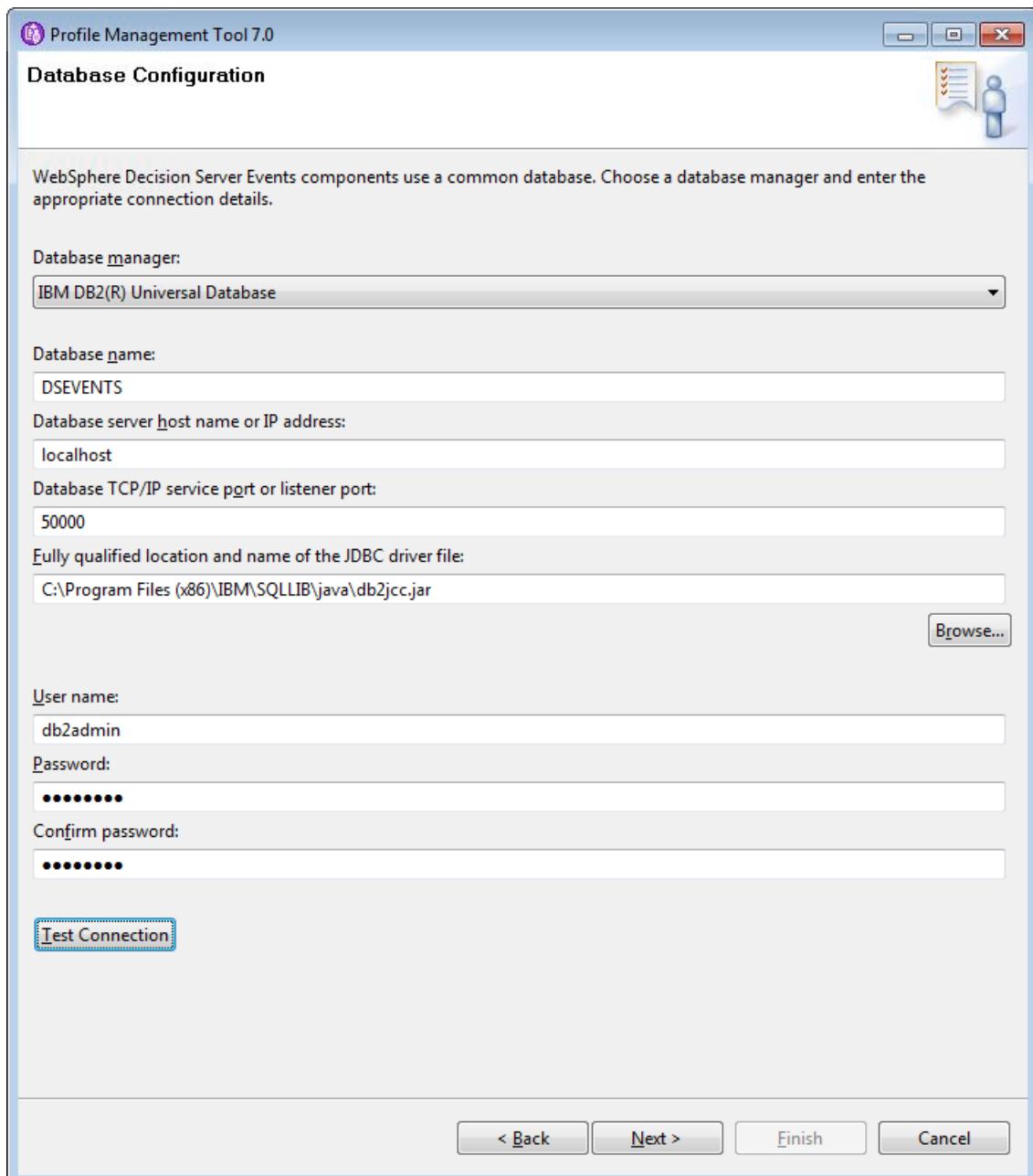
A profile can be augmented in a number of ways. To augment it to run Decision Services Events, we select the "Decision Server Events" entry.



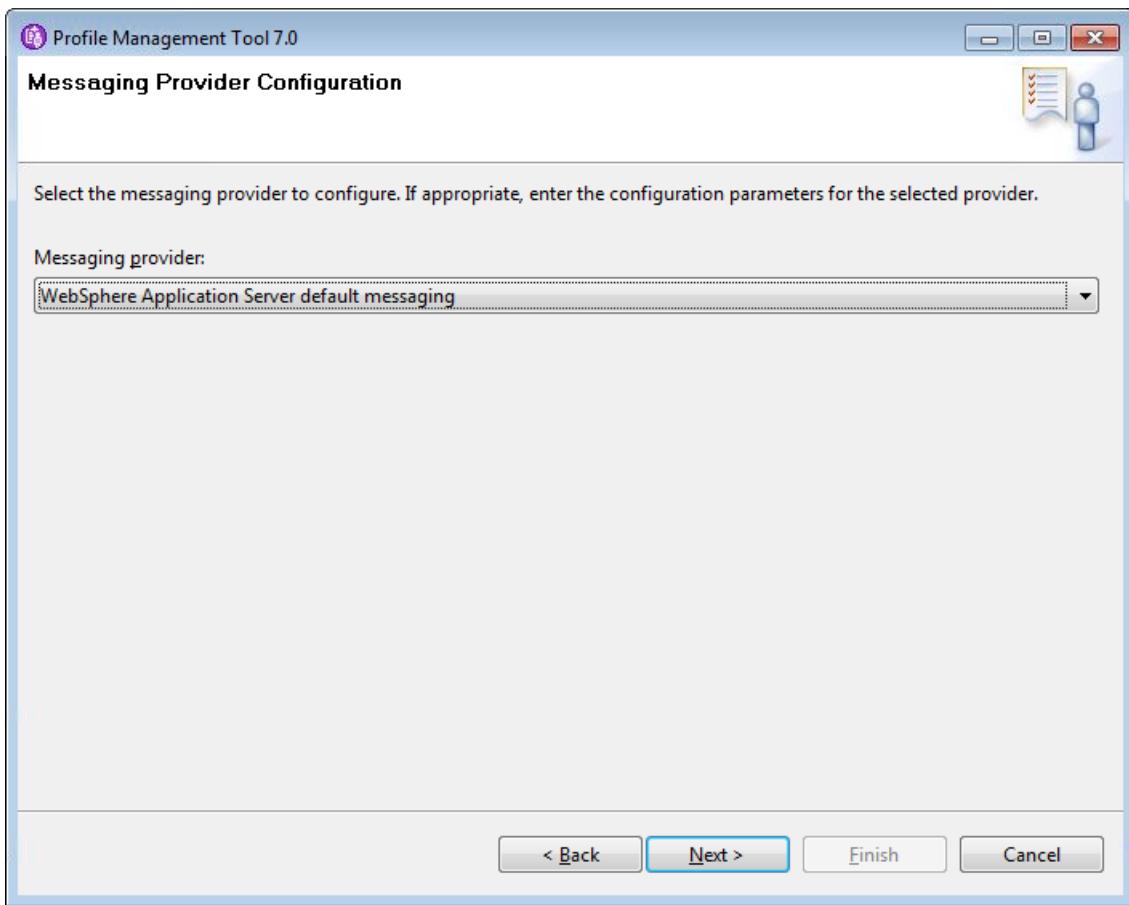
Next we are asked for the install location in which the code for Decision Server Events can be found. This is the directory into which WODM was installed.



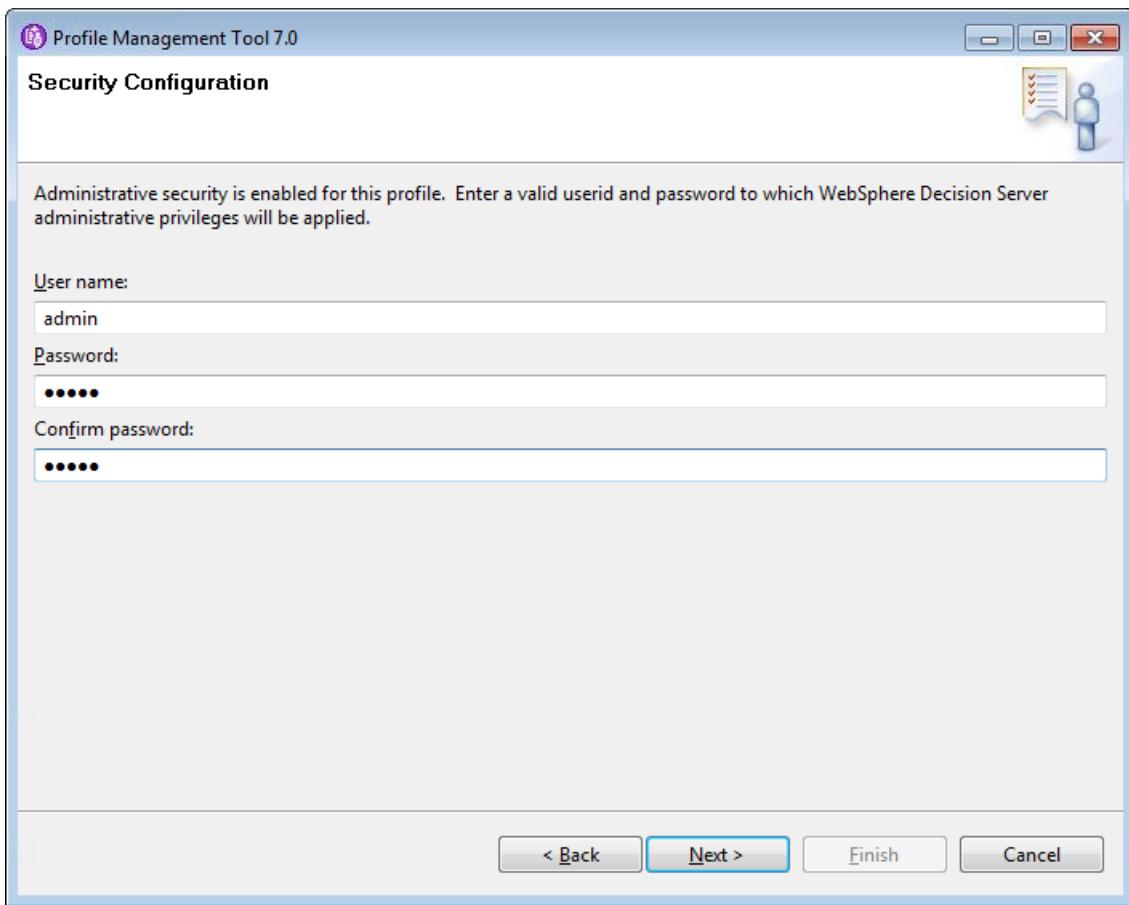
Decision Server Events uses a database at run-time. Next we define the database type and connection details we wish to use.



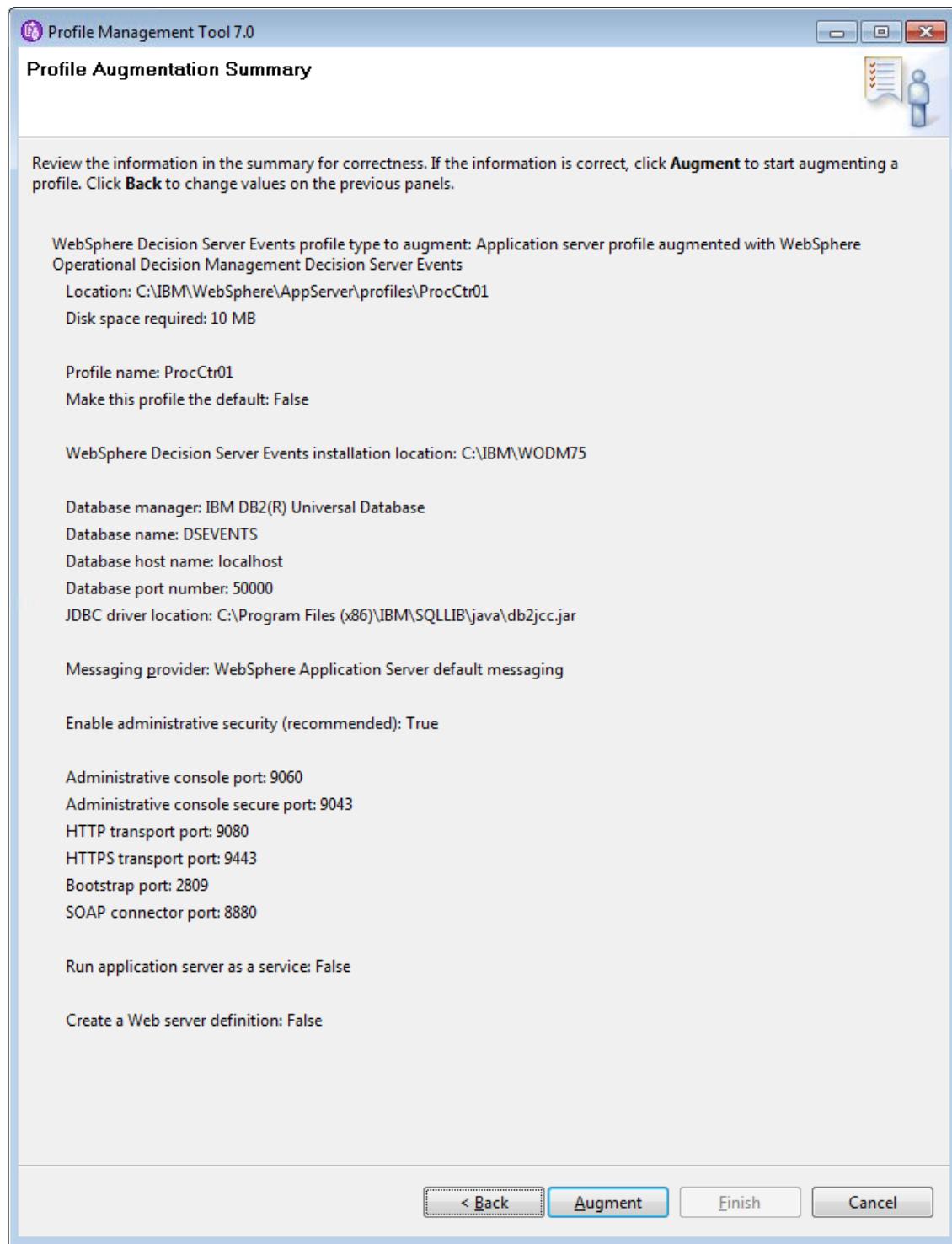
Decision Services Events also uses messaging for its operation. We can use a variety of messaging types including the default (SI Bus) or WebSphere MQ. Next we define which type of messaging we wish to employ.



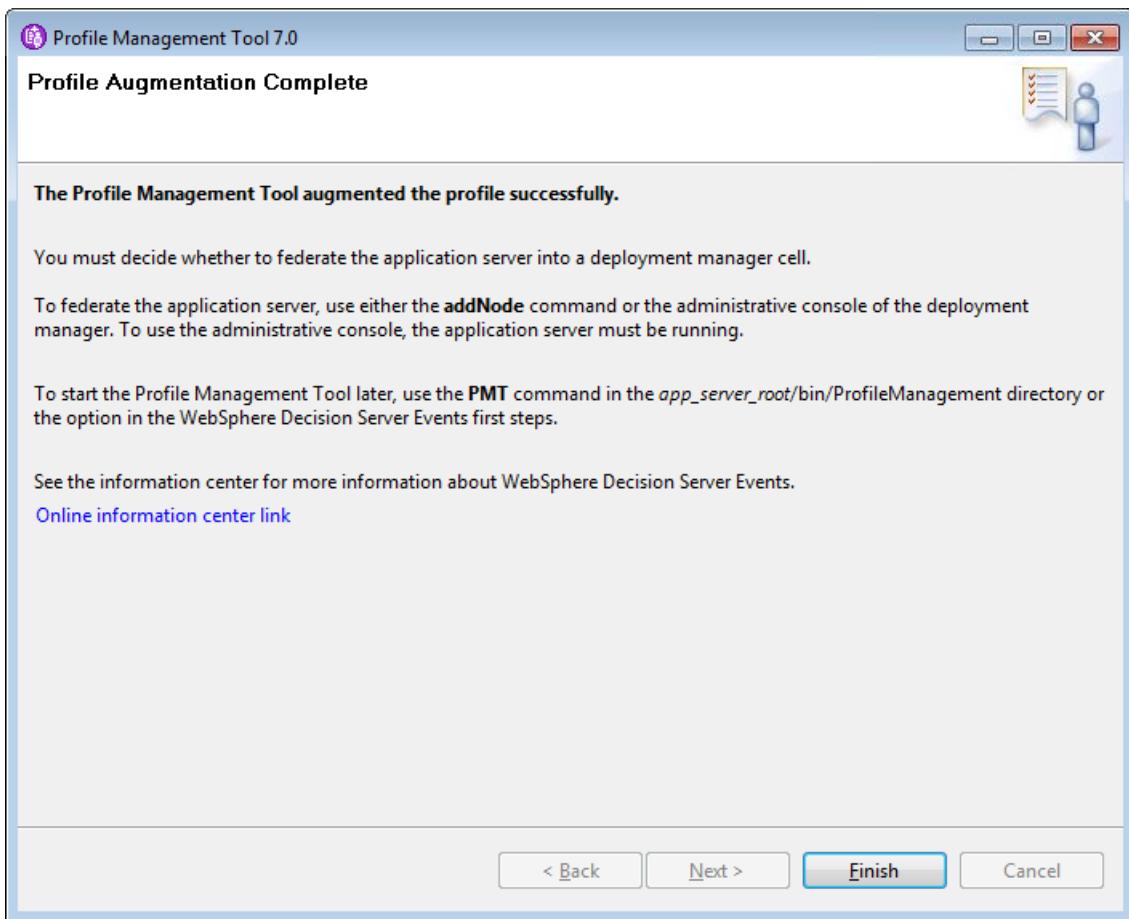
In order to perform certain configuration changes, the augmenter needs to know the WAS administration userid and password.



Finally, a summary of installation options is presented.



Augmentation now proceeds. If all has gone well, then at the conclusion we will see the following dialog:



The screenshot shows the "Profiles" view of the Profile Management Tool. The window has a toolbar with "File", "Window", and "Help" menus, and icons for "Profile Management Tool" and "Welcome". Below the toolbar is a navigation bar with "Profiles" selected. The main area displays a table of profiles:

Profile name	Environment
ProcCtr01	Application server
	Application server with Feature Pack for XML Version 1.0
	Application server with Feature Pack for SCA Version 1.0
	Application server with Feature Pack for SCA Version 1.0 with SDO 2.1.1
	IBM BPM Standard, Process Center stand-alone profile
	IBM BPM Advanced, Process Center stand-alone profile
	Application Server for WebSphere eXtreme Scale
	Application server profile augmented with WebSphere Operational Decision Management Decision Server Events
+ ProcCtr02	Application server
+ Test01	Application server

## WODM Documentation

### ***IBM Product Documentation***

The documentation for WODM can be found in the IBM [InfoCenter](#). The installation guide is a separate PDF found [here](#).

### ***Web Page Information Sources***

- [IBM Home page for JRules](#)

- [IBM Support page for JRules](#)
- [IBM JRules InfoCenter- v7.1](#)
- [IBM Internal – JRules Experts](#)

## ***DeveloperWorks***

- DeveloperWorks - [IBM technology in the financial markets front office, Part 2: Invoke conditional business rules based on analyzing data streams](#) – 2010-08-05
- DeveloperWorks - Integrating WebSphere Commerce with ILOG JRules for dynamic rules – 2010-08-04
- DeveloperWorks - [Integrate WebSphere ILOG JRules with IBM Content Manager Enterprise Edition](#) – 2010-05-13
- DeveloperWorks - [Building smarter event-driven business processes with the WebSphere BPM suite](#) – 2010-05-03
- DeveloperWorks - [Integrate IBM FileNet P8 BPM with IBM WebSphere ILOG JRules using Web services](#) – 2010-04-29
- DeveloperWorks - [Mapping rule-based decision services to rulesets](#) – 2010-03-24
- DeveloperWorks - [WebSphere ILOG rules for COBOL](#) – 2010-03-20
- DeveloperWorks - [Author and deploy rules using WebSphere ILOG JRules Business Rules Management System](#) – 2010-03-02
- DeveloperWorks - [Integrate ILOG JRules with WebSphere Process Server via asynchronous messaging](#) – 2010-03-01
- DeveloperWorks - [Test and simulate rules using WebSphere ILOG Business Rules Management System](#) – 2010-03-01
- DeveloperWorks - [IBM WebSphere ILOG Business Rule Management System applied in the financial industry](#) – 2010-02-18
- DeveloperWorks - [Overview of ILOG JRules and WebSphere Process Server integration](#) – 2010-02-10
- DeveloperWorks - [Synchronize WebSphere ILOG Rule Studio with Rule Team Server](#) – 2009-11-17
- DeveloperWorks - [Modify a decision table using WebSphere ILOG Rule Team Server](#) – 2009-11-17
- DeveloperWorks - [Synchronize WebSphere ILOG Rule Solutions for Office with Rule Team Server](#) – 2009-11-17
- DeveloperWorks - [Rule authoring using WebSphere ILOG Rule Solutions for Office](#) – 2009-11-17
- DeveloperWorks - [Modify a decision table using WebSphere ILOG Rule Solutions for Office](#) – 2009-11-17
- DeveloperWorks - [Use WebSphere ILOG Rule Team Server to query rules repository and work with rule history](#) – 2009-11-17
- DeveloperWorks - [Create a business object model using WebSphere ILOG JRules Rule Studio](#) – 2009-11-16
- DeveloperWorks - [Rule Authoring with WebSphere ILOG Rule Team Server](#) - 2009-11-16

## ***Redbooks***

- [Patterns: Integrating WebSphere ILOG JRules with IBM Software](#) - SG24-7881-00
- [Making Better Decisions using WebSphere Operational Decision Management](#) – REDP-4836
- [Proven Practices for Enhancing Performance: A Q & A for IBM WebSphere ILOG BRMS 7.1](#) - REDP-4755

## **WODM Architecture**

WODM is composed of the following components

### ***WebSphere Decision Center***

Decision Center is the management repository for artifacts to be stored. It is accessed via browser interfaces. It is designed for business users to work with.

### ***WebSphere Decision Server***

Decision Server is the run-time engine used to execute business rules.

This includes sub-components such as:

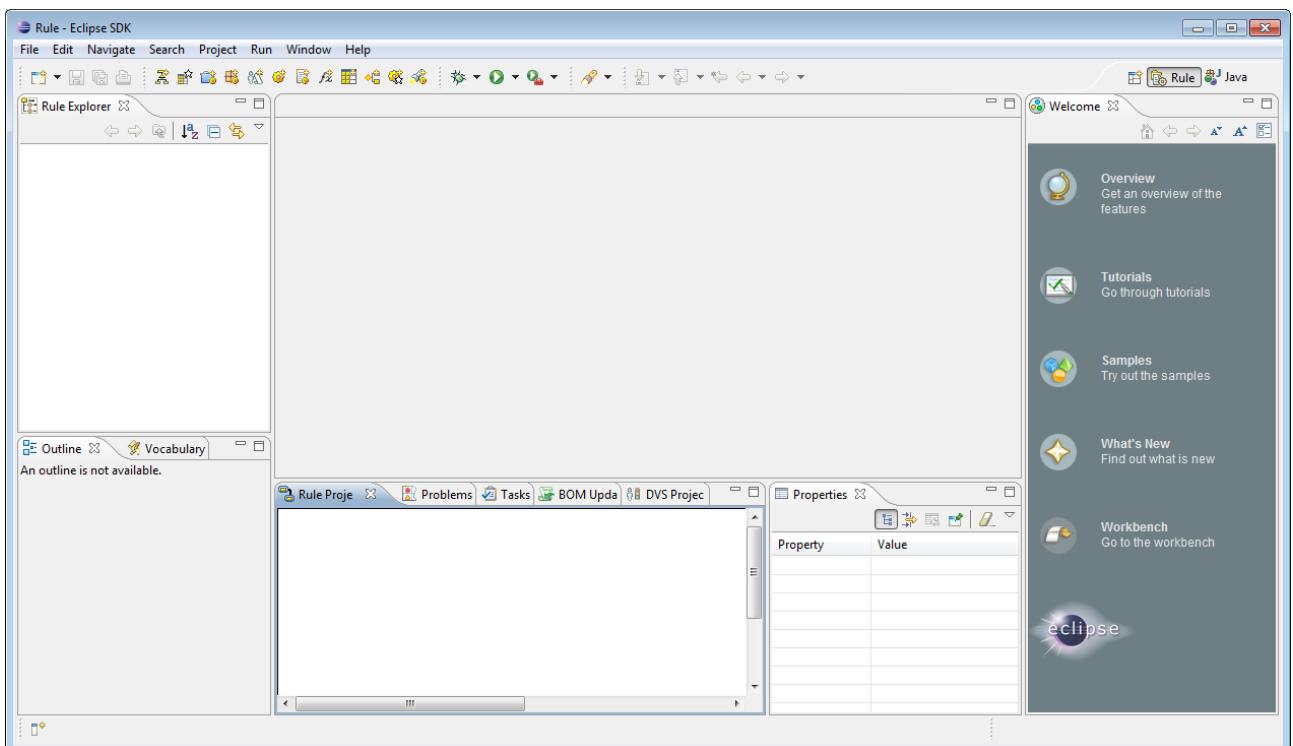
- WebSphere Decision Server Event run-time

## WODM Terms

- BAL – Business Action Language
- DVS – Decision Validation Service
- IRL – ILOG Rule Language
- Verbalization – Creating a vocabulary that business users can use to edit the rules

## Rule Designer

Rule Designer is the development component of WODM. It is built on the classic Eclipse framework.

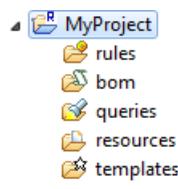


Once launched, Rule Studio exhibits the default Eclipse icon in the system tray:



## Project Structure

When a new Rules Project is created, container folders are automatically added. The project itself is decorated with an "R" for "Rules".

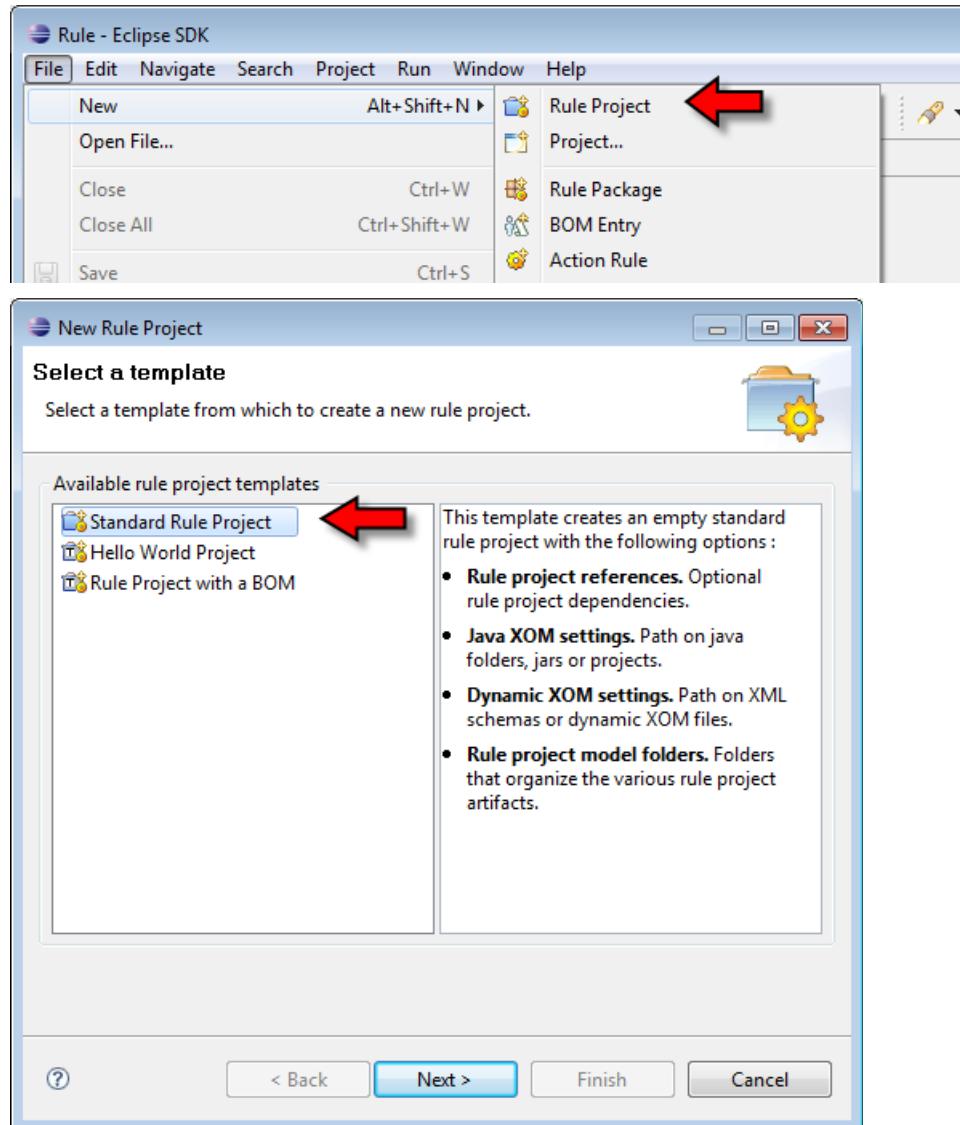


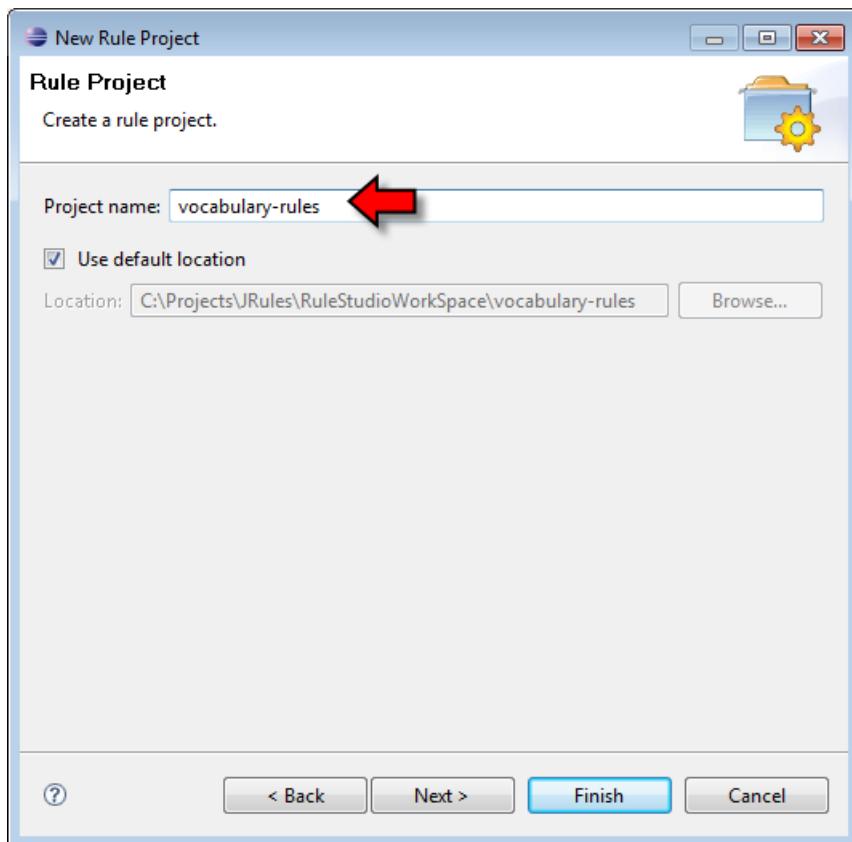
## ***Learning WODM***

The InfoCenter documentation contains a sample tutorial.

## Rule Projects

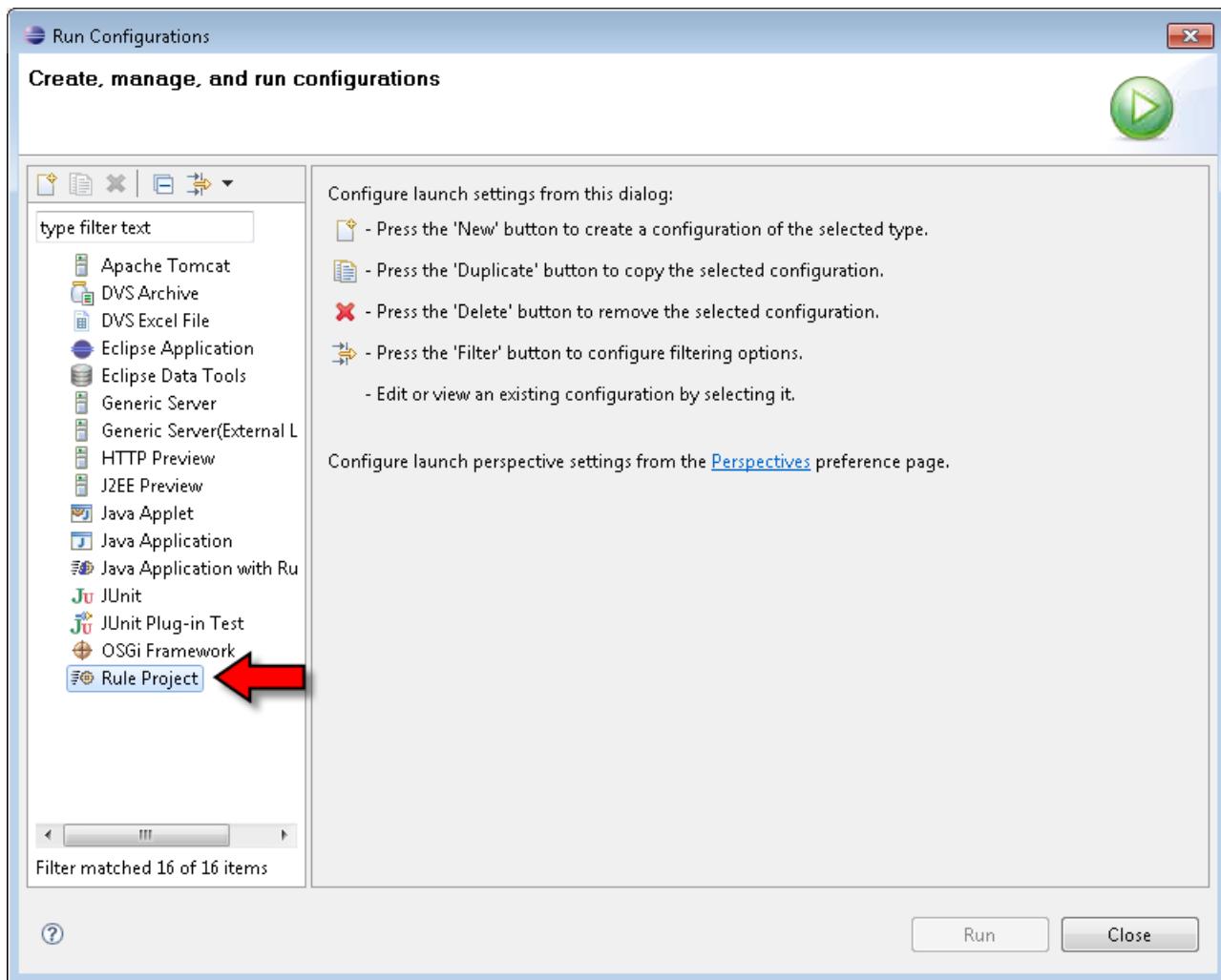
A Rule Project is a project created within Rule Studio to hold/house the artifacts for the solution.





## ***Running a test execution***

In the Eclipse Run Configuration section, we have the ability to define a Rule Project

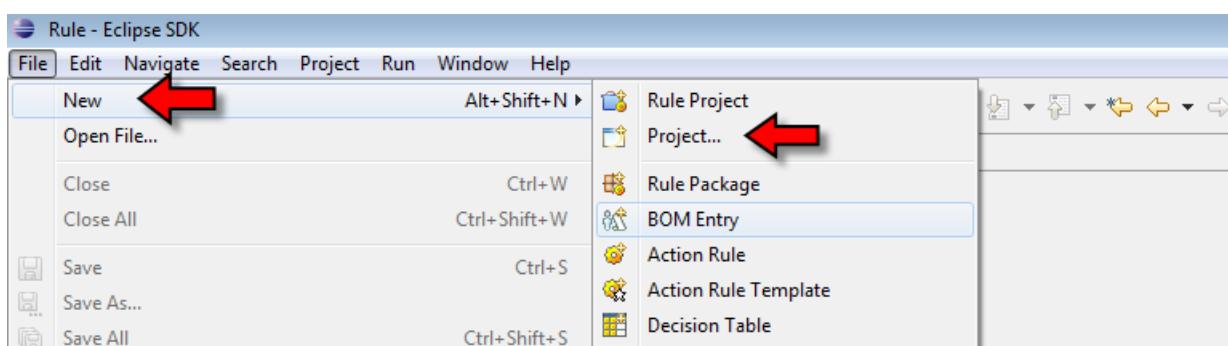


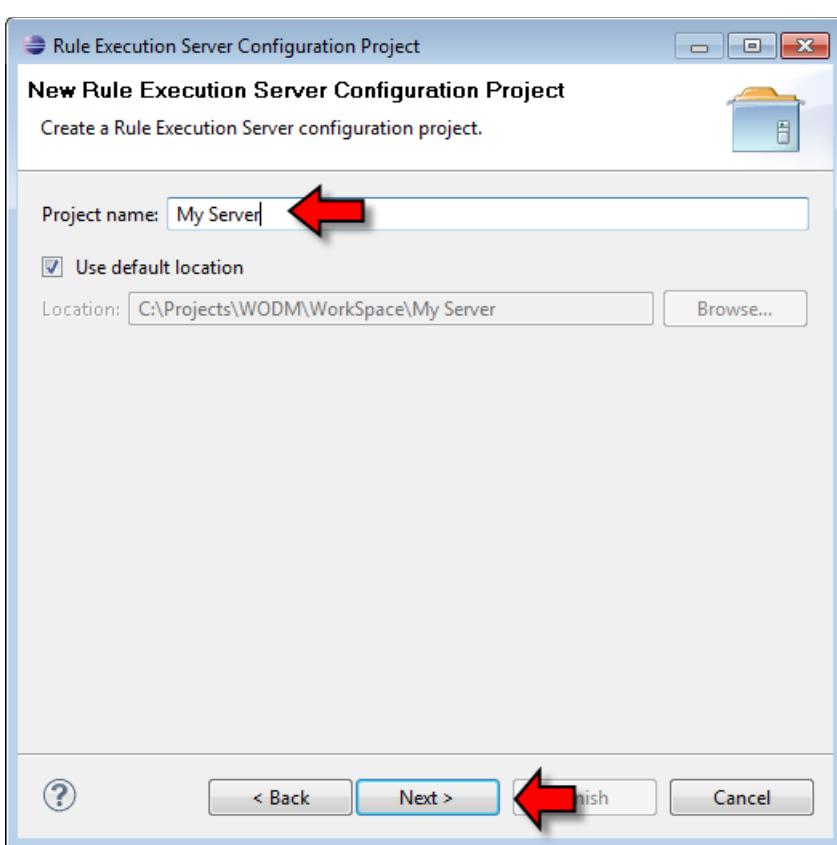
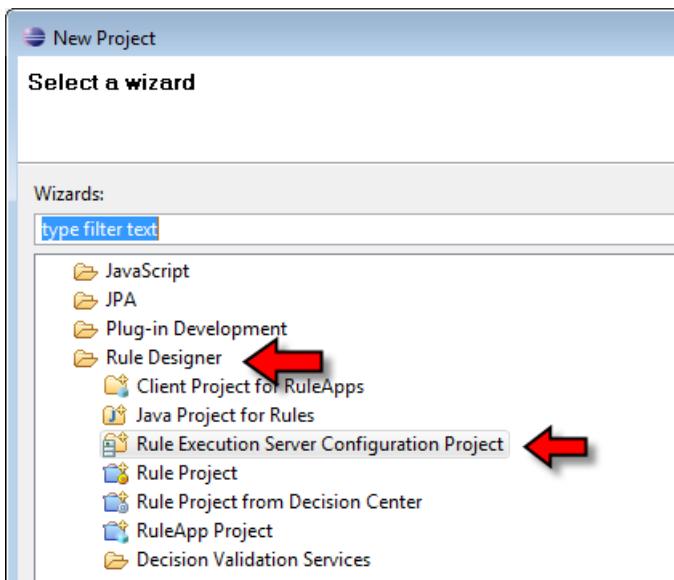
## Integrating with WPS

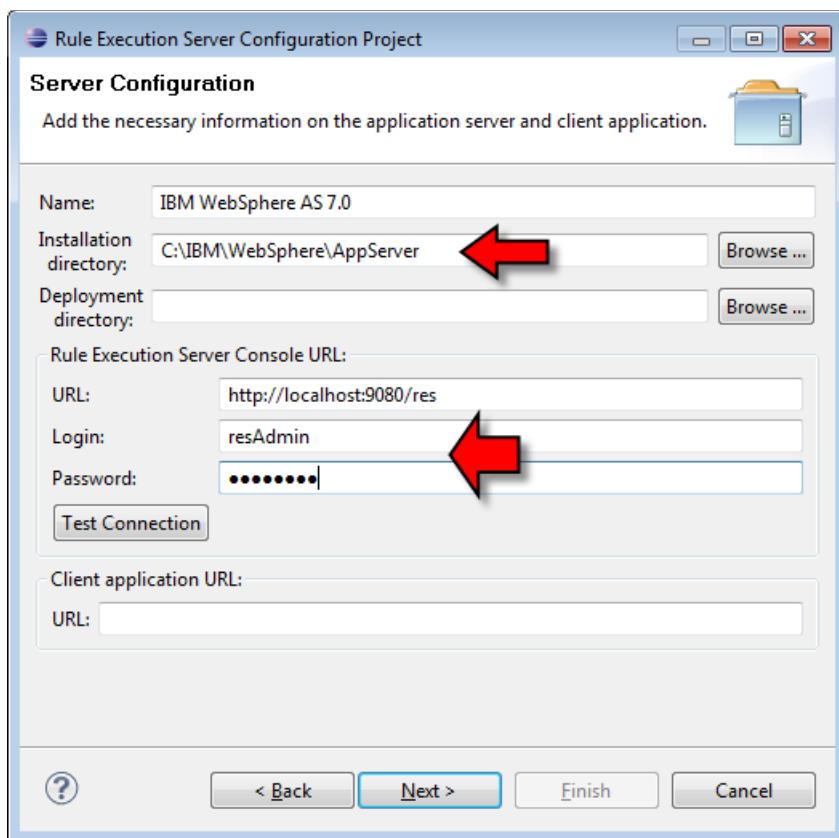
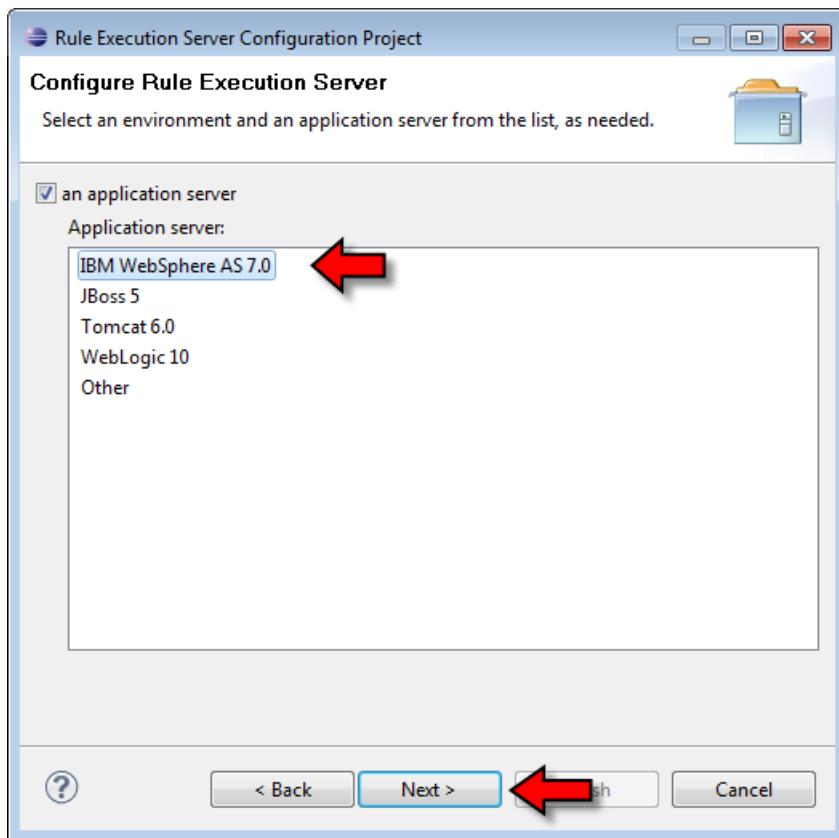
<http://www-01.ibm.com/support/docview.wss?uid=swg27017588>

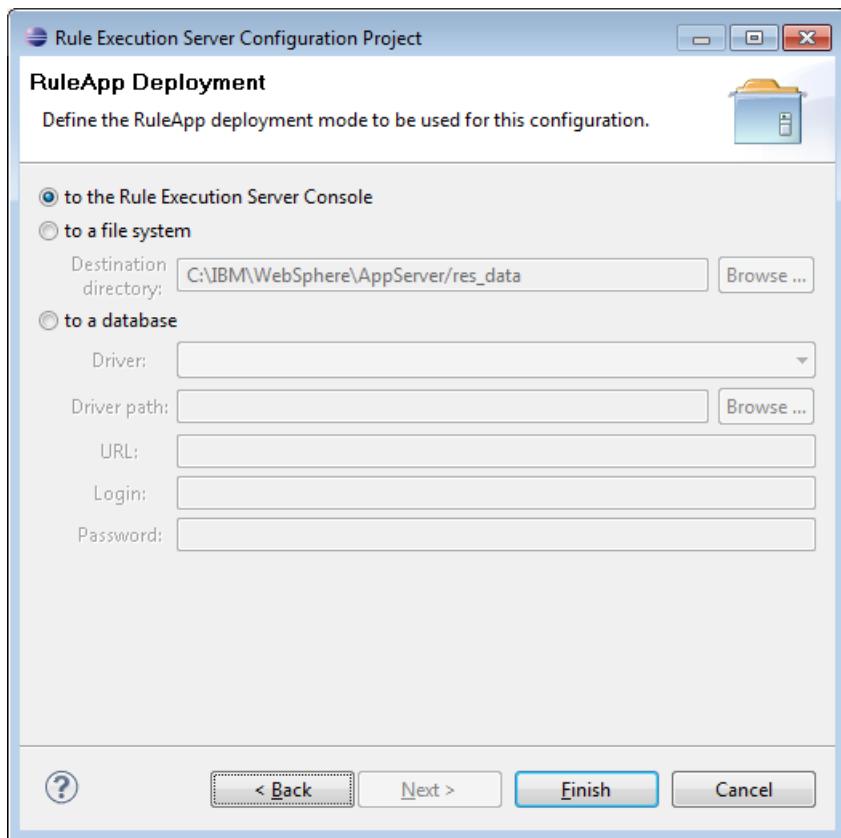
### **Defining a Rule Execution Server configuration**

When a Decision Server is configured, we may want to deploy an application directly to it. One of the easier ways to achieve this is to first create a server definition which can then be named and referred to when we wish to deploy.









**IBM WebSphere AS 7.0**

### Rule Execution Server Configuration: IBM WebSphere AS 7.0

**Environment**

an application server  
 Rule Execution Server Console Deployed

**Server Information**

Server: IBM WebSphere AS 7.0

Installation directory: C:/IBM/WebSphere/AppServer

Deployment directory: [Browse ...](#)

**Rule Execution Server Console**

URL: http://localhost:9080/res  
 Login: resAdmin  
 Password: \*\*\*\*\*

You can:  
[Open](#) the Rule Execution Server Console.  
[Test](#) the configuration connection.

**Client Application**

URL: [Open your client application.](#)

**Deployment**

You can:  
[Deploy](#) one or more RuleApp(s) to this Rule Execution Server.

**RuleApp Deployment**

The RuleApp deployment used for this configuration is :

to the Rule Execution Server Console  
 to a file system

Destination directory: [Browse ...](#)

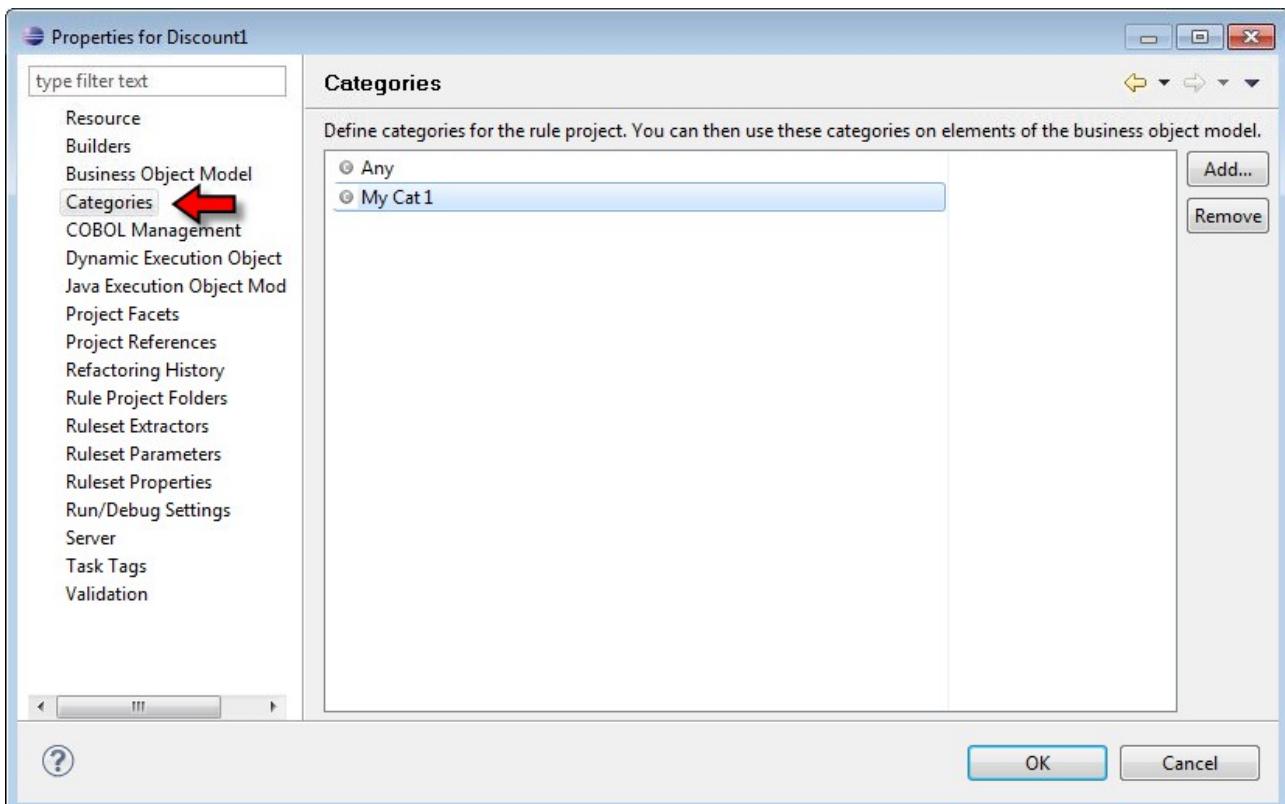
**to a database**

Driver: [Browse ...](#)  
 Driver path: [Browse ...](#)  
 URL: [Browse ...](#)  
 Login: [Browse ...](#)  
 Password: [Browse ...](#)

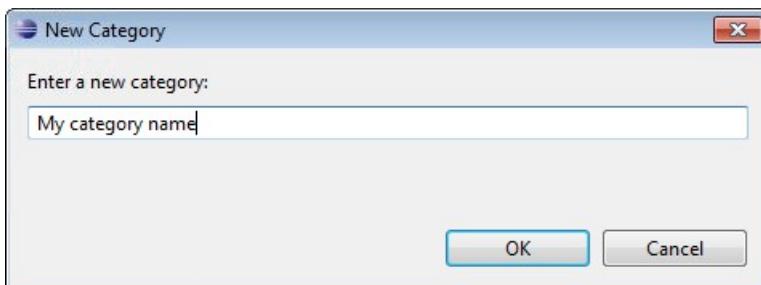
[Overview](#) IBM WebSphere AS 7.0.esc

## Categories

Categories can be defined by opening the properties of the Rule Set project.



New categories can be added by pressing the Add button after which a dialog appears into which the name of the new category to be created may be entered.



## The Vocabulary

Consider the idea of rejecting a Loan. In Java, a Loan class may be coded as:

```
class Loan
{
 ...
 public double amount;
 ...
}
```

if we want to reject a loan because the amount is too high, in Java we would code:

```
Loan myLoan = new Loan();
...
if (myLoan.amount > 100000)
{
 // Reject the loan
}
```

This is all good for a programmer, but our goal with BRM is to be able to express rules at a higher

level allowing easier creation and customization.

Instead, we might want to code:

```
if
 the amount of the loan is more than 100000
then
 reject the loan
```

Examining this for a few minutes seems to show that there is a "mapping" between the language we use and the data available. For example the English 'the loan' is a place holder for an instance of the Java class "loan". And 'the amount' is a place holder for a field in the Loan Java class. What we need to do is create what JRules calls a "Vocabulary".

The Vocabulary is how the rules are verbalized. These are the terms that can be used to construct language rules. In Rule Studio, the vocabulary of an entry is defined in the BOM entry for an object. In addition, Rule Studio provides a Vocabulary View that can be used to see the language for an associated type:

The screenshot shows the 'Vocabulary for vocabulary-rules' window in Rule Studio. The left pane displays a tree structure under the 'customer' node. The 'customer' node has several child nodes, each starting with '@{a customer}'. Some of these nodes have orange dots next to them, indicating they are expanded. One node, '@ make it {a boolean} that {a customer} is loyalty program member', has a blue arrow pointing to it from the right side. The right pane contains a list of terms, with the first one being 'stock'. There are also icons for 'Edit', 'Delete', and 'Generate automatic variable'.

- ▷ **System**
- ▷ **Vocabulary**
- ▷ **customer**
  - ① @ {a customer} is loyalty program member
  - ① @ {the birth date} of {a customer}
  - ① @ {the birth day of month} of {a customer}
  - ① @ {the birth month} of {a customer}
  - ① @ {the birth year} of {a customer}
  - ① @ {the first name} of {a customer}
  - ① @ {the last name} of {a customer}
  - @ make it {a boolean} that {a customer} is loyalty program member
- ▷ **rental agreement**

## ***The Business Term***

The idea of the Business Term is the "class" or "container" for further data. Examples of a "Business Term" include a "loan" or a "customer". The Business Term is edited in the "Class Verbalization" section of the BOM model.

The screenshot shows the 'Class Verbalization' section for the 'stock' business term. It includes options to remove or edit documentation, generate automatic variables, and edit the term itself. A navigation phrase 'the stock, a stock, the stocks....' is listed below the term input field.

**Class Verbalization**

Remove the verbalization.       Edit the documentation.

Generate automatic variable

Term:

**i** the stock, a stock, the stocks....

The idea of a Navigation phrase allows us to delve into the content of the business term item.

**Member Verbalization**

- [Remove](#) the verbalization.
- [Create](#) a navigation phrase.
- [Create](#) an action phrase.
- [Edit](#) the subject used in phrases.

**Action : "set the name of a stock to a string"** ✖

Template:  ✖

**Navigation : "the name of a stock"** ✖

Template:  ✖

If the data type of the attribute is a boolean, then the phrase is termed a predicate phrase.

An Action phrase is an action that can be applied to the object such as a setter on the data or an execution of a piece of Java code.

When we look at a phrase, we see that the phrase is made up of two kinds of "things". These things are either *placeholders* or *text*. For example, in the following phrase:

set the name of {this} to {name}

The elements I have marked in blue are text while those in red are placeholders.

Placeholders are marked by being surrounded in curly braces. When building rules, the placeholders are replaced with instances of information.

A special placeholder called {this} refers to the current instance of the object.

## Categories

## Functions

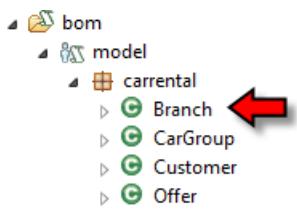
## Execution Object Model (XOM)

The Execution Object Model (XOM) is the description of the physical data supplied to the rules. The XOM is mapped to the Business Object Model (BOM) and then the rules executed against the BOM. The XOM can be described either through Java classes or through XSD.

## Business Object Model (BOM)

The Business Object Model or BOM is the core of WODM. It allows us to describe data and the vocabulary associated with that data to allow business rule editing to be user-friendly. Think of a BOM entry similarly to a Java Class.

A BOM entry has some core attributes. These include a "Name" which is the unique name of the BOM data. A BOM's name is further qualified by a "Name-space". The combination of name-space and name must be unique in the environment. When a BOM entry is created in Rule Studio, it shows up in the BOM folder as a Class icon:



When a BOM is opened in Rule Studio, it is opened in the BOM editor:

The BOM editor has a sequence of category tabs (shown at the bottom) which group together aspects of a single BOM.

## BOM Class category

The Class category contains various sections that describe the BOM class as a whole. These include:

General Information

Class Verbalization

Members

Domain

Categories

Custom Properties

BOM to XOM Mapping

### ***BOM Member category***

The Member category contains various section that pertain to a single member of the BOM. These include:

General Information

Member Verbalization

Arguments

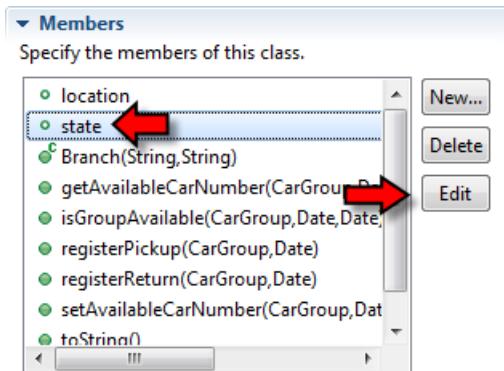
Domain

Categories

Custom Properties

BOM to XOM Mapping

The details of an individual member of a BOM can be edited by selecting a Member in the Class category and clicking the Edit button:



Once done, the BOM editor changes to the Member category page with the member selected for editing:

## BOM Data types

The data type of a member can be defined as an arbitrary Java type but care should be taken if the project is to be integrated with other systems.

The concept of a domain is a constraint on the value of a given member. Think of the potential values being taken from a "domain" or "set" of such values. Potential domains include:

- Literals – a named set of possible and fixed values
  - Bounded – Upper and lower bounds for values
  - Collection ....
  - Static References ....

- Others

### ***BOM implementation files***

Inside Rules Designer, a single BOM definition is physically stored as three distinct files. These are:

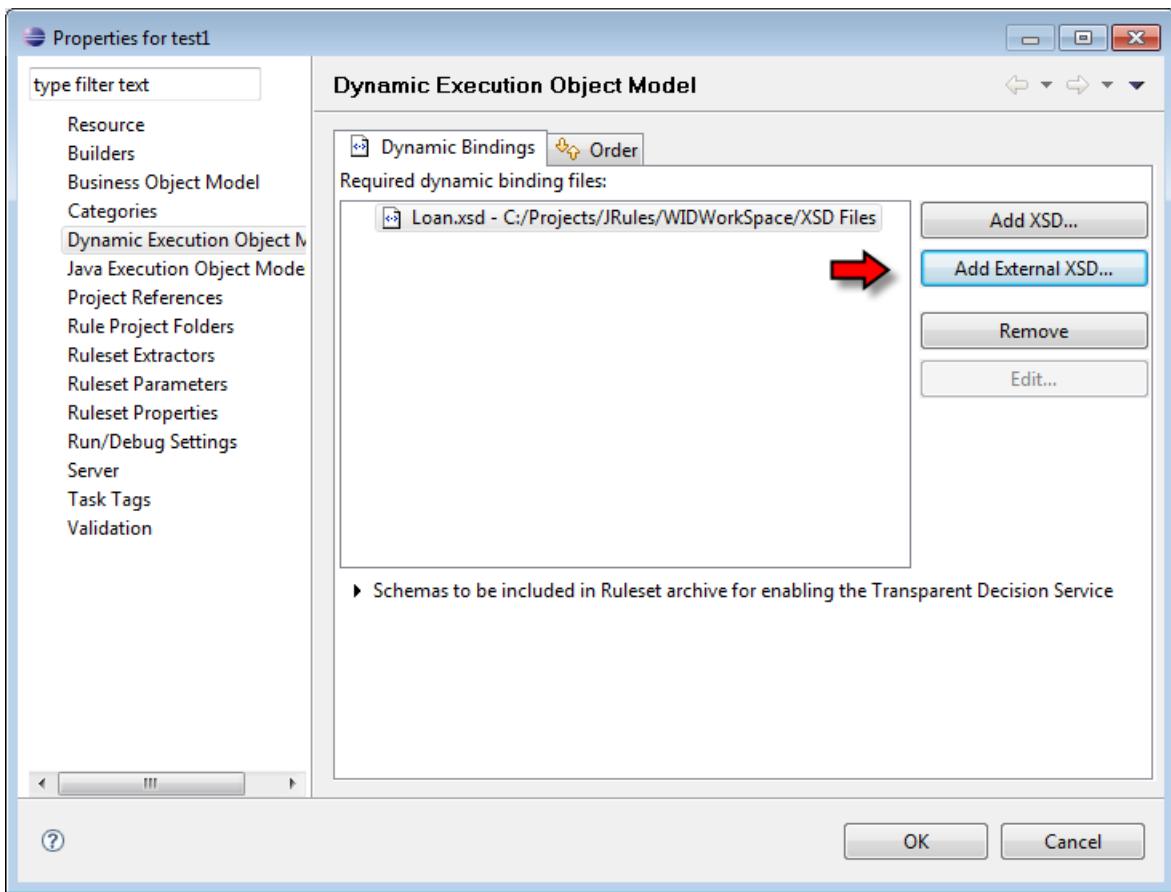
- The BOM file which describes the content and structure of the BOM being defined
- One or more VOC files that define the vocabulary used by the BOM. There will be one file for each language in which the vocabulary is described
- The B2X file which describes the mapping to and from a BOM to a XOM

## **Working with XSDs**

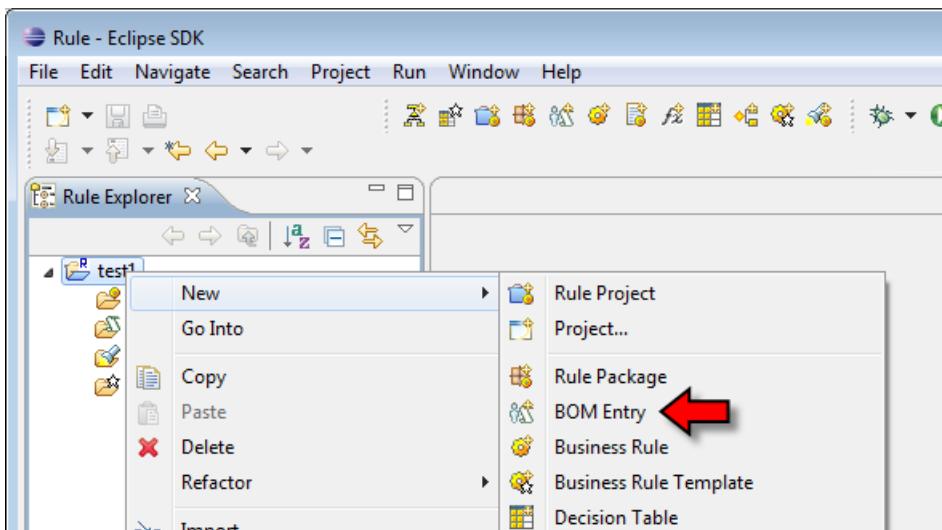
Assuming that we have an XSD file that describes some data, it must be made known to the rules project before a BOM can be created from it.

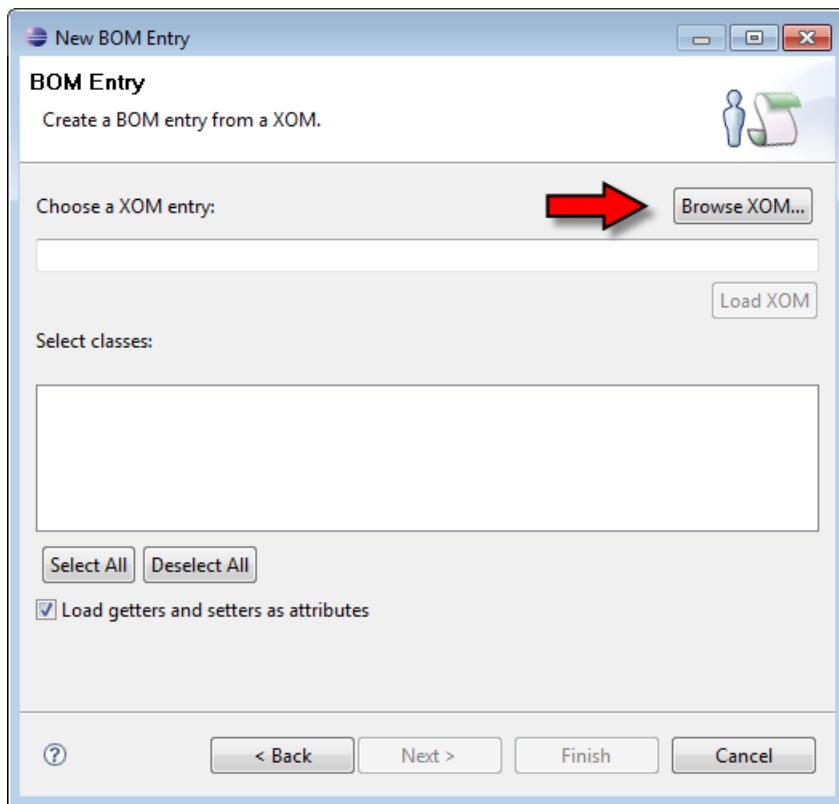
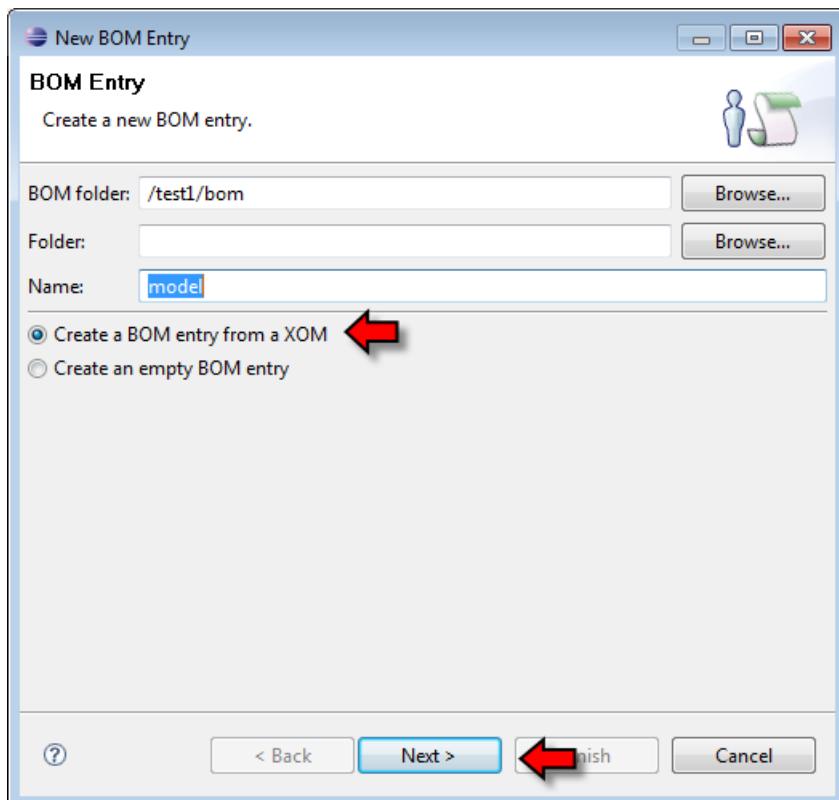
From the project properties, we can select the Dynamic Execution Object Model and add an XSD that is part of a project or add an external XSD. Imagine we have an XSD that defines a simple Loan

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://MyBank/Loan/"
 xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://MyBank/Loan/">
 <complexType name="Loan">
 <sequence>
 <element name="name" type="string" minOccurs="1"></element>
 <element name="amount" type="double" minOccurs="1"></element>
 <element name="age" type="int" minOccurs="1"></element>
 </sequence>
 </complexType>
</schema>
```

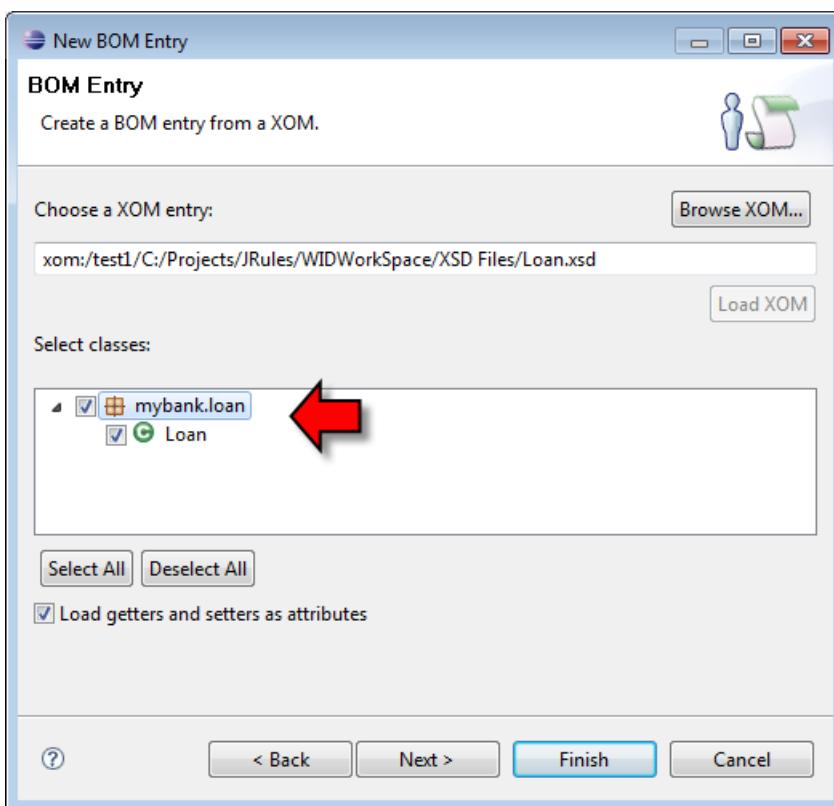
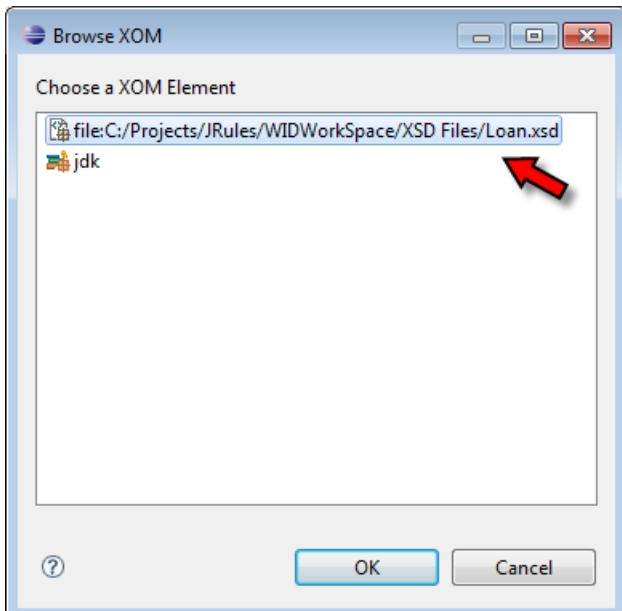


Once this has been done, we can then create a BOM from the XSD.





Because we previously defined the XSD to the project, it now becomes a choice in the known XOM entries.



At the conclusion of this step, we now have a BOM created that was based on the definition of the XSD.

# Rule Sets

A Rule Set can be thought of as an executable that runs under the context of a rule engine. A Rule Set is a collection of one or more rules. Before deployment, the rule set is turned into a rule set archive.

At deployment time, one more concept comes into play namely that of a RuleApp. A RuleApp is a collection of one or more Rule Sets.

A Rule Set contains the following:

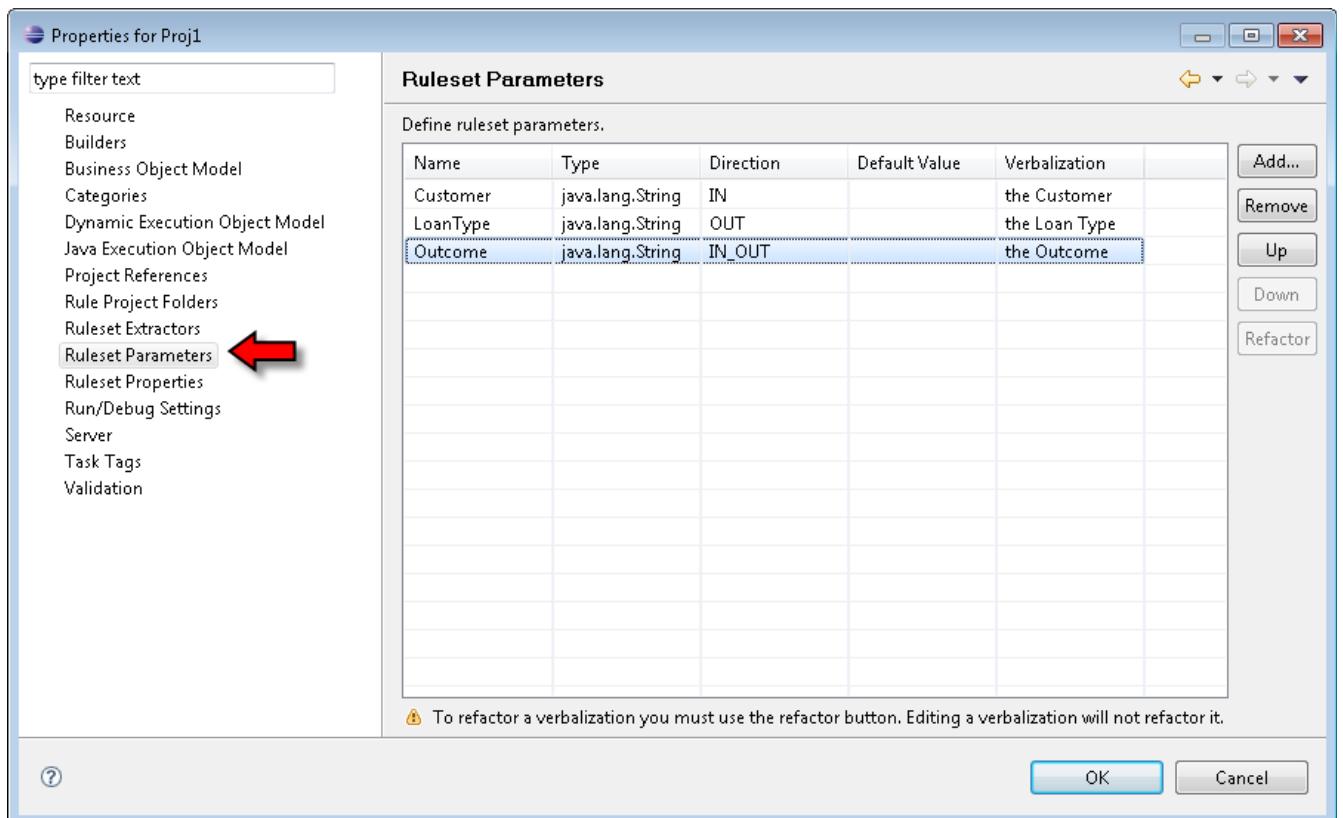
- Action Rules
- Decision Tables
- Decision Trees
- Technical Rules
- RuleFlows
- Technical Functions
- Variable Sets
- Rule Projects

## ***Rule Set Parameters***

A Rule Set has associated with it the concept of parameters. It is the parameters of a Rule Set that describe its "signature" to the outside world. Each parameter of a ruleset has the following associated with it:

- name
- type
- direction
  - IN
  - OUT
  - IN\_OUT

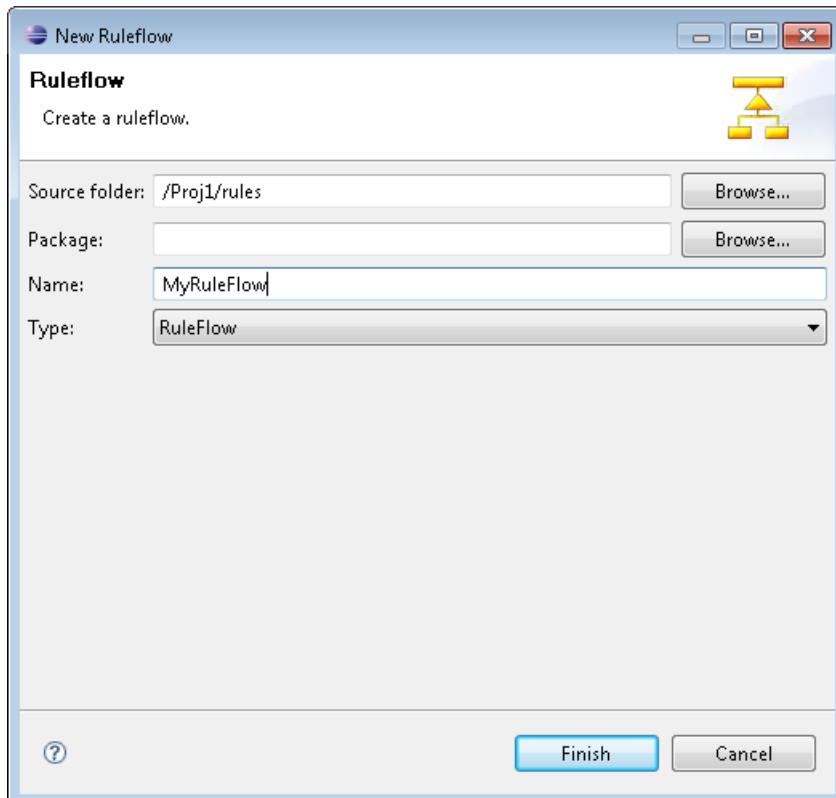
Rule Set parameters are created for a Rule Set by selecting the properties for the enclosing project.



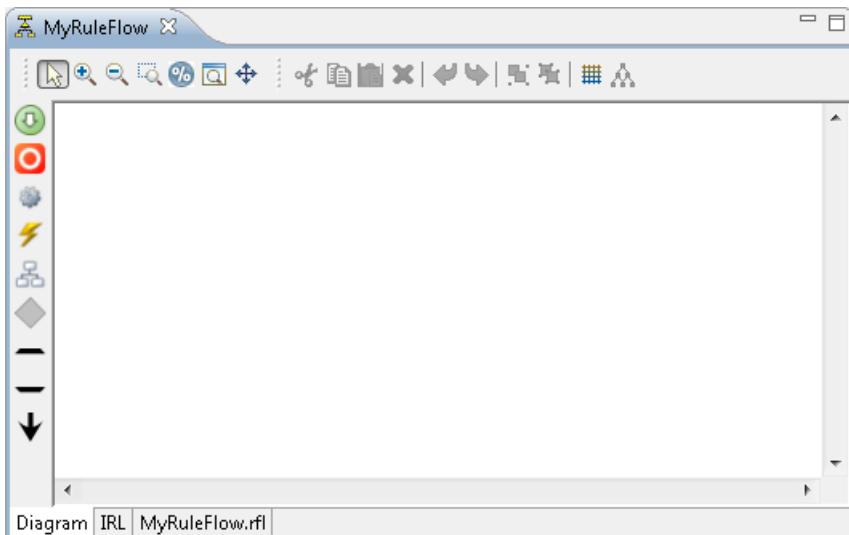
## Rule Flow

A Rule Flow is a sequence of rules that include sequencing of execution.

### Creating a Rule Flow



## Rule Flow editor



	Start node
	End node
	Rule Task
	Action Task

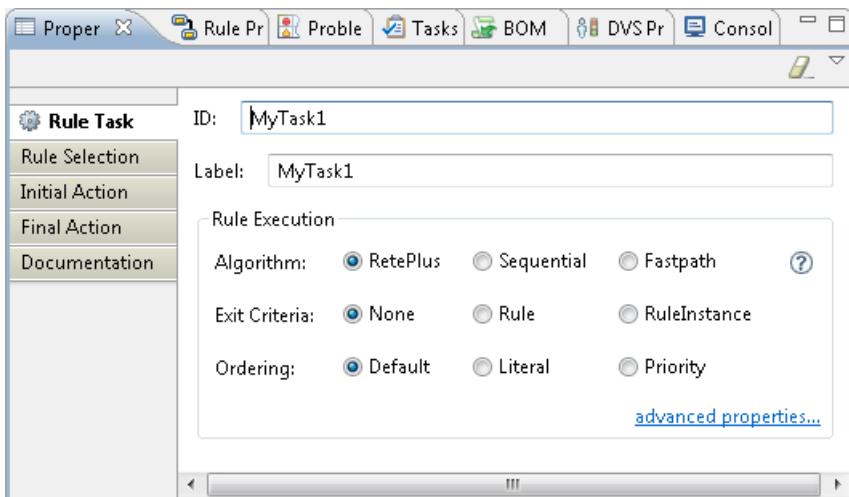
When a Rule Flow has been created, it can be found in the project under the rules folder as shown below:



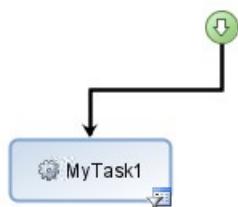
The icon for a Rule Flow is a "flowchart" like tree.

Rule Tasks are the core work horses for a Rule Flow.

When a Rule Task is selected in the Rule Flow diagram editor, the Properties view changes to show the details of the task.

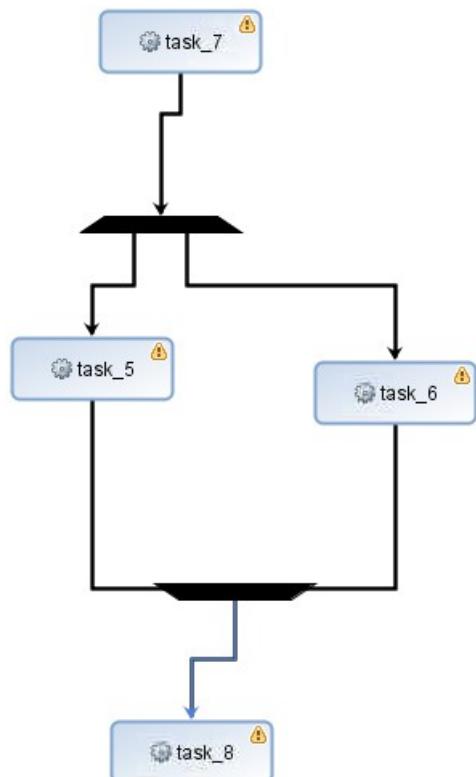


Transitions are the explicit wires that link the rule nodes together. A transition is directional meaning that it starts at one node and ends at another.

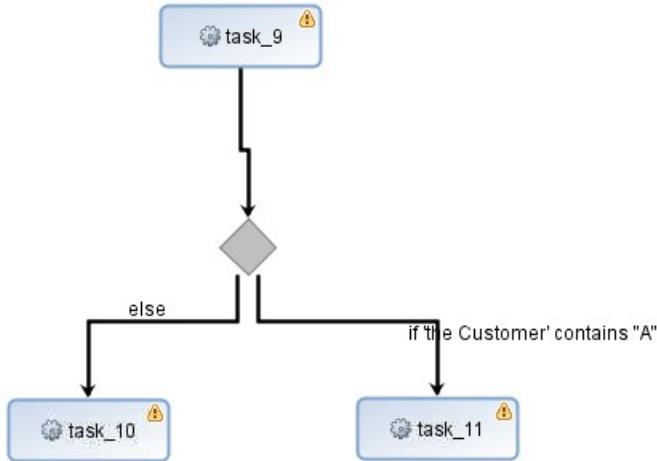


A transition can have a true/false expression associated with it. The transition will only be followed in the expression condition evaluates to true during execution.

We can create fork and join actions to have tasks executed in parallel and then later resynchronized.



As an alternative to using explicit transition expressions, a choice or gateway activity may be used:



Initial actions are BAL or IRL that are executed before an activity is executed. Initial actions can be applied to:

- Start nodes
- Rule tasks
- Action tasks
- Subflow tasks

Similarly, final actions are BAL or IRL that are executed upon completion of an activity. Final actions can be applied to:

- End nodes
- Rules tasks
- Action tasks
- Subflow tasks

## Business Policies

Think of a Business Policy as a decision that affects the operation of the business. These policies can be expressed as:

- Business Rules
- Decision Tables
- Decision Trees
- Technical Rules

## Business Action Language

The Business Action Language (BAL) is the language used to describe rules within WODM. Think of it as a "near" English representation of logic or decisions that have to be made. Structurally, it

consists of four separate parts:

- definitions
- if
- then
- else

The first part is the optional "definitions" while the following three parts are the "conditions". Both are discussed following.

## **BAL Definitions**

In the BAL definitions section, you define variables that will be used in the rule. A definition is made with the keyword "definitions" followed by assignments to variables. Variables are named entities that can be set to either numbers, text or existing business terms. An individual definition can be guarded with a condition expression using the format "where"

i.e.

set <variableName> to <value> where <expression>;

A variable can be defined to a list of values using the "all" option:

set <variableName> to all <values> where <expression>;

The variableName can contain whitespace if the name is bracketed with single quotes.

## **BAL Conditions**

Within the BAL code we can define "conditions". Conditions are simply "if ... then ... else ..." constructs such that the "if" part defines an expression and only when the expression evaluates to true do we execute the "then" part. If the expression was false, then we execute an optional "else" part.

The screenshot shows the 'Business Rule: Rule1' window. The 'General Information' tab is selected, displaying 'Name : Rule1' and 'Categories: Any'. The 'Category Filter' tab is also visible. The 'Documentation' tab is present. The 'Code' tab is selected, showing the following BAL code:

```
if
 the status of passenger is "GOLD"
then
 print "The passenger is gold!" ;
```

At the bottom, there are tabs for 'Intellirule', 'IRL', and 'Rule1.brl'.

The condition expression can use variables provided in the BAL definition section or as parameters on the rule project itself.

The expression is built using a set of available building blocks. These are grouped into three types called:

- comparison
- existence
- set membership

Contains / does not contain	Test for string containment within another string.
Ends with / does not end with	Test for string ending with another string.
Equals / does not equal	
Is / is not	Test that one value equals another
Is between <number> and <number>	Test for a number in a range.
Is empty / is not empty	Test for the string being empty.
Is more than / is at least	Test for a number being more than another
Is less than / is at most	Test for a number being less than another
Starts with / does not start with	Test for string starting with another string.

There is at least one	
There is no	
The number of	
There are	
There are at least	
There are at most	
There is one	

Is one of	
Is not one of	
Contain	
Do not contain	

Multiple conditions can be linked together with the AND and OR constructs.

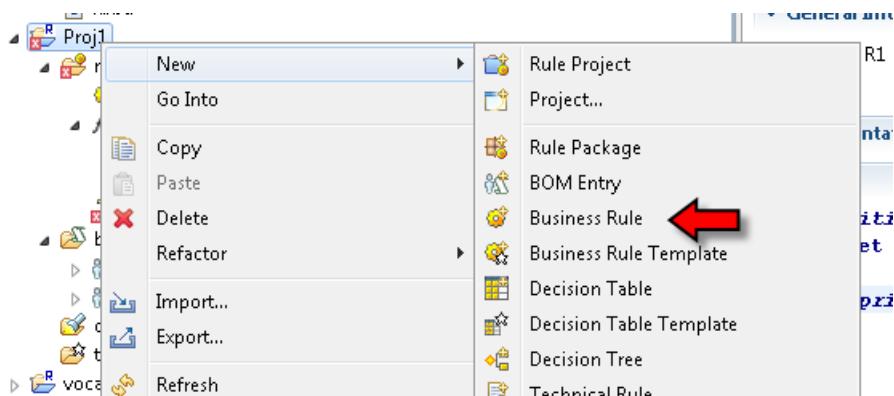
### **Rule Actions**

Following the then part of the rule is the Rule Action section. A rule action is executed if the condition evaluates to true. The action can either change the value of an object or execute a method/function.

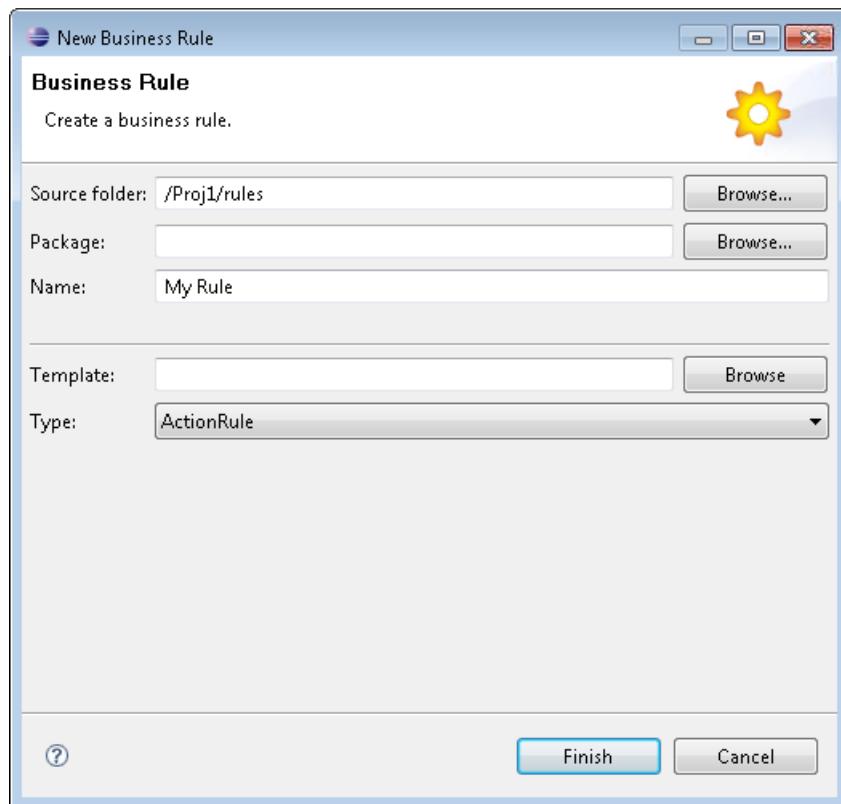
An optional else part can be used to define a rule action that is executed if the condition expression evaluates to false.

# Business Rules

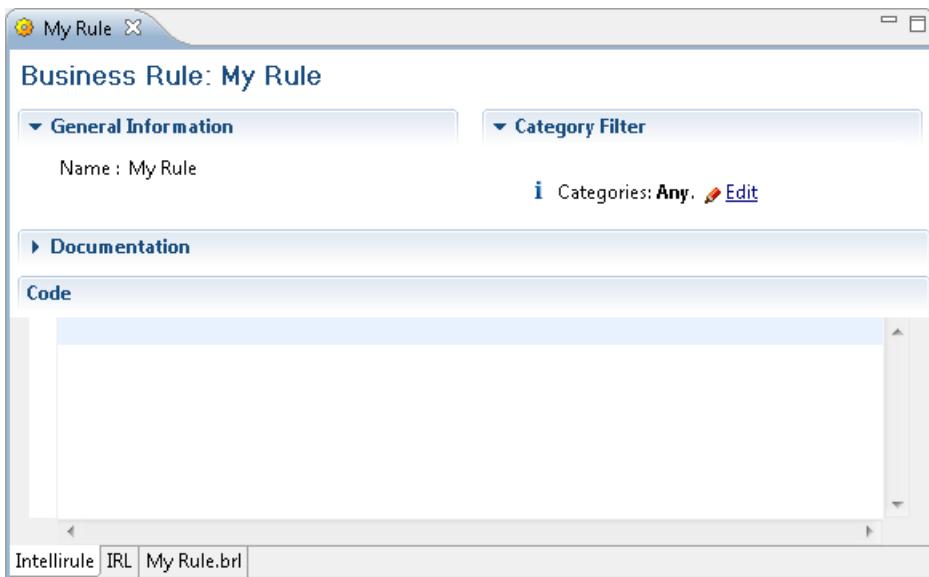
A rule is added to a project from the add context menu on that project:



A rule creation panel is shown which allows the name of the new rule to be supplied:



The rule editor provides the place where the details of the rule can be edited.



The section called "Code" is where the rule details can be entered. This is added as BAL.

## Decision Tables

We have already seen that decisions can be described as rules in the BAL language that read very much like English. Expressing decisions in this rule language is very straightforward but there are times when this may not be desired. Consider a business rule that wishes to capture the following requirement:

"When customers order goods, we wish to give them an incentive to order more at one time. To that end, we will offer discounts based on the value of an order. If they order more than \$500, we will give them a 10% discount. If they order more than \$1000 we will give them a 15% discount and if they order more than \$5000 we will give them a 20% discount".

We could achieve this effect with three rules that might look as follows:

```

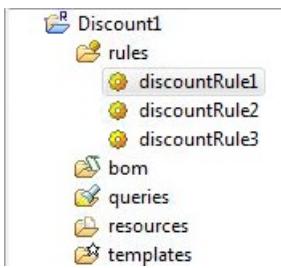
Content
if 'the value of the order' is between 500 and 1000
then
set discount to 0.1;

Content
if 'the value of the order' is between 1000 and 5000
then
set discount to 0.15;

Content
if 'the value of the order' is at least 5000
then
set discount to 0.2;

```

Unfortunately, this isn't as clear as it could be. In this example the logic is spread over three rules and to get the whole picture would take some work on our part as we opened each rule and related it to each of the others.



Now let us consider an alternative mechanism for capturing the rules that is called a Decision Table. A decision table that captures our desired rule may look as follows:

Table1

Value of the order		Discount	
	at least	less than	
1	\$0	\$500	0%
2	\$500	\$1,000	10%
3	\$1,000	\$5,000	15%
4	\$5,000	\$9,999,999	20%

Hopefully a quick look at this immediately shows how much more intuitive it is to describe our rules in this structure.

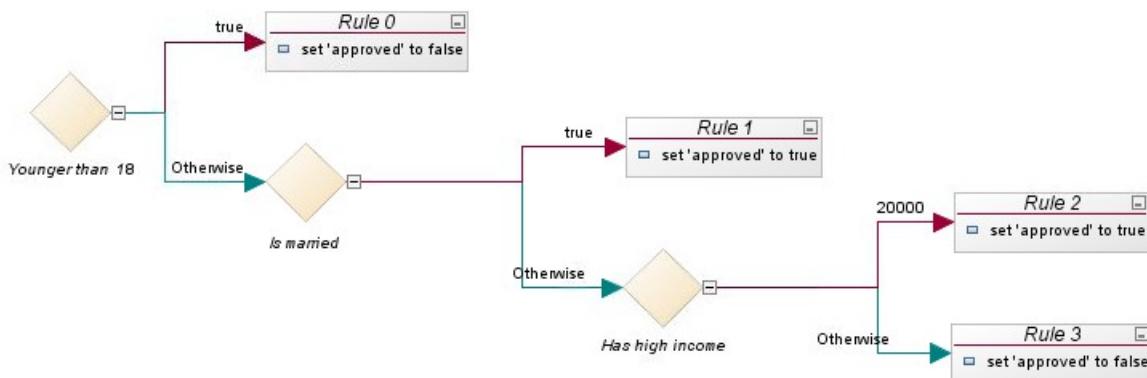
## Decision Trees

We have seen two previous decision description capabilities. We have seen decision rules as an English language style description mechanism and decision tables as a tabular style and both serve their purposes. WODM provides a third style of decision description called a decision tree.

Consider a decision that has a requirement expressed as:

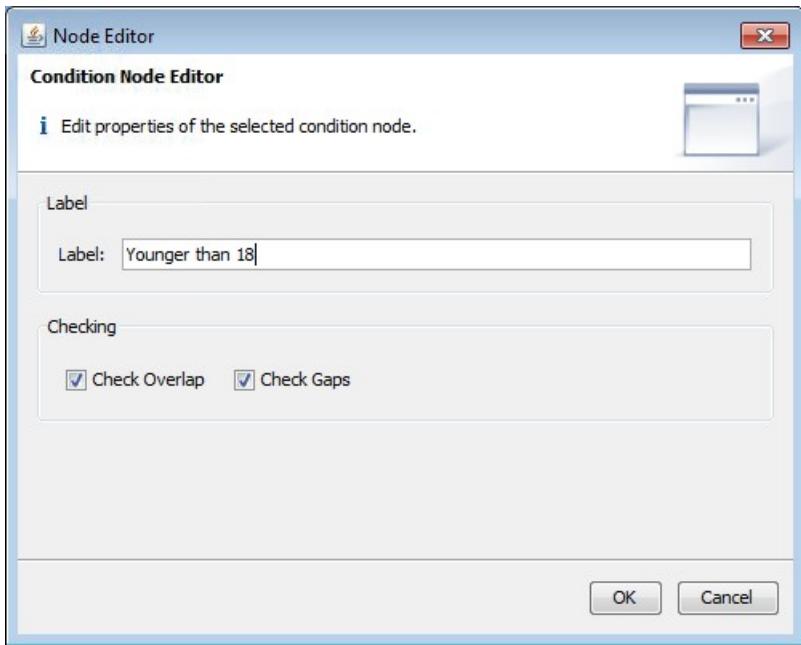
"We are considering whether to approve a loan. If the client is under the age of 18, we will deny. If the client is single, they will be approved only if their income is over \$20,000."

Using the decision tree style of describing rules we can model this as follows:

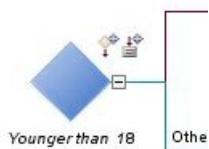


The diamonds in the diagram represent decision expressions. The boxes represent actions to be performed. The links coming out of decisions represent potential values of the expression.

In the nomenclature of the product, the decision diamonds are called "condition nodes". Double clicking a condition node opens a dialog into which the label can be entered. The label appears beside the condition node in the diagram:



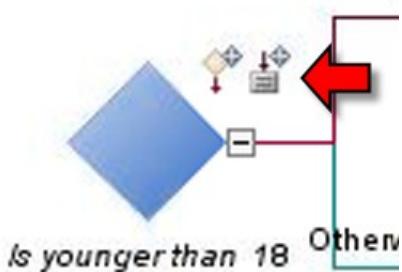
To set the expression associated with a condition node, click that node. Its color will change to blue to show that it is selected:



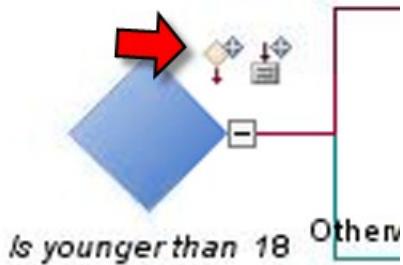
At the top of the editor, an expression can be entered to reflect the decision being asked:



When a condition node is selected, branches can be added. This can be achieved through the icon that looks like:



This icon can be found on both the diagram and in the menu bar of the editor. An additional condition node can be added to the left of the current condition or action by clicking on the other icon:

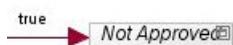


Take care when deleting condition nodes as all their following branches will also be deleted.

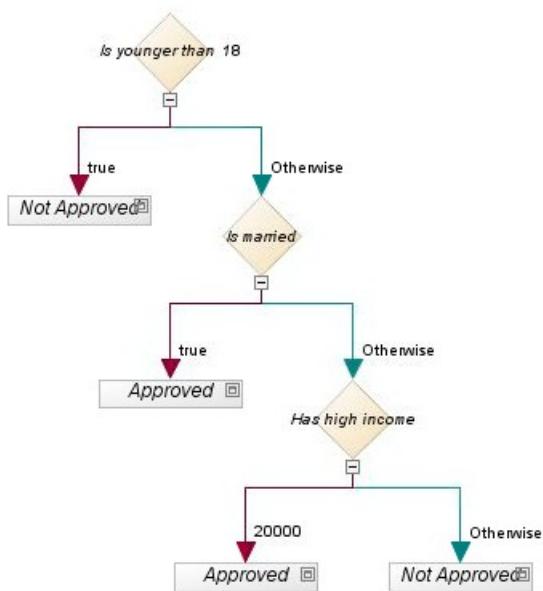
Actions can be shown expanded to show their content:



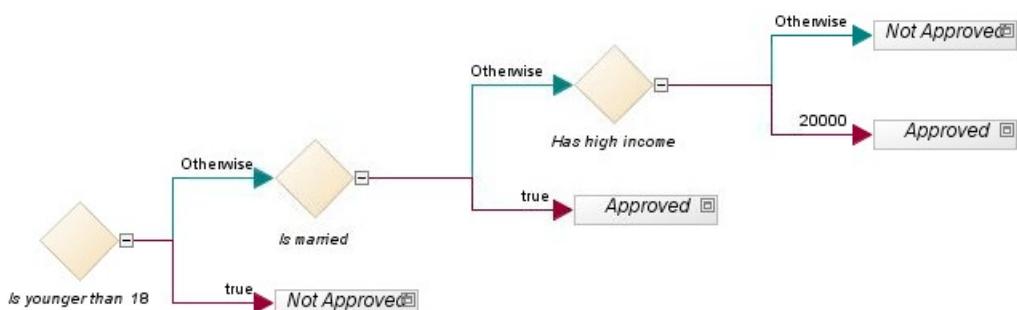
or collapsed to just show their labels:



The decision tree diagram can be oriented in two ways, either vertically:



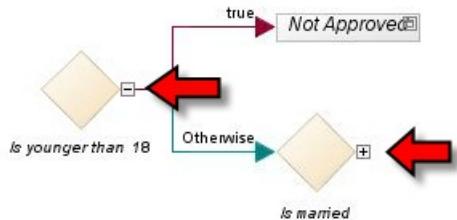
or horizontally



the choice of orientation can be made from the menu in the editor:

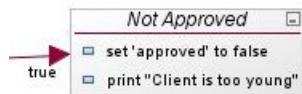


The branches following a condition node can be "collapsed" meaning that they are hidden from view. This can be achieved by clicking on the box associated with the outgoing branches:

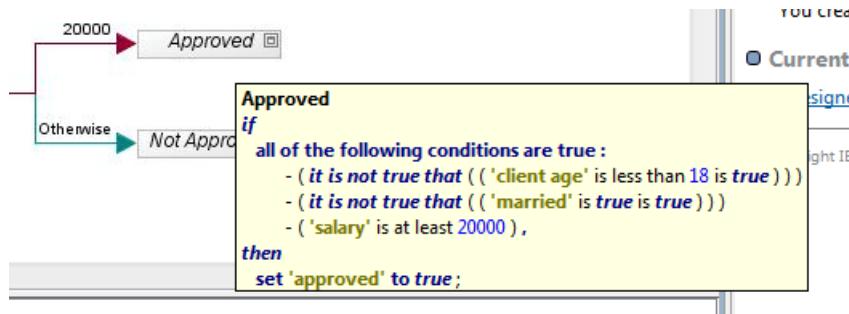


When defining branches, one of them can be flagged as an "otherwise" branch meaning it will be followed if none of the other expressions associated with the condition node evaluate to true.

An action can be defined to perform multiple functions by adding additional actions resulting in an "action set".



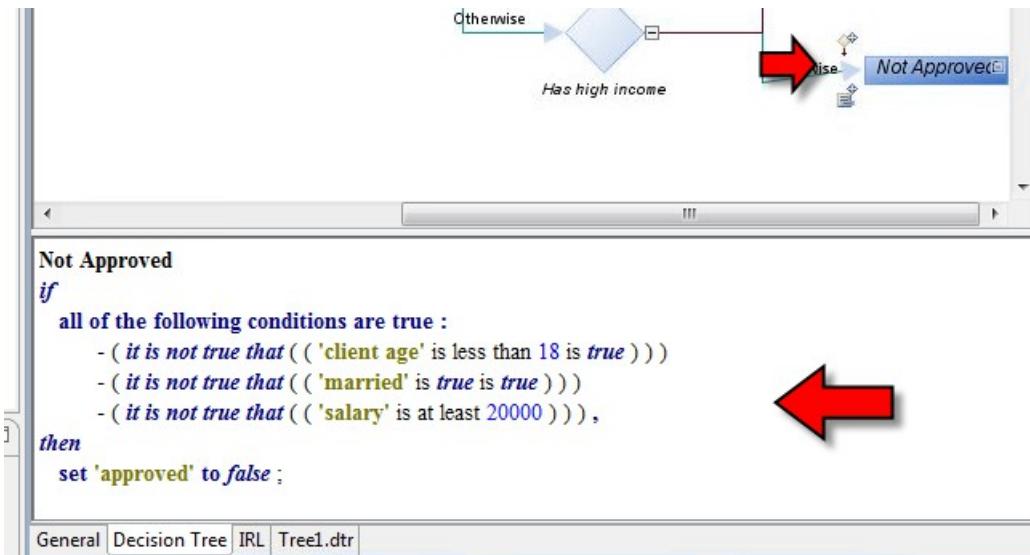
Hovering over any particular action will show the rule in more detail.



An area of the editor window can be set aside to always show the selected rule by clicking on this icon shown in the editor menu bar:



The result looks as follows:



## WODM – Event Processing

Prior to the release of WODM, IBM marketed a product called WebSphere Business Events (WBE) which was itself an acquisition many years prior. WBE as a separate product is now been consumed into WODM and has been significantly enhanced. The features of WBE that are now in WODM are known as event processing.

Historically, a technology discipline known as "Complex Event Processing" or "CEP" dealt with the arrival of events over time and performing actions based on patterns of those events. From a sales and marketing perspective, the "C" in "CEP" was not acceptable so the term was renamed to "Business Event Processing" or "BEP".

The feature of WODM that provides event processing is called "Decision Server Events" (DSE).

### Event Processing Architecture

It is extremely useful to understand the architecture of Event Processing in order to get the most out of it. We will start with the notion of an "Event". An event is a "message" that arrives from outside of DSE. How those messages arrive and what they look like is a story for later but let us simply assume that somehow these messages do arrive. The arrival of such a message corresponds to the arrival of an "event". The message itself is "typed" to represent a type of event. In addition to an event message having a type it can also carry data with in it. The content of this data is called "an event object" and is comprised of a flat level of named fields.

The next thing I want to introduce is a concept called a "rule". A rule is triggered by the arrival of an event. Rules are mapped to particular event types so a rule is not simply fired when any event occurs but only when events of a specific type occur. The rule contains logic which again will be discussed later. The logic determines what should happen as a result of the event arriving. At a high level, the primary outcome of an event arriving and a rule being examined is the firing of an action.

An action, similar to an event, is simply a "message". However, an action, unlike an event, is an outbound message. The action is the message "pushed-out" from DSE as a result of a rule choosing to do so. Similar to an event, an action message is also typed and can also carry an action payload.

Turning back to the notion of the rule, in order for the rule to determine what it should do, the rule is supplied with data. Commonly, that data comes from the incoming event payload. IBM has chosen to add a level of abstraction here. Instead of rules working directly against event payload data, rules are written to work against a logical data description called a "Business Object". When an event arrives, a business object is populated from the event object data and then the event object is discarded leaving only the business object. It is the business object that the rule operates against. Similarly, if a rule chooses to fire an action, the action is passed a copy of the business object and the business object's content is used to populate the action fields for the outgoing action message.

DES solutions are built in an Eclipse environment called "Event Designer".

The runtime for DES is a Java EE EAR application called "wberuntimeear" that is deployed to a WAS server. The DES runtime utilizes the services of a JDBC database to store its rules and event history and also uses JMS as the source of events.

## ***IBM Decision Server Events – Knowledge***

### **InfoCenter**

The InfoCenter for IBM Decision Server Events can be found here:

<http://publib.boulder.ibm.com/infocenter/dmanager/v7r5/index.jsp>

## ***Installing the Event Processing Business Space widgets***

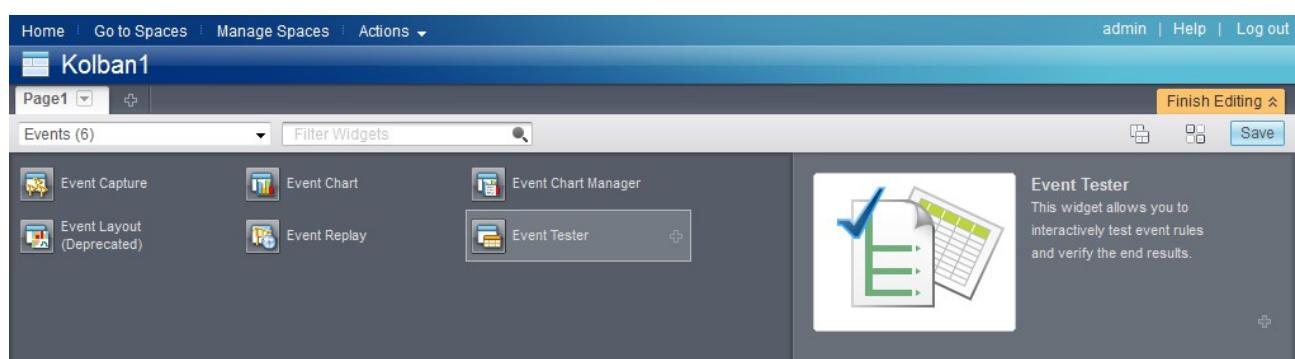
Business Space widgets are supplied for working with Event Processing. These must be installed. The widgets can be found in the <Install>/widgets folder.

```
AdminTask.installBusinessSpaceWidgets([
 '-nodeName', '<nodeName>',
 '-serverName', '<serverName>',
 '-widgets', '<InstallDir/widgets>'])
```

This can be run from wsadmin after starting it with:

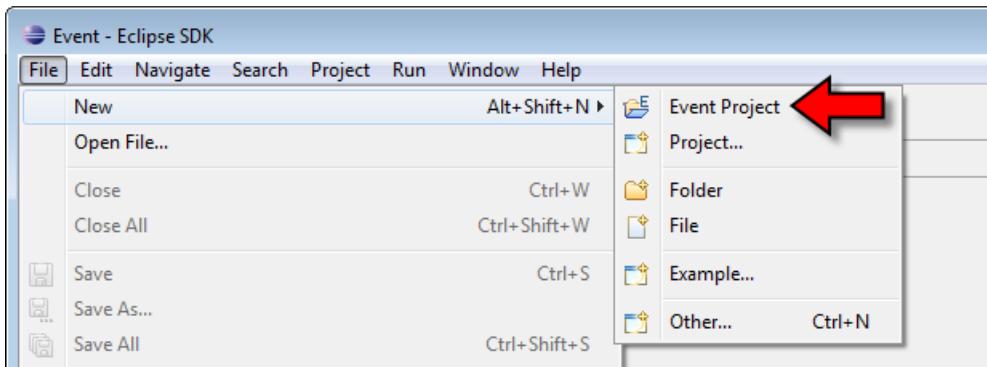
wsadmin -lang jython -conntype NONE

After installing, a reboot of the server is recommended. The widgets can be validated as having been installed by examining Business Space and seeing that there is a new "Events" category with six new widgets:

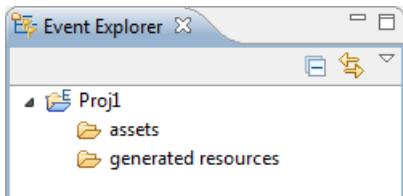


## ***Creating an Event Project***

An Event Project can be created from the Event Perspective:



once created, it looks like:



### **The DES console**

<http://localhost:9080/wbe/common/login.jsp>

## **The Rule Execution Server**

When design and development of the rule solution has completed, the solution has to be made runnable so that client can make use of the decisions. This is achieved by deploying the solution to a Rule Execution Server.

The Rule Execution Server is the core run-time component of WODM. It is deployed on a WebSphere Application Server.

The Rule Execution Server is itself broken down into a series of logical components:

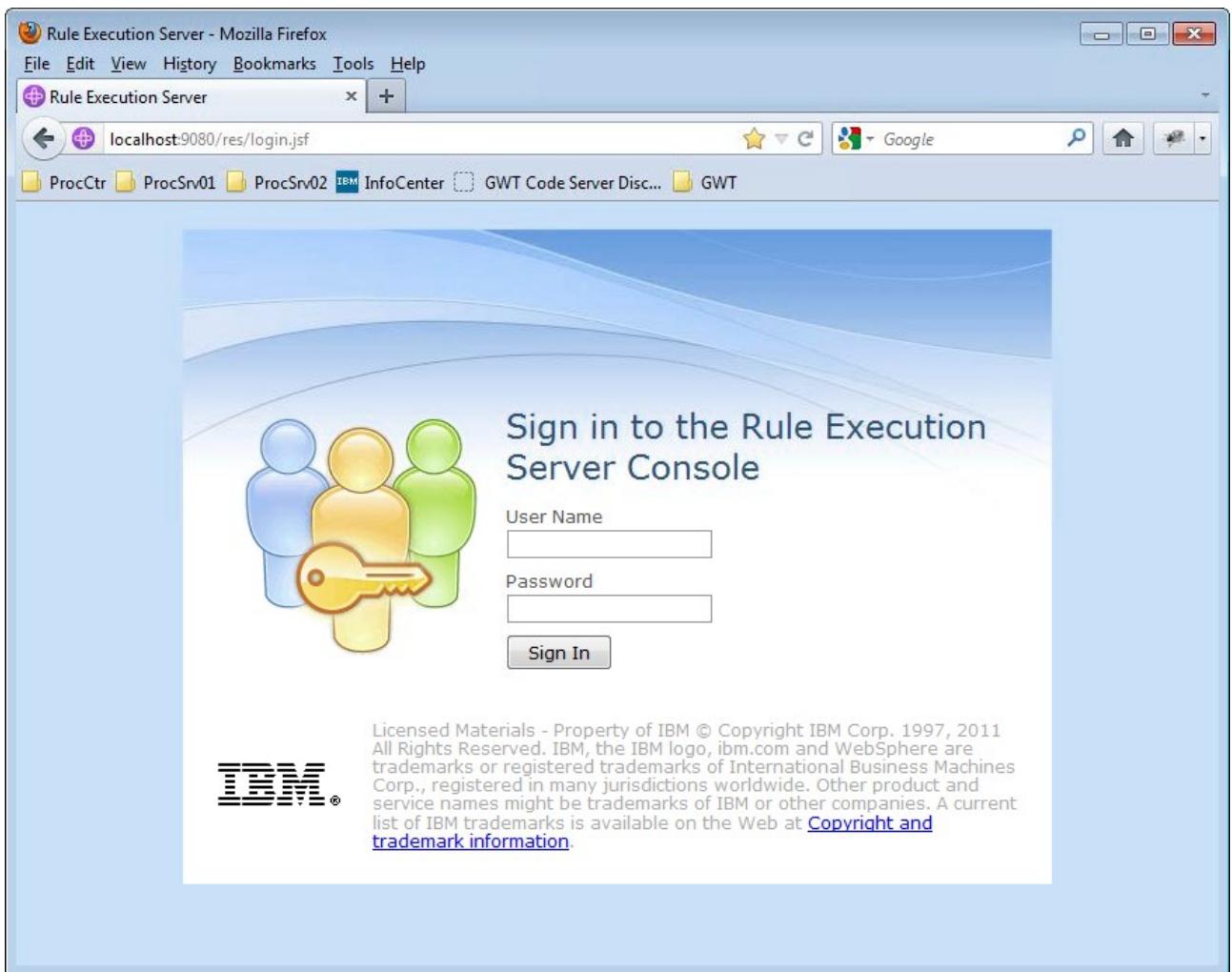
- Execution Unit (XU)
- Java EE execution components
- Java SE execution components
- JMX management and execution model
- Management console
- Transparent decision services

### **Rule Execution Server Console**

Once the Rule Execution Server has been installed, a console can be launched to perform various management functions against it. The console can be launched at:

<http://localhost:9080/res>

An initial login window is shown:



after login, the main window shows the available options:

A screenshot of the IBM Rule Execution Server Console. The top navigation bar includes the IBM logo, "Rule Execution Server Console", and user status "resAdmin Sign Out". The menu bar has tabs for "Home", "Explorer", "Decision Warehouse", "Diagnostics", and "Server Info". The "Home" tab is currently selected. Below the menu, there's a "Welcome to the Rule Execution Server Console" message. Underneath, there are four items with descriptions: "Explorer" (Use the Explorer to deploy, browse, and modify RuleApps.), "Decision Warehouse" (Search and view decision traces.), "Diagnostics" (Run the server diagnostics to verify installation.), and "Server Info" (View server configuration information and logged events.). At the bottom left, a copyright notice reads "© Copyright IBM Corp. 1987, 2011".

## **Hosted Transparent Decision Services – Web Services**

Imagine that we want decisions to be made from calling applications which are remote from a Decision Server. In order to do this, we can expose rules as Web Services and have the rules executed when a Web Service call is made.

To activate this function install the EAR found at:

```
<Install>/executionserver/applicationservers/WebSphere7/jrules-res-htds-WAS7.ear
```

After installing this component rules are then exposed as Web Services for invocation.

Within the Rules Execution Server console, if we drill down to a particular rule project, we can access the Hosted Transparent Decision Services (HTDS) options. This includes the ability to retrieve the WSDL description of the exposed Web Service.

The screenshot shows the 'HTDS WSDL Options' panel. It includes a 'Hide HTDS WSDL Options' link, a 'HTDS WSDL Options' icon, and four entries: 'Hosted transparent decision service location' (http://localhost:9080/DecisionService), 'Web service endpoint' (http://localhost:9080/DecisionService), 'Decision service target namespace' (http://www.ibm.com/rules/decisionservice/AgeTestApp/AgeTest), and 'Decision service parameter target namespace' (http://www.ibm.com/rules/decisionservice/AgeTestApp/AgeTest/param). There is also a 'Retrieve HTDS WSDL File' link.

Selecting the option to retrieve the WSDL file results in a further panel:

The screenshot shows the 'Retrieve HTDS WSDL File' dialog box. It has a title bar with a sun icon and the text 'Retrieve HTDS WSDL File'. Below it is a URL field containing '/AgeTestApp/1.0/AgeTest/1.0'. Underneath are several checkboxes: 'Latest ruleset version', 'Latest RuleApp version', 'Decision trace information', 'Inline types in separate XSD files', and 'Compatibility mode' (set to 7.0). At the bottom are 'Cancel', 'View', and 'Download' buttons.

from here, additional options can be supplied.

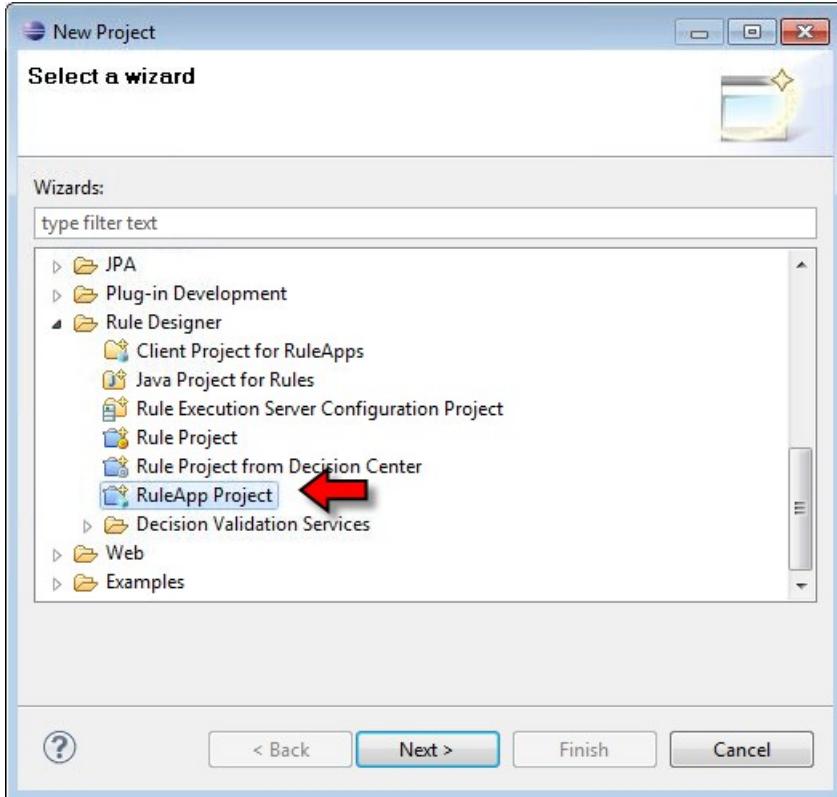
## **Rule Application**

The Rule Application (RuleApp) is the deployable component to the Rule Execution Server. The RuleApp is defined in Rule Designer as a new project (of type RuleApp). To create a RuleApp project, open the Rule Designer file menu and select New > Project.

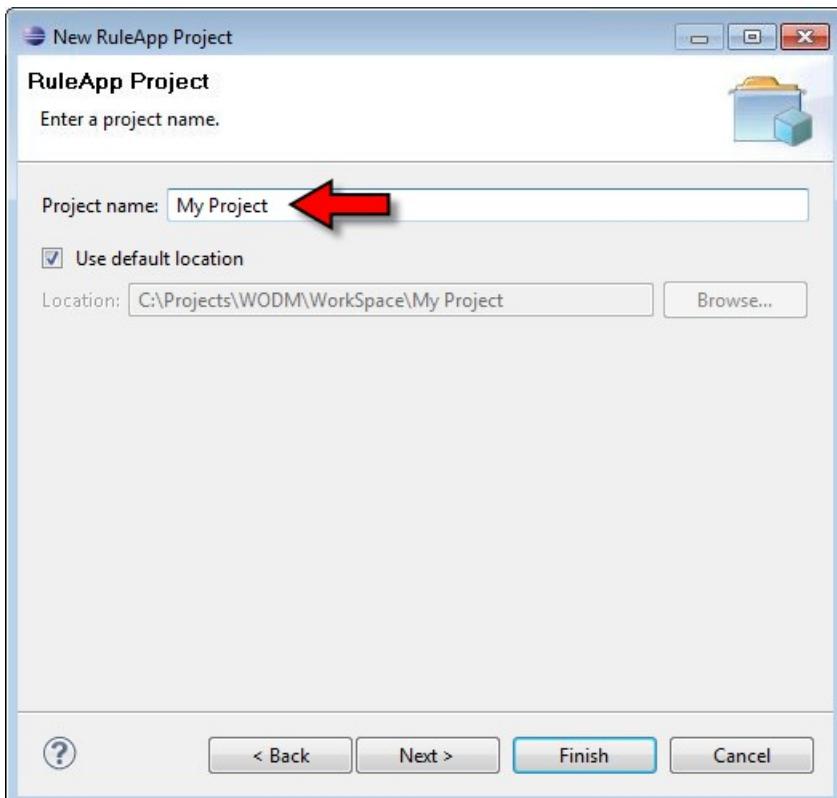
The screenshot shows the 'File' menu of the 'Rule - Eclipse SDK' application. The 'New' submenu is open, displaying options: 'Rule Project' (highlighted with a red arrow), 'Project...', 'Rule Package', 'BOM Entry', 'Action Rule', 'Action Rule Template', 'Decision Table', and 'Decision Table Template'. The 'File' menu also contains 'Edit', 'Navigate', 'Search', 'Project', 'Run', 'Window', and 'Help'.

In the dialog of allowable project types, the RuleApp Project type can be found in the Rule

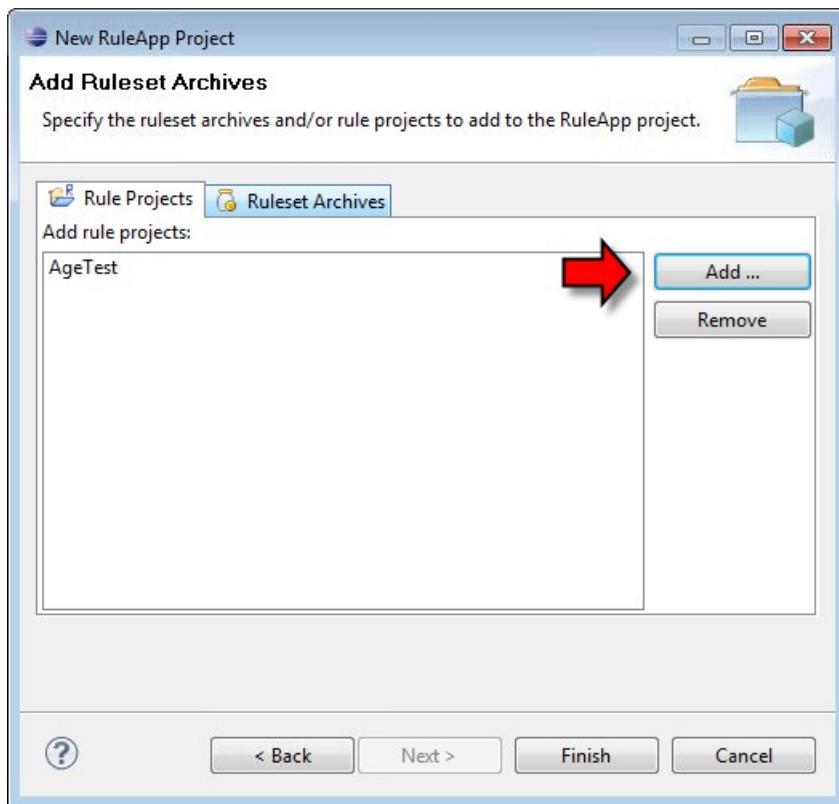
## Designer catalog.



Selecting to create a new Rule Designer project opens a new dialog into which the name of the new RuleApp project may be entered.



At this time the project's creation can be completed or else Rule Projects may be added.



Once done, the RuleApp has now been created and has its own editors for configuring the primary attributes of a RuleApp project.

**General Information**

Name: MyProject  
Display name: My Project  
Version: 1.0  
Creation date: Thursday, March 22, 2012

Description:

**Deployment**

You can:

- [Export](#) a RuleApp archive.
- [Deploy](#) a RuleApp to one or more Rule Execution Servers.

**Ruleset Archives**

This RuleApp project has the following archives:  
AgeTest

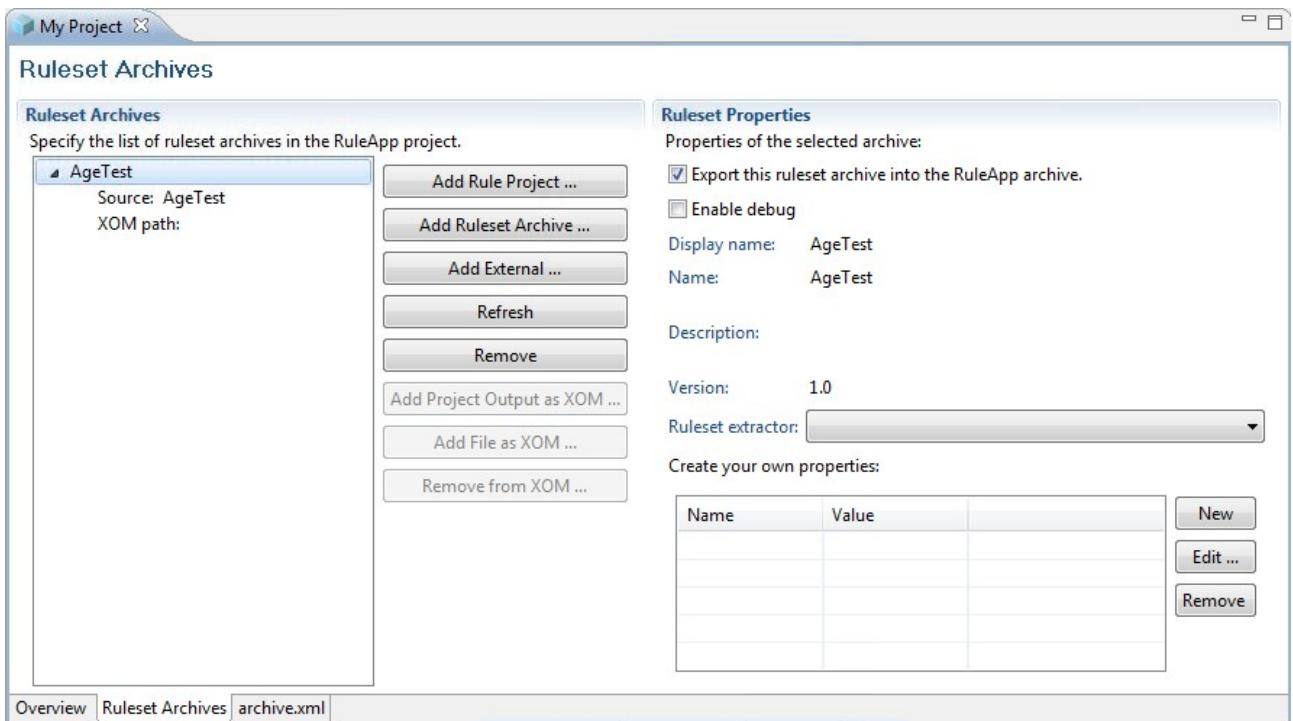
**RuleApp Properties**

Define the properties to be attached to the RuleApp.

Name	Value

New  
Edit ...  
Remove

Overview Ruleset Archives archive.xml



Overview Ruleset Archives archive.xml

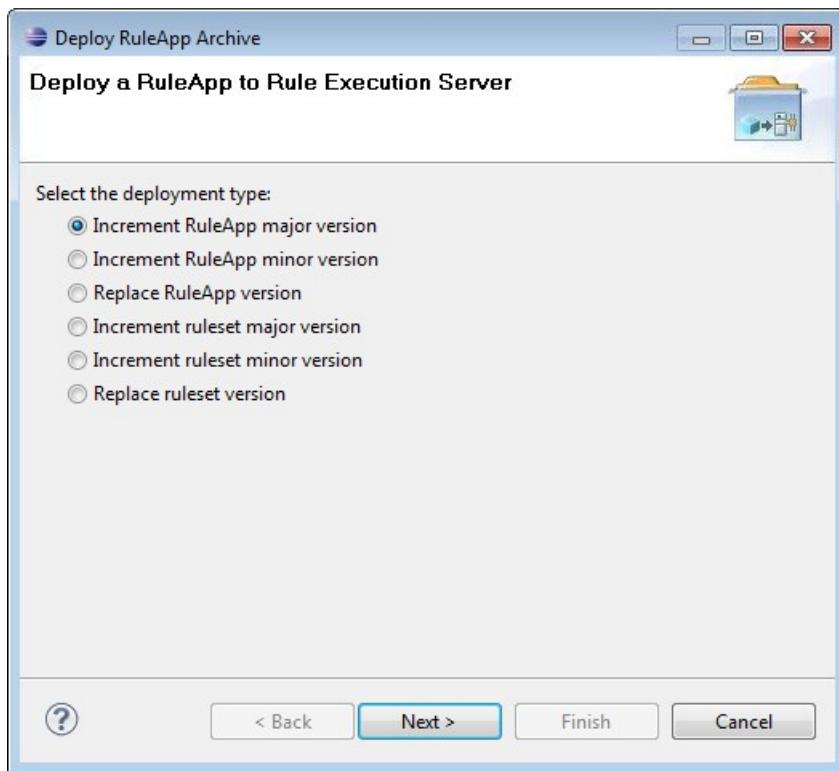
## Deploying a RuleApp

Once a RuleApp has been created, it can be deployed from Rule Designer directly into a Rule Execution Server. This may be useful for development testing but is unlikely to be desirable in a production system. Instead, a Rule App can be exported as an "archive" which can then be deployed either from the Rule Execution Server console or from an Ant task called res-deploy.

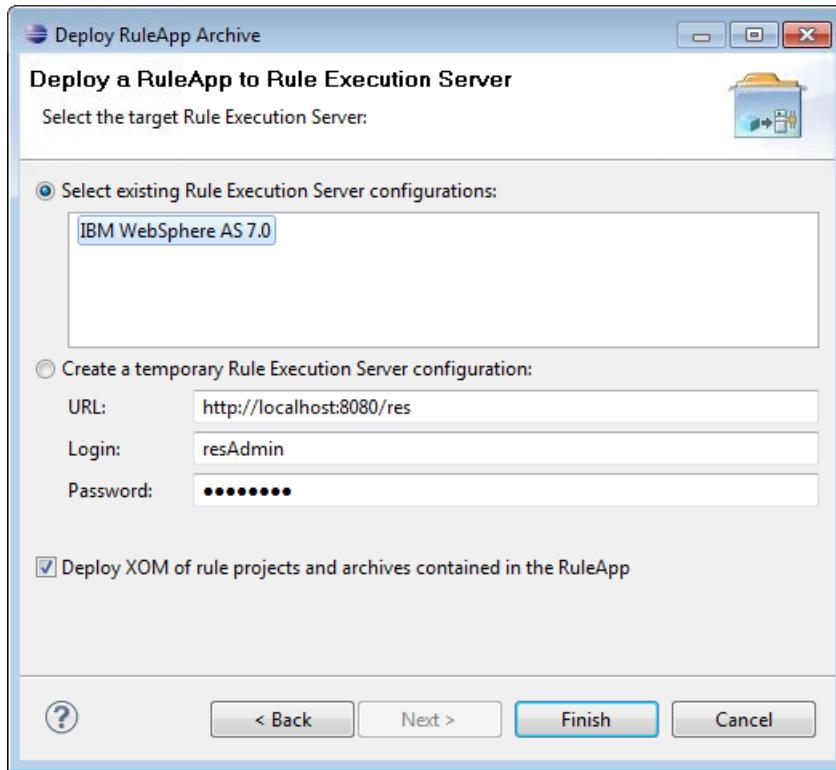
To deploy the RuleApp to the Rule Execution Server from Rule Designer, we can select Deploy from the RuleApp Project:



Next we are asked how we want to augment or replace any existing version that may already be on the server:



Finally, we are asked about the location of the Rule Execution Server to which we wish to deploy:



For the other deployment solutions, a Rule App must first be exported from the Rule Designer. The result of an export is a new file on the file system. This file contains everything needed to deploy the RuleApp to a Rule Execution Server. The file is a ".jar" file.

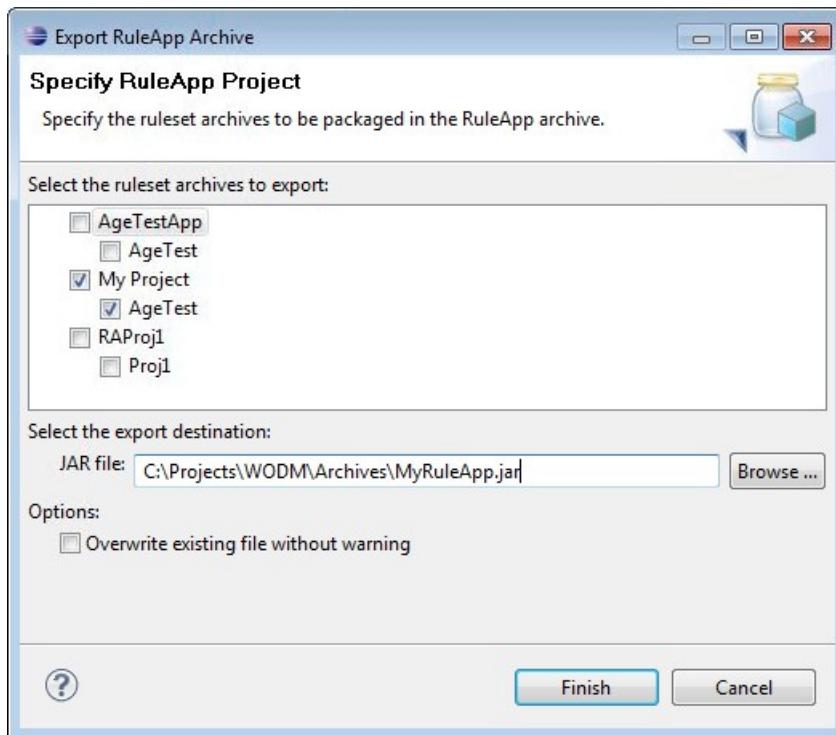
## Deployment

You can:

[Export a RuleApp archive.](#)



[Deploy a RuleApp to one or more Rule Execution Servers.](#)



Once the deployment archive has been created, we can deploy it to a Rule Execution Server through the Rule Execution Server console.

IBM Rule Execution Server Console

Home Explorer Decision Warehouse Diagnostics Server Info

Explorer > RuleApps

**RuleApps View**

Add RuleApp Deploy RuleApp Archive **Get RuleApps** (red arrow)

**RuleApps**

Total Number of RuleApps 1

1 RuleApp(s) Name:  View only:

Select All	Name	Version	Creation Date	Number of
<input type="checkbox"/>	RAProj1	1.0	Mar 19, 2012 4:45:40 PM GMT-05:00	1

RuleApp 1 - 1 of 1 prev 10 next 10

Navigator

- + RuleApps (1)
  - Resources
  - Libraries
- + Decision Services (1)

IBM Rule Execution Server Console

resAdmin S

Home Explorer Decision Warehouse Diagnostics Server Info About | Print View

Explorer > RuleApps > Deploy RuleApp

**RuleApps View**

Help

**Deploy RuleApp Archive**

Local path of RuleApp Archive:  **Browse...** (red arrow)

Versioning Policy

Increment RuleApp major version  
 Increment RuleApp minor version  
 Replace RuleApp version  
 Increment Ruleset major version  
 Increment Ruleset minor version  
 Replace Ruleset version

**Cancel** **Deploy** (red arrow)

Navigator

- + RuleApps (1)
  - Resources
  - Libraries
- + Decision Services (1)

## RuleApps View

Help ?

### Deployment Report

RuleApp archive deployed MyRuleApp.jar  
Versioning Policy Increment RuleApp major version

Details		
Archive content	Operation	Result
/MyProject/1.0 /AgeTest/1.0	+ Element added + Element added	/MyProject/1.0 /AgeTest/1.0

Ok

## *RuleApp Properties*

Name	Value	
		<a href="#">New</a>
		<a href="#">Edit ...</a>
		<a href="#">Remove</a>

# **WODM – Decision Center**

Decision Center is where business users can manage rules executed by Rules Execution Servers. In previous releases of the product, the function known as Decision Center used to be called "Rule Team Server". This still surfaces from time to time in the documentation and commands.

## *Configuring Decision Center*

Decision Center requires the following security configurations:

## Groups

- rtsAdministrator
  - rtsConfigManager
  - rtsUser
  - rtsInstaller
  - Validator
  - Eligibility

A JDBC Data Source must be defined called jdbc/ilogDataSource

The EAR file for deployment of Decision Center is:

<Install>/teamserver/applicationservers/WebSphere7/jrules-teamserver-WAS7.ear

After installation, when the Decision Center is first opened, a set of configuration options are presented:

IBM. Decision Center

About | Print View | Help    resAdmin Sign Out

Install

Install Home Step 1: Configure Database Step 2: Setup Message File Step 3: Setup Groups Step 4: Set Persistence Locale Step 5: Set Configuration Parameters

## Installation Settings Wizard

Use the Installation Settings Wizard to complete or modify the installation. To complete an initial installation, click Next. To modify an existing installation, click on the corresponding step.

[⑦ Click here for help on using the Installation Settings Wizard](#)

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

IBM. Decision Center

About | Print View | Help    resAdmin Sign Out

Install

Install Home > Step 1: Configure Database

## Configure the Decision Center database

Configure the Decision Center database by first generating the SQL script that creates the database schema from the rule model and extensions you specify, then executing the script and initializing the database

**Select the extension files you want to use:**

Default extensions  Custom extensions (brmx/brdx)  Custom extensions (Zip)

[Generate SQL](#)

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

IBM. Decision Center

About | Print View | Help resAdmin Sign Out

**Install**

Install Home > Step 2: Setup Message File

Install Home Step 1: Configure Database Step 2: Setup Message File Step 3: Setup Groups Step 4: Set Persistence Locale Step 5: Set Configuration Parameters

**Associate message files to extensions**

Upload message files to control the display of extensions

**Message file(s)**

de  
 en  
 es  
 fr  
 it  
 ja  
 ko  
 nl  
 pl  
 pt\_BR

Previous 1 2 Next

13 Results

New Delete

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

IBM. Decision Center

About | Print View | Help resAdmin Sign Out

**Install**

Install Home > Step 3: Setup Groups

Install Home Step 1: Configure Database Step 2: Setup Message File Step 3: Setup Groups Step 4: Set Persistence Locale Step 5: Set Configuration Parameters

**Setup groups**

To use Decision Center project security as well as its permissions mechanism, all groups (except rtsAdministrator) must be declared in both the application server and here

No item found

New Delete

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

**IBM. Decision Center**

About | Print View | Help    resAdmin    Sign Out

**Install**

Install Home > Step 4: Set Persistence Locale

Install Home Step 1: Configure Database Step 2: Setup Message File Step 3: Setup Groups Step 4: Set Persistence Locale Step 5: Set Configuration Parameters

**Set the persistence locale of the repository**

Specify the persistence locale, which determines the language in which rules are stored in the Decision Center database (jdbc/logDataSource). The current persistence locale is English (United States). Do not specify a locale if the current language is suitable

Locale:

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

**IBM. Decision Center**

About | Print View | Help    resAdmin    Sign Out

**Install**

Install Home > Step 5: Set Configuration Parameters

Install Home Step 1: Configure Database Step 2: Setup Message File Step 3: Setup Groups Step 4: Set Persistence Locale Step 5: Set Configuration Parameters

**Set configuration parameters**

Create, modify, or delete configuration parameters used to customize Decision Center

No item found

New Edit Delete

Exit Previous Next Finish

© Copyright IBM Corp. 1987, 2011

IBM. Decision Center

About | Print View | Help    resAdmin Sign Out

Install

Install Home

Step 1: Configure Database

Step 2: Setup Message File

Step 3: Setup Groups

Step 4: Set Persistence Locale

Step 5: Set Configuration Parameters

**Installation log**

You performed the following operations:

Performed operations  
 1 - Database management  
    Schema creation

Note that you will have to publish a rule project from Rule Designer to Decision Center. You will now be rerouted to the Decision Center sign-in page

OK

© Copyright IBM Corp. 1987, 2011

IBM. Decision Center

About | Print View | Help    resAdmin Sign Out

Configure

**Diagnostics**

Expand All  Collapse All

About

Manager Bean Access

Data source

Extensions

Verbalizers

Database

© Copyright IBM Corp. 1987, 2011

---

## Business Process Management scenarios

A common request is to show IBPM performing some specific role. This request is made by

prospective consumers of the product. IBPM is not the only BPM offering in the marketplace so it seems only natural to ask for such a comparative demonstration. Since IBPM is **not** a turn-key solution but is instead middle-ware that is to be used to build a solution, such demonstrations of capability need to be built. To build a new demonstration each time a customer asks for one is going to be too expensive. Instead, what we should look to do is to create a portfolio of existing demonstrations that can be reused. The characteristics of these demonstrations are:

- Be well documented
- Include a presentation to describe what is to be seen
- Be executable on the very latest versions of the product
- Include a video presentation to be shown when no execution environment or demonstrator is available

It is also my belief that these scenarios should also have the following additional characteristics

- Be short
- Be simple
- Be as industry agnostic as possible

### ***Scenario: Exception Handling***

In many processes, handling of work requests can be mostly automated or else follows a very well oiled sequence of steps that has been well tuned over time. What has been seen as a great expense to businesses is what is known as exception handling. Exception handling is the task of handling process work requests that didn't follow the desired or expected path. Here are some examples:

- A customer's credit card is declined in an on-line purchase
- A medication to be sent to a patient has an adverse interaction with another medicine also supplied to the customer
- A stock purchase can not be fulfilled because the stock being purchased has been suspended
- An insurance form is missing information. The VIN of the vehicle being insured has not been supplied.

In these examples, the process can not be completed until some other steps have been performed. These steps are usually manual in nature involving a staff member being told of the nature of the problem and that staff member deciding what should happen next.

One of the primary characteristics of exception handling is determining which staff members would be appropriate for resolving a particular issue. A simple solution would be to assign the exception processing to all members of a particular team or group. In many cases however, this will not be satisfactory. Instead, the request should be sent to a set of folks who have the skills or responsibility to perform the work. In this case, the determination of which users should be included may be a function of the nature or details of the exception being processed.

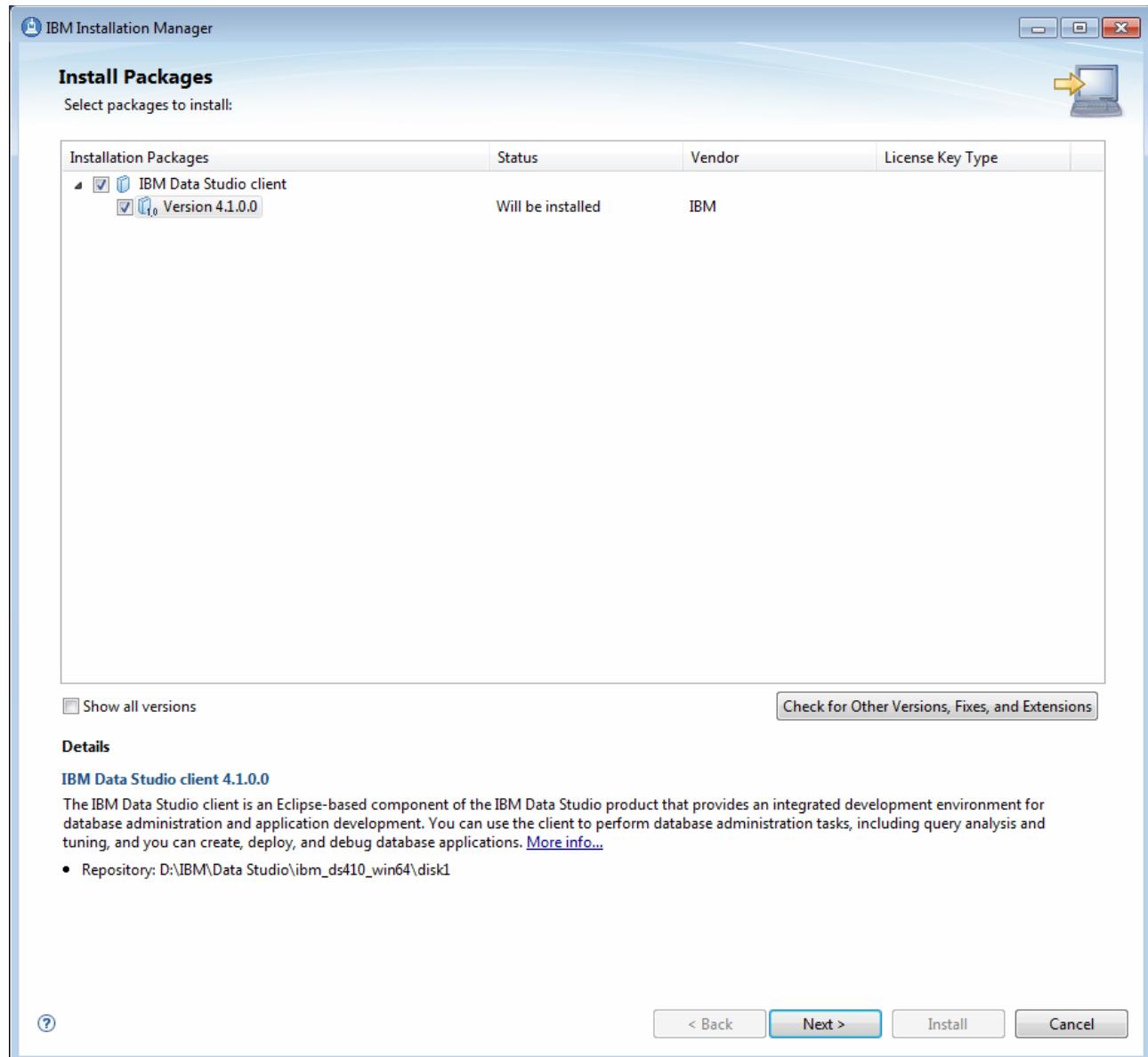
## **Advanced IBPM Troubleshooting**

### ***Data Studio***

For the BPM v8.5 release and onwards, DB2 is now at v10. This saw the replacement of the DB2 Control Center tool with the IBM Data Studio product. This component is available for download for free here:

<http://www-03.ibm.com/software/products/us/en/data-studio>

Here is an installation walk-through



IBM Installation Manager

## Install Packages

Read the following license agreements carefully.

Install Licenses Location Features Summary

IBM Data Studio client  
IBM Data Studio client  
Notices

International Program License Agreement

Part 1 - General Terms

BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, CLICKING ON AN "ACCEPT" BUTTON, OR OTHERWISE USING THE PROGRAM, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF LICENSEE, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND LICENSEE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,

\* DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, CLICK ON AN "ACCEPT" BUTTON, OR USE THE PROGRAM; AND

\* PROMPTLY RETURN THE UNUSED MEDIA, DOCUMENTATION, AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM IT WAS OBTAINED FOR A REFUND OF THE AMOUNT PAID. IF THE PROGRAM WAS DOWNLOADED, DESTROY ALL COPIES OF THE PROGRAM.

1. Definitions

"Authorized Use" - the specified level at which Licensee is authorized to execute or run the Program. That level may be measured by number of users, millions of service units ("MSUs"), Processor Value Units ("PVUs"), or other level of use specified by IBM.

"IBM" - International Business Machines Corporation or one of its subsidiaries.

"License Information" ("LI") - a document that provides information and any additional terms specific to a Program. The Program's LI is available at [www.ibm.com/software/sla](http://www.ibm.com/software/sla). The LI can also be found in the Program's directory, by the use of a system command, or as a booklet included with the Program.

"Program" - the following, including the original and all whole or partial copies: 1) machine-readable instructions and data, 2) components, files, and modules, 3) audio-visual content (such as images, text, recordings, or pictures), and 4) related licensed materials (such as keys and documentation).

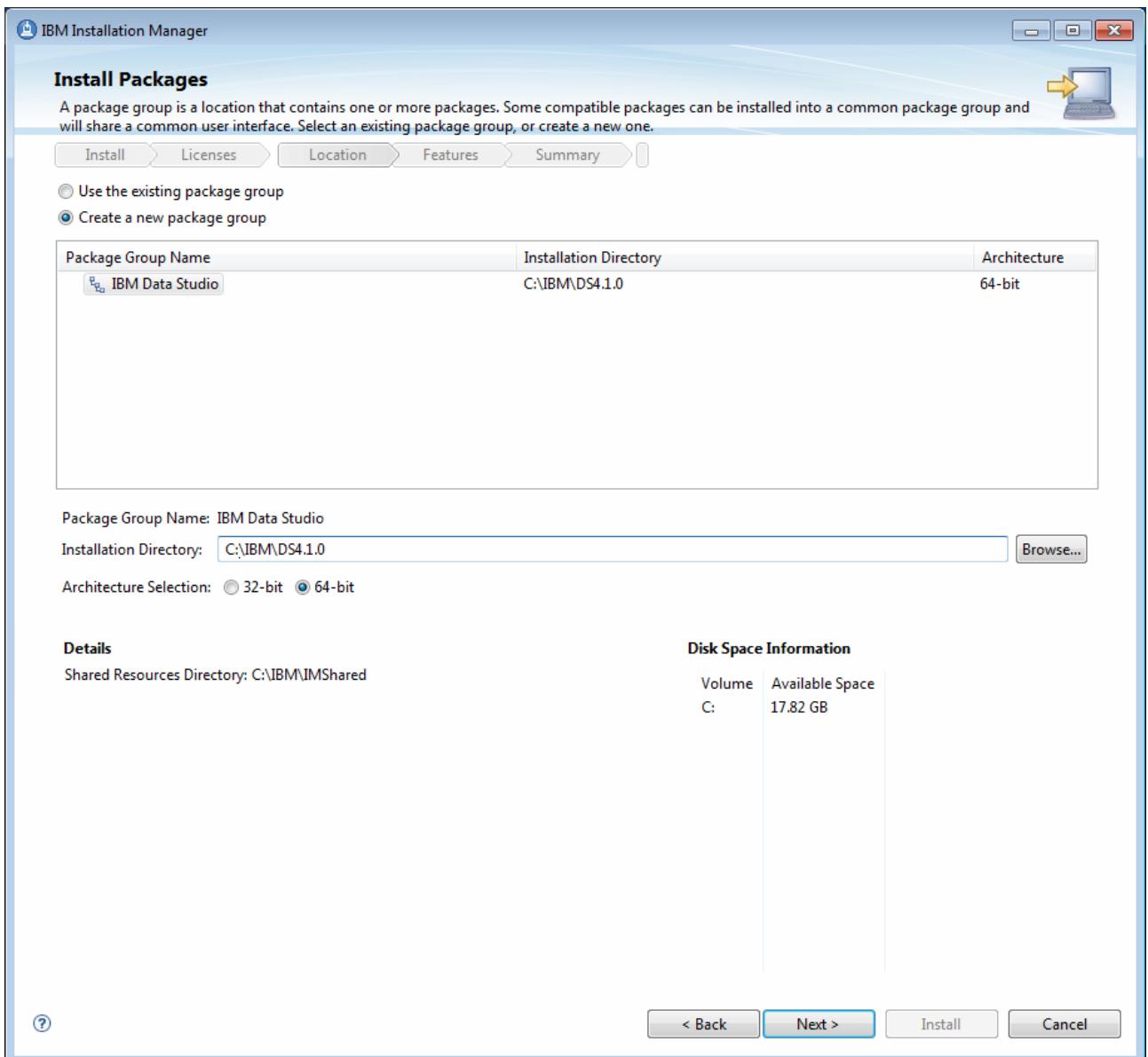
"Proof of Entitlement" ("PoE") - evidence of Licensee's Authorized Use. The PoE is also evidence of Licensee's eligibility for warranty, future update prices, if any, and potential special or promotional opportunities. If IBM does not provide Licensee with a PoE, then IBM may accept as the PoE the original paid sales receipt or other sales record from the party (either IBM or its reseller) from whom Licensee obtained the Program, provided that it specifies the Program name and Authorized Use obtained.

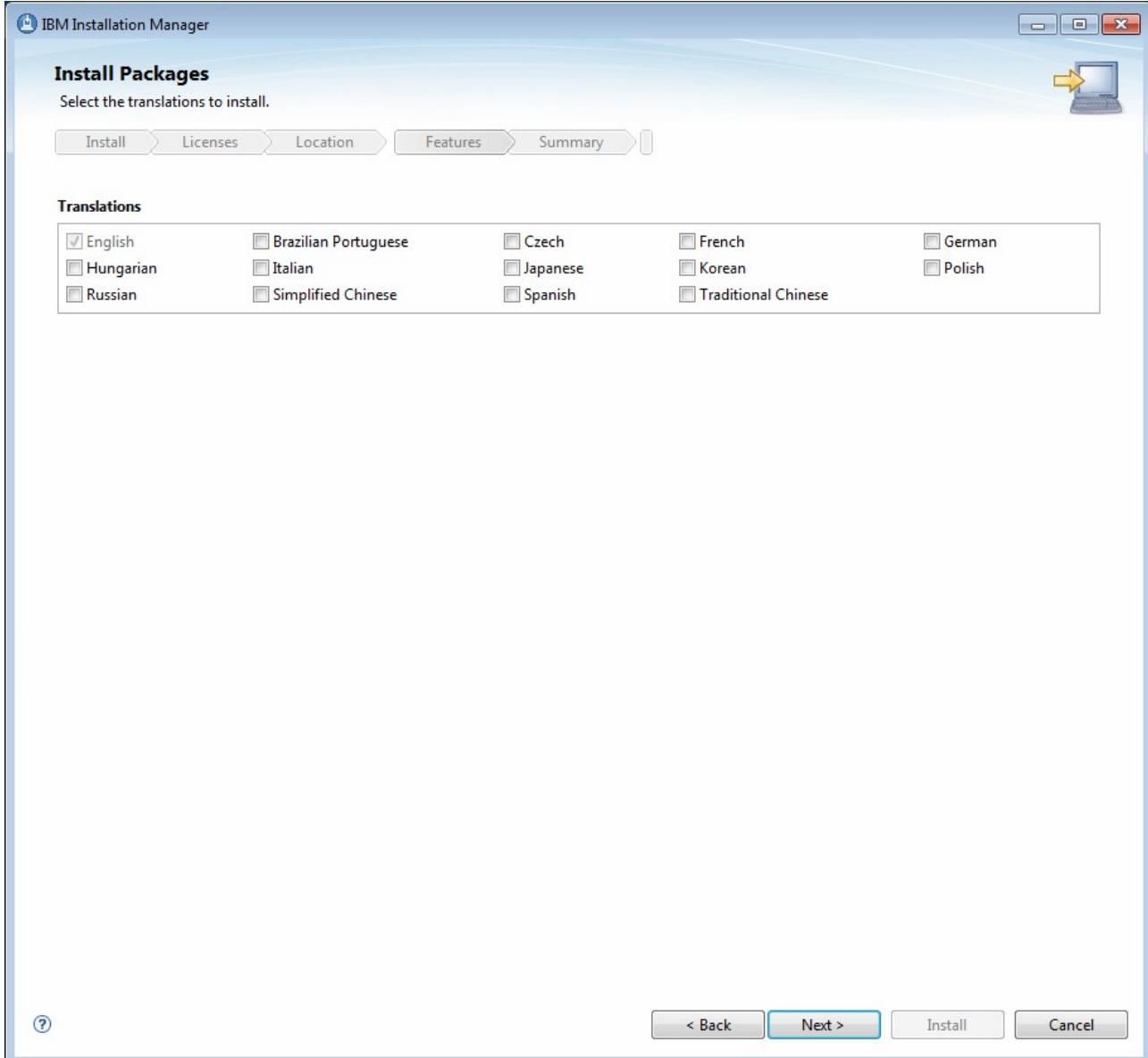
"Warranty Period" - one year, starting on the date the original Licensee is granted the license.

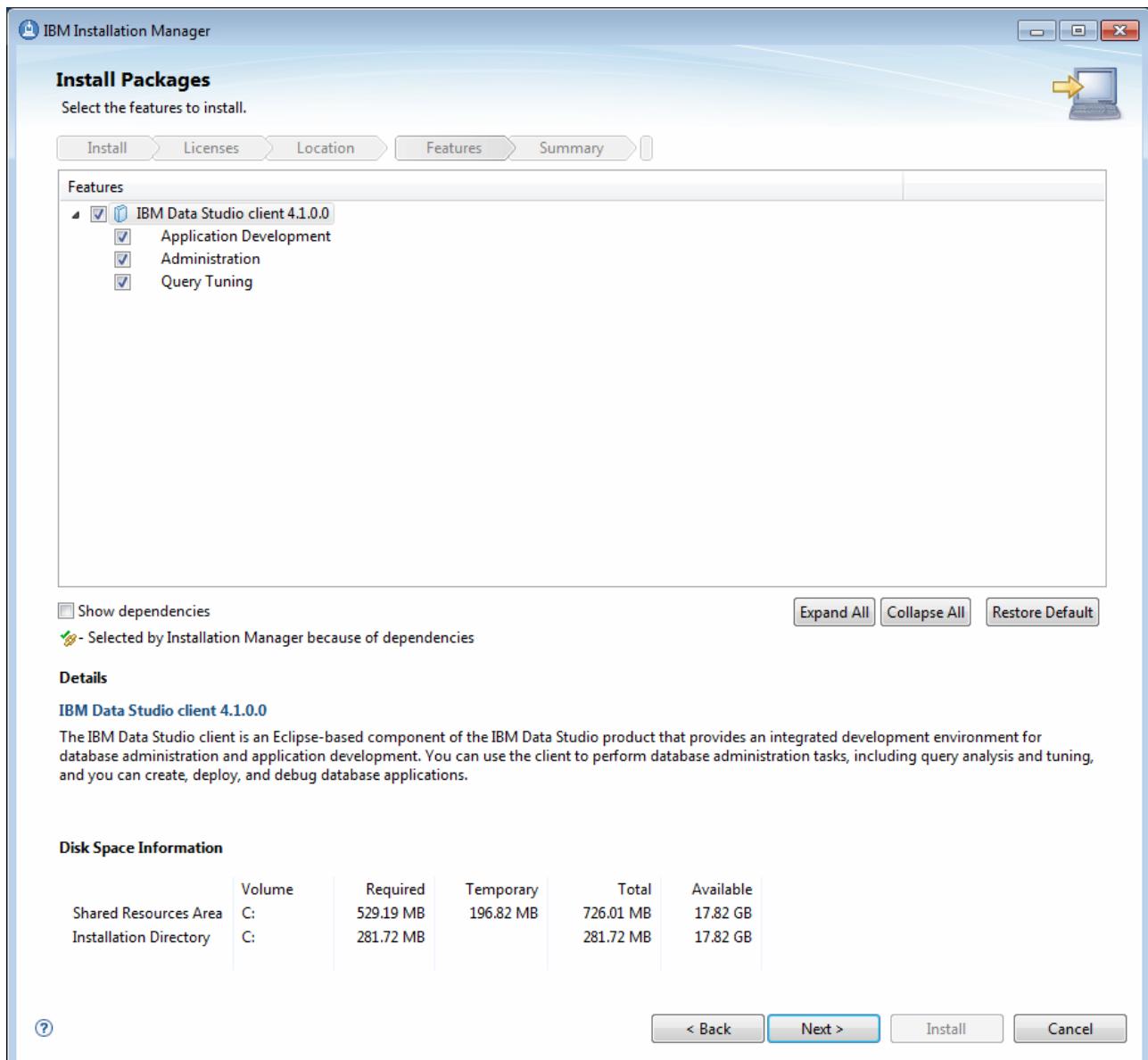
I accept the terms in the license agreements  
 I do not accept the terms in the license agreements

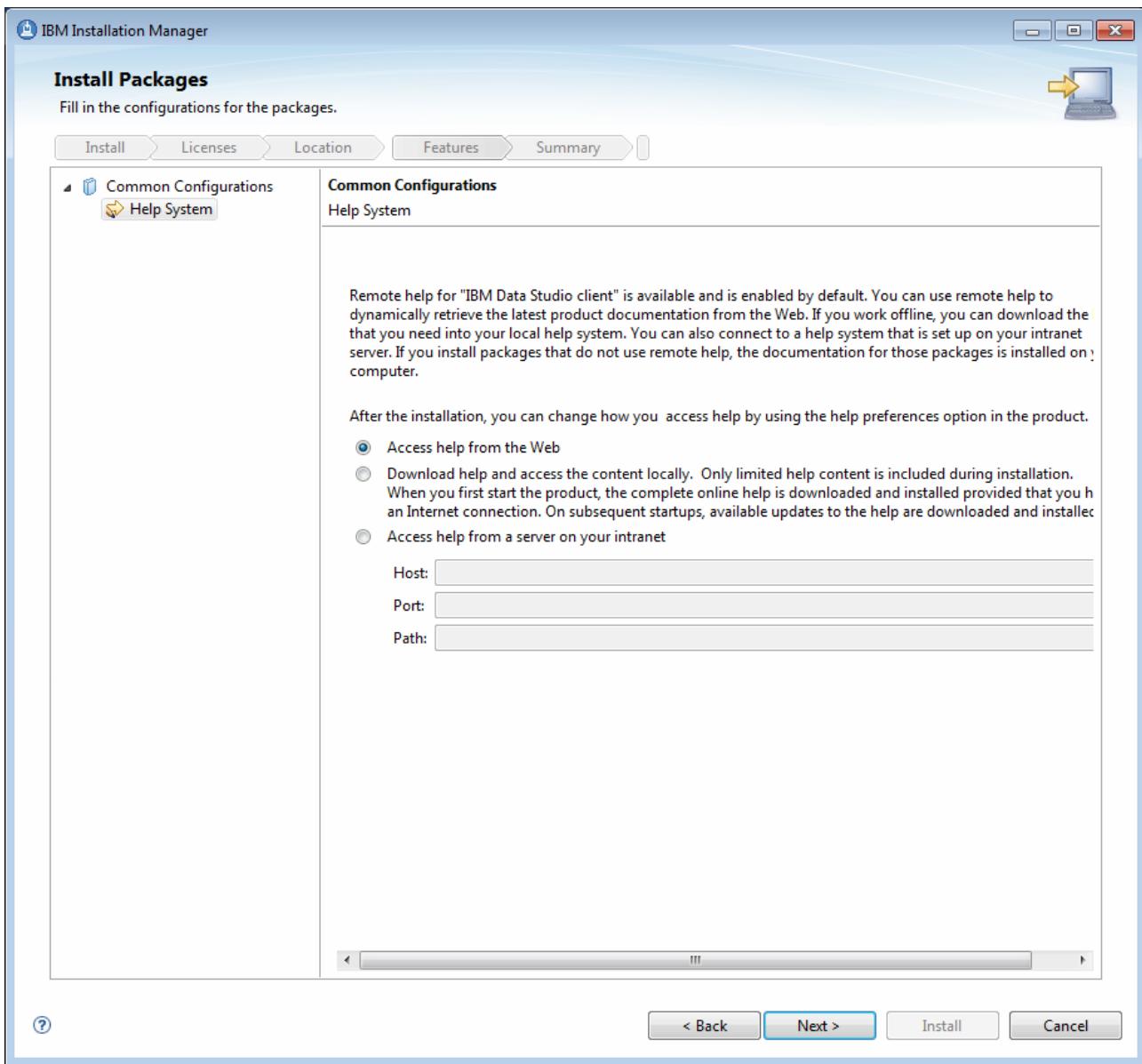
Print All...

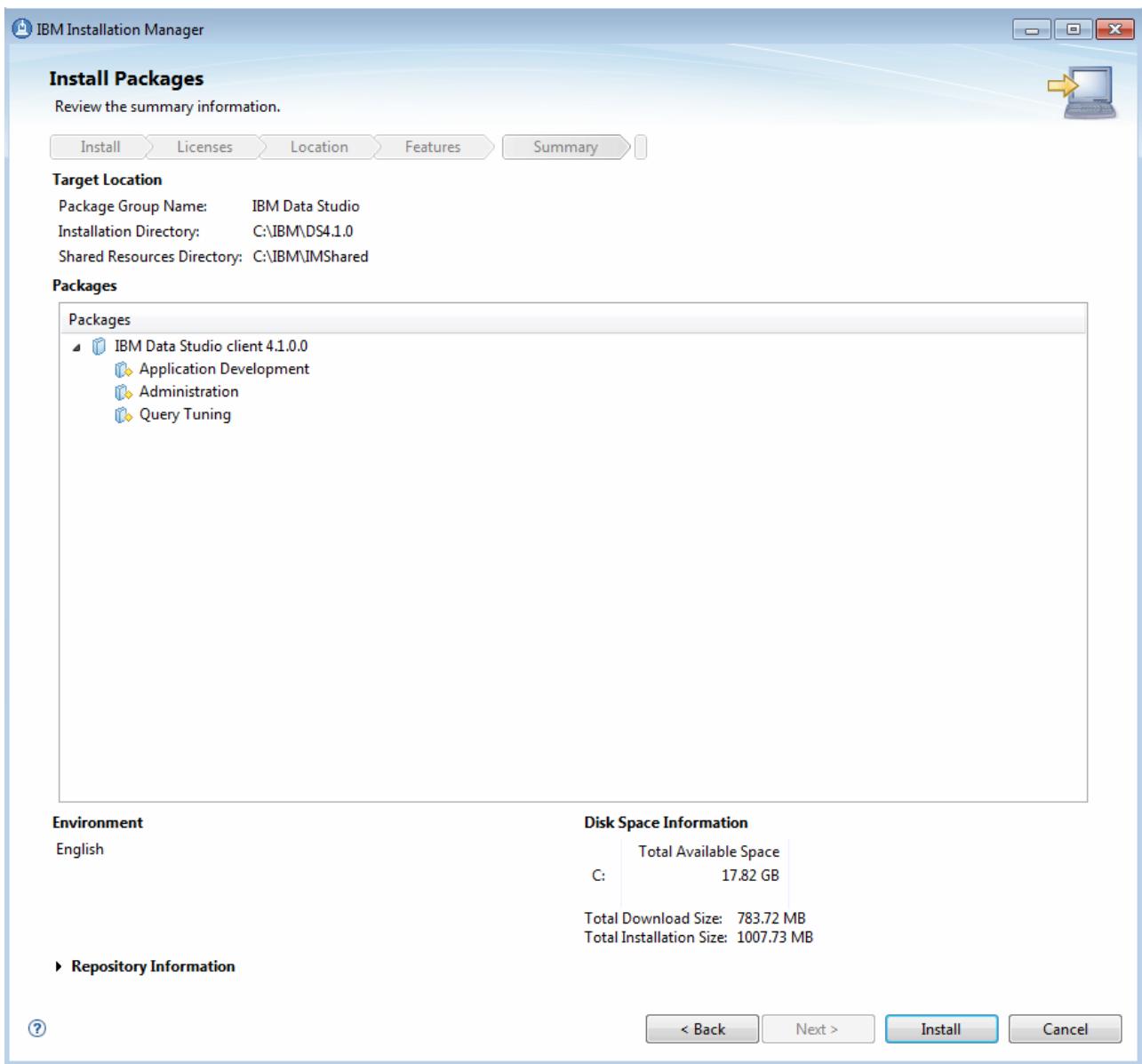
< Back Next > Install Cancel

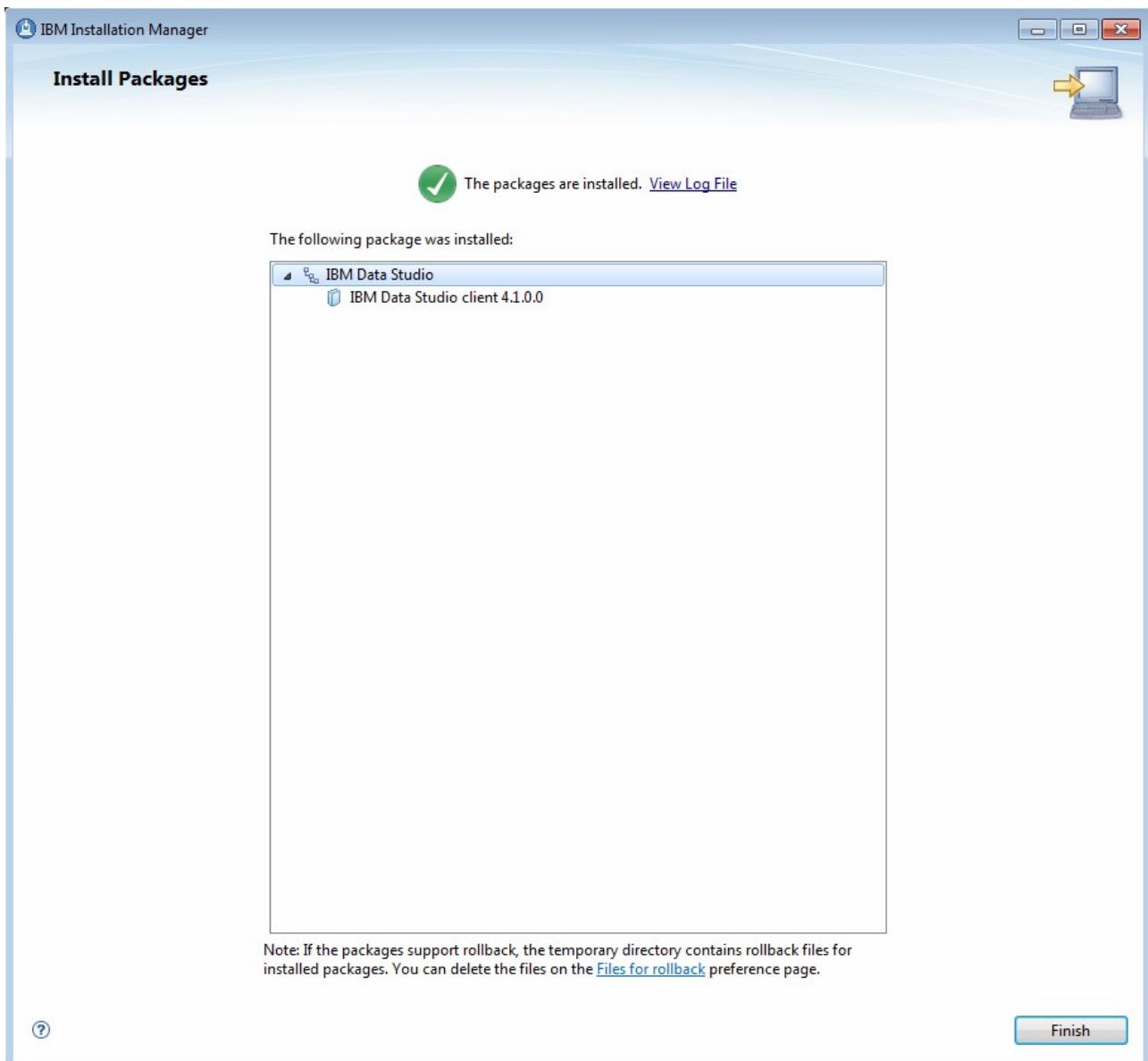












See also:

- DeveloperWorks - [Migrating from DB2 Control Center to IBM Data Studio](#) - 2012-01-19

## ***DB2 Troubleshooting***

db2admin was found to be consuming 100% CPU after a power outage and restart. A recipe was found to "reset" it with the following recipe:

```
db2admin stop
db2admin drop
db2admin create /user=db2admin /password=db2admin
db2admin start
```

To display the message text associated with a SQLSTATE error, open a DB2 command window and enter:

```
? nnnnn
```

Where nnnnn is the SQLSTATE value. A human readable description of the message will be displayed.

To explicitly grant authorities to a DB user for debugging, run the following script:

```

CONNECT TO PDWDB;

REVOKE ACCESSCTRL, DATAACCESS ON DATABASE FROM USER KOLBAN;

GRANT
CREATETAB,BINDADD,CONNECT,CREATE_NOT_FENCED_ROUTINE,IMPLICIT_SCHEMA,LOAD,CREATE_
EXTERNAL_ROUTINE,QUIESCE_CONNECT ON DATABASE TO USER KOLBAN;

CONNECT RESET;

```

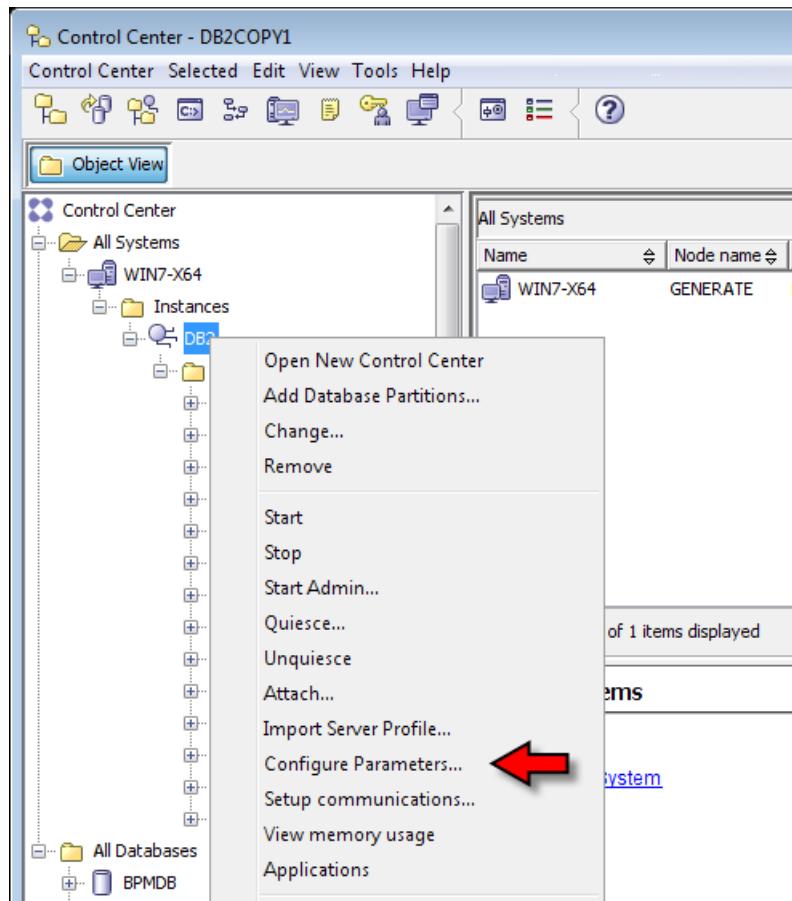
## Insufficient DB2 connections

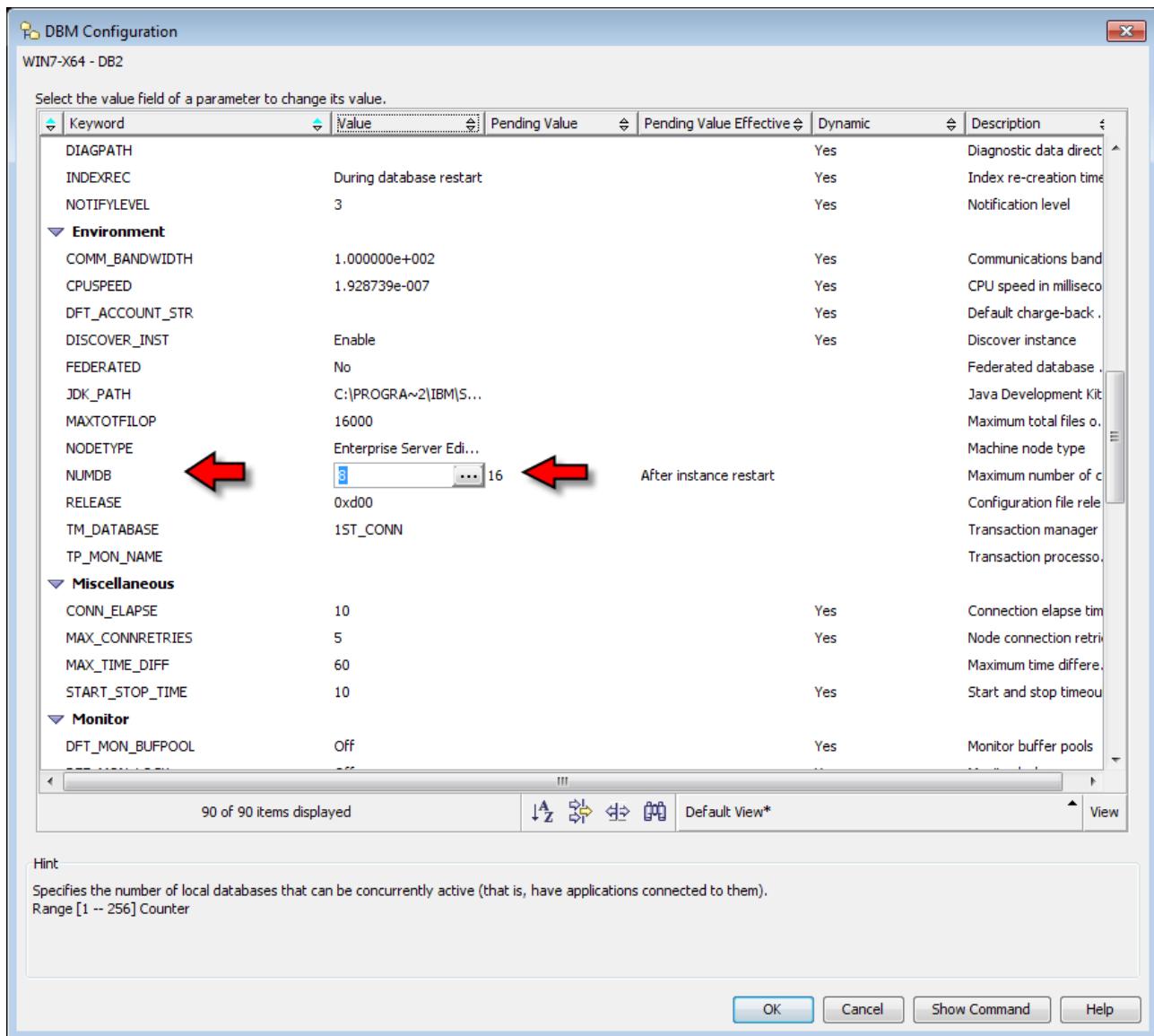
If things start to fail and we get error code SQLSTATE 57032, this means that we have too many databases open. Using the db2 get dbm config command, look for the entry called:

Max number of concurrently active databases (NUMDB)

The default appears to be 8.

The entry can be changed in the DB2 configuration





After making this change, DB2 must be stopped and restarted which means stopping all the servers too.

# Eclipse

The Open Source Eclipse project can provide a wealth of tools that can be used in conjunction with IBM BPM:

- [soapUI - http://www.soapui.org/eclipse/update/site.xml](http://www.soapui.org/eclipse/update/site.xml)
- Java EE and Web Tools



The Java EE and Web Tools package is ideal for building Java EE WAR applications for deployment into a WAS server.

- [Class Locator Plugin](#)

# Unix Notes

## X-Windows

When working on Unix systems, it is sometimes useful to install an X-Windows server on a Windows environment. A good free version is called Xming:

<http://www.straightrunning.com/XmingNotes/>

When a client wishes to send an X-Window to an X-Server, the X-Server needs to allow the incoming client. For testing purposes, Xming can be started with the "-ac" flags which allow all connections, effectively disabling security. This is not recommended for a long term solution but useful in a demo or PoC environment.

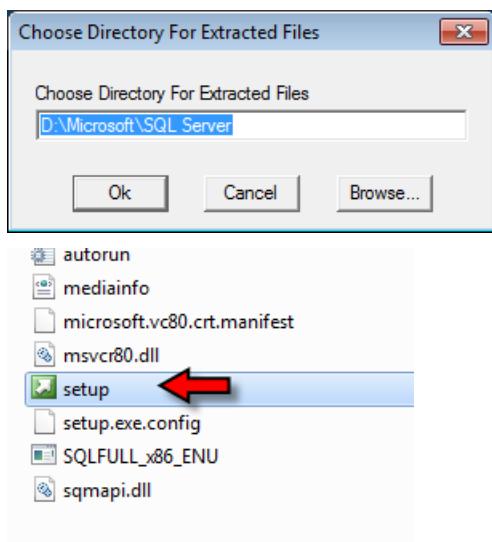
## Database Installation

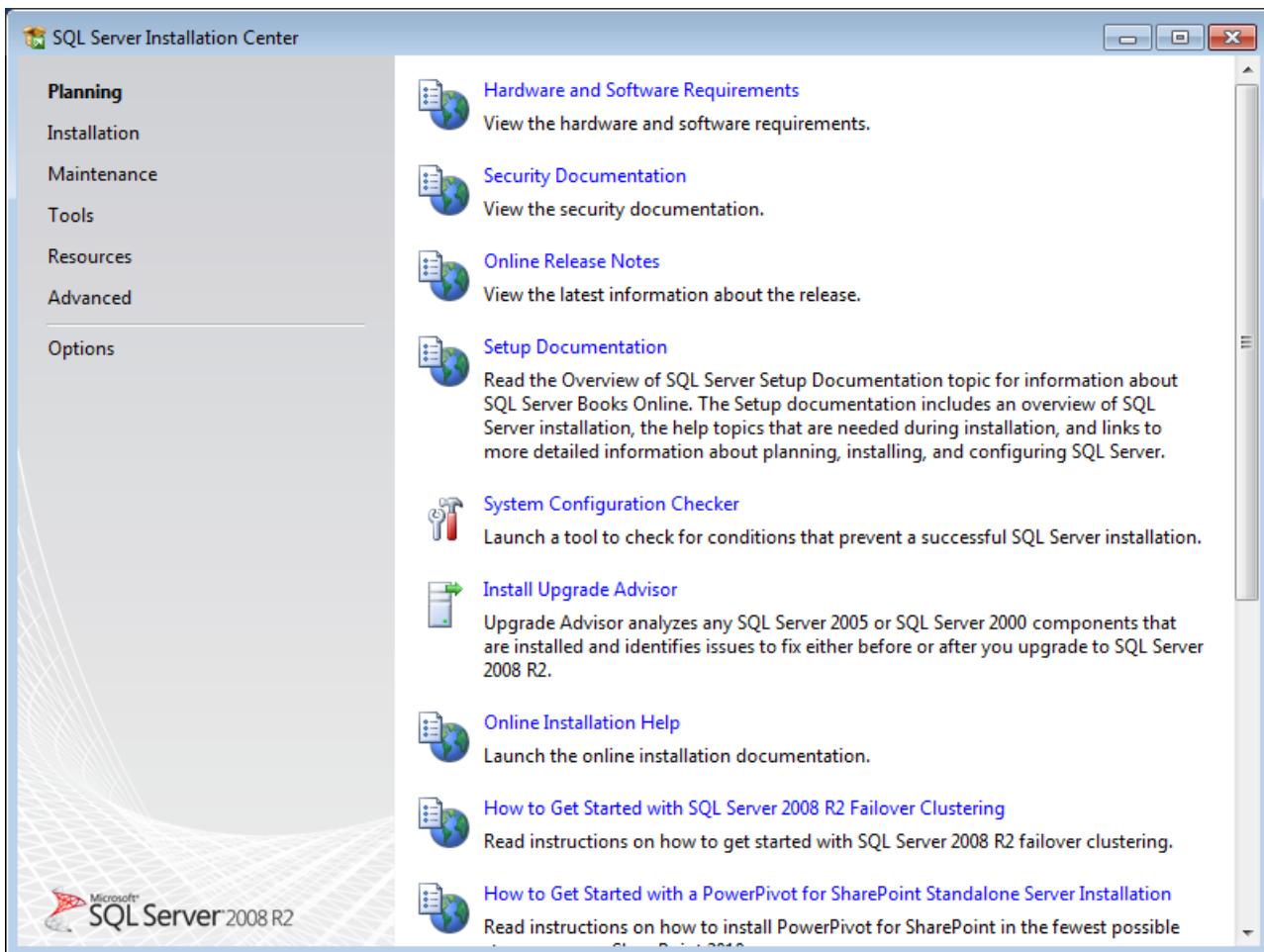
### MS SQL Server 2008 Installation

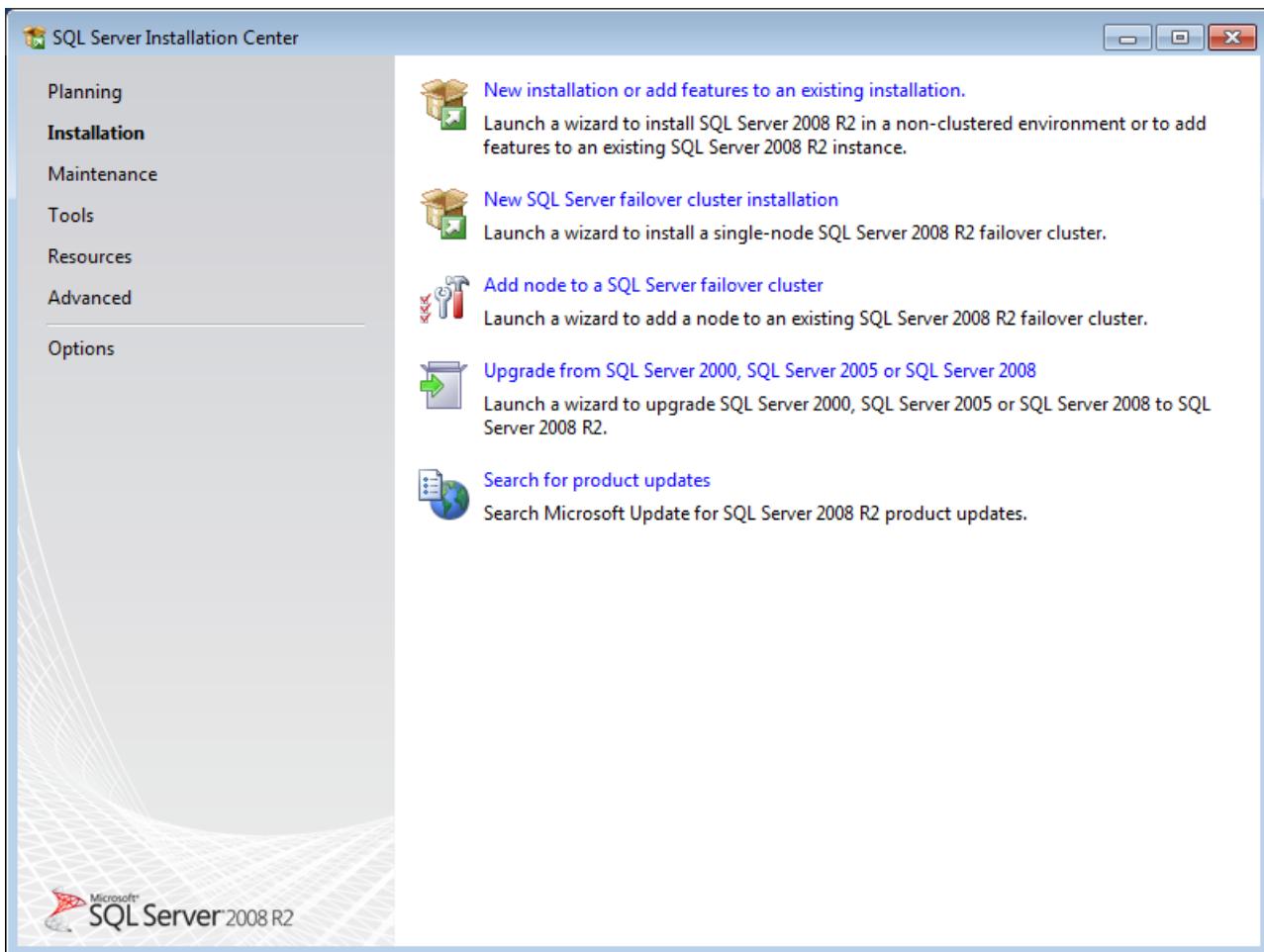
An evaluation copy of MS SQL Server 2008 was obtained from the Microsoft web site.

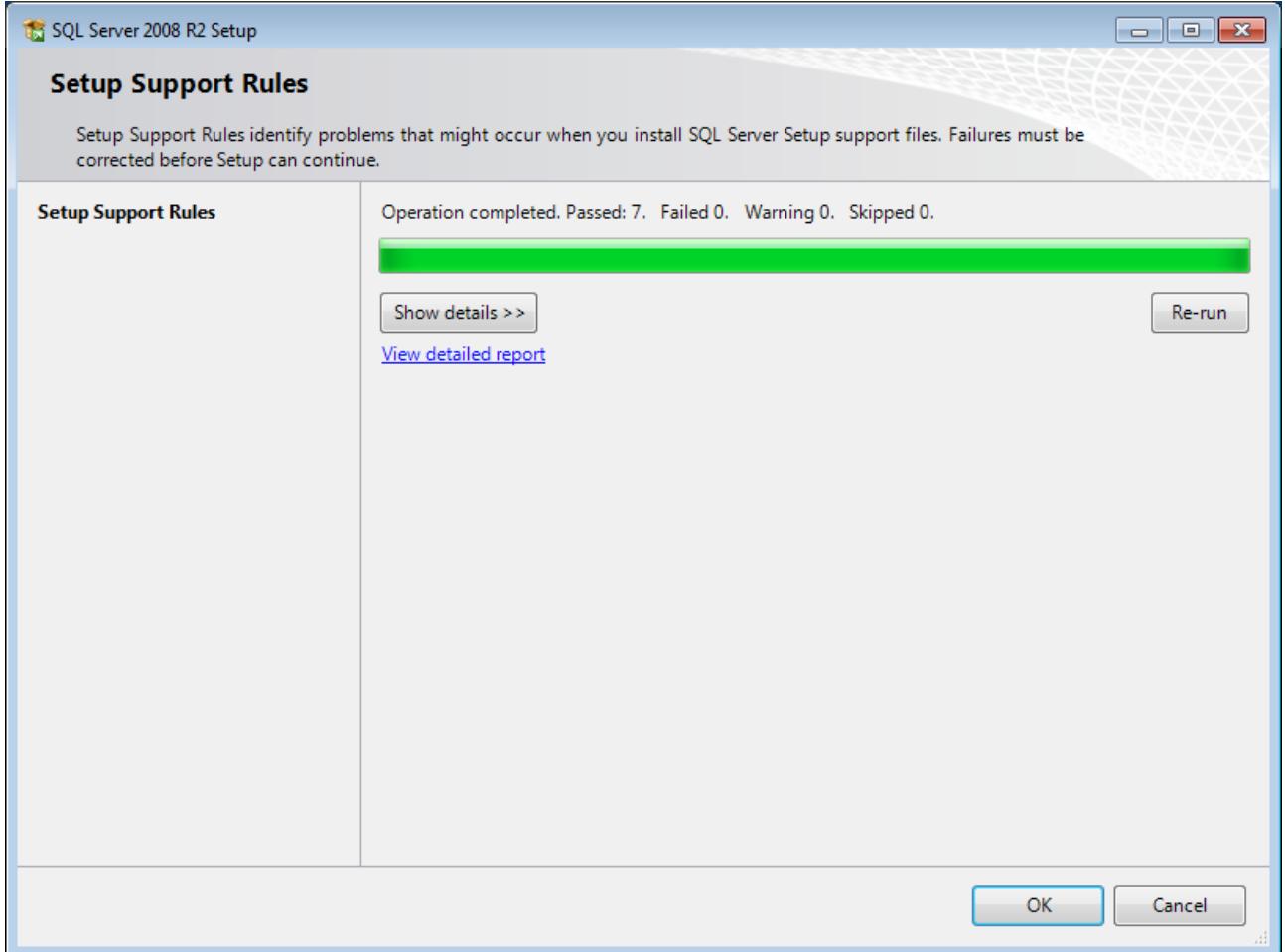
Name	Date modified	Type	Size
SQLFULL_x86_ENU	12/3/2011 9:29 AM	Application	1,374,266 KB

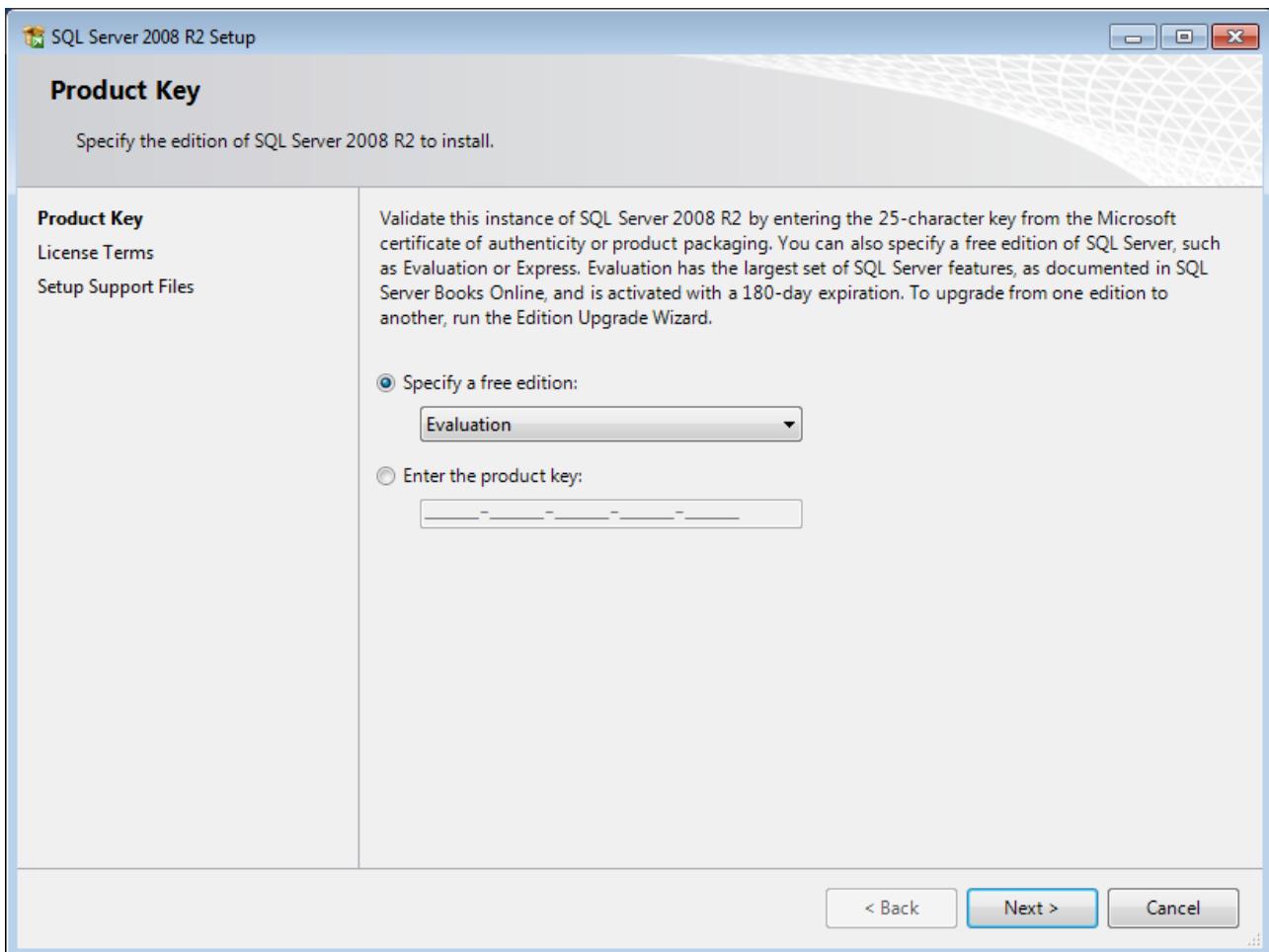
When executed, it prompted for a directory into which to extract the installation image.

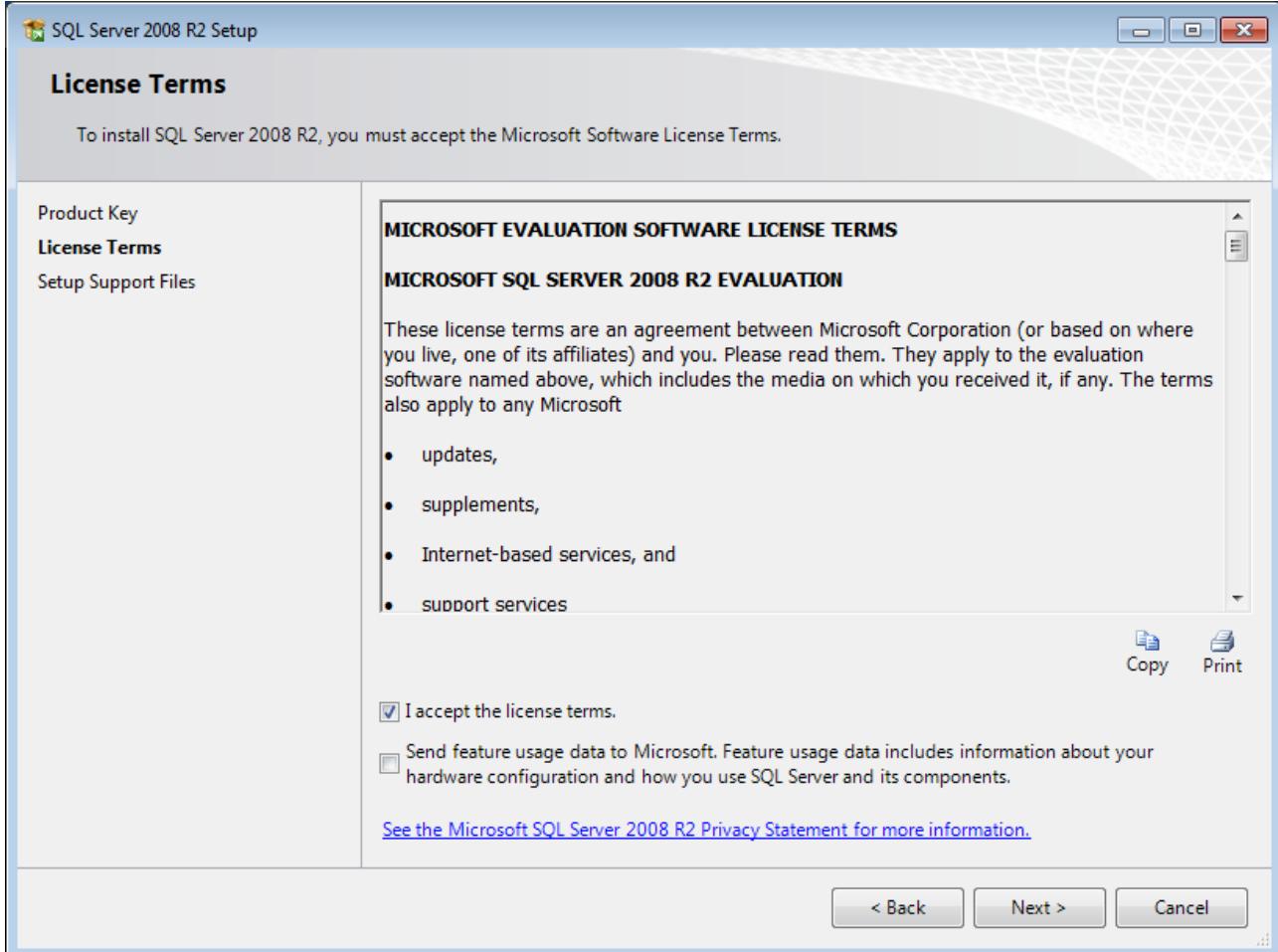


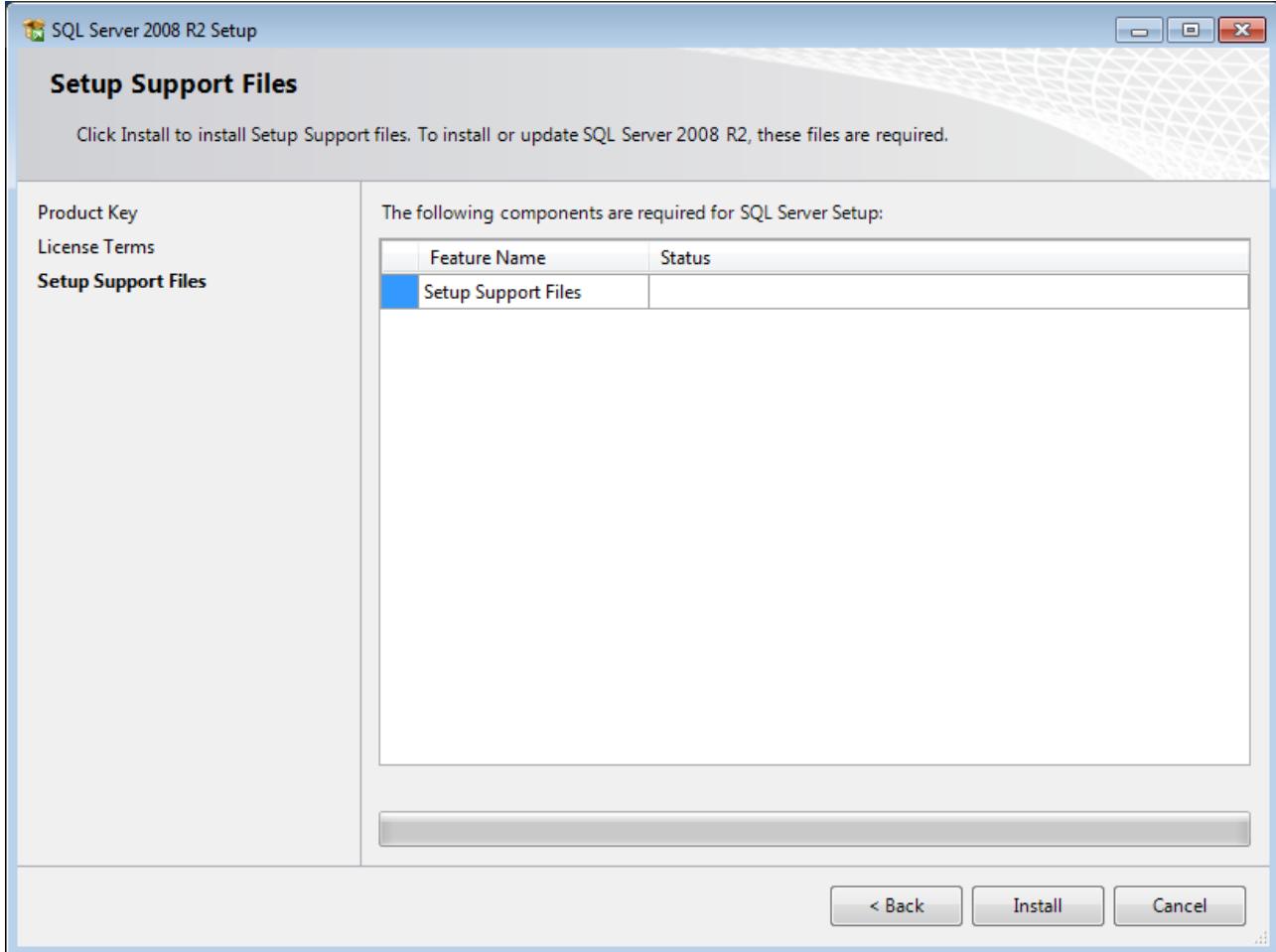


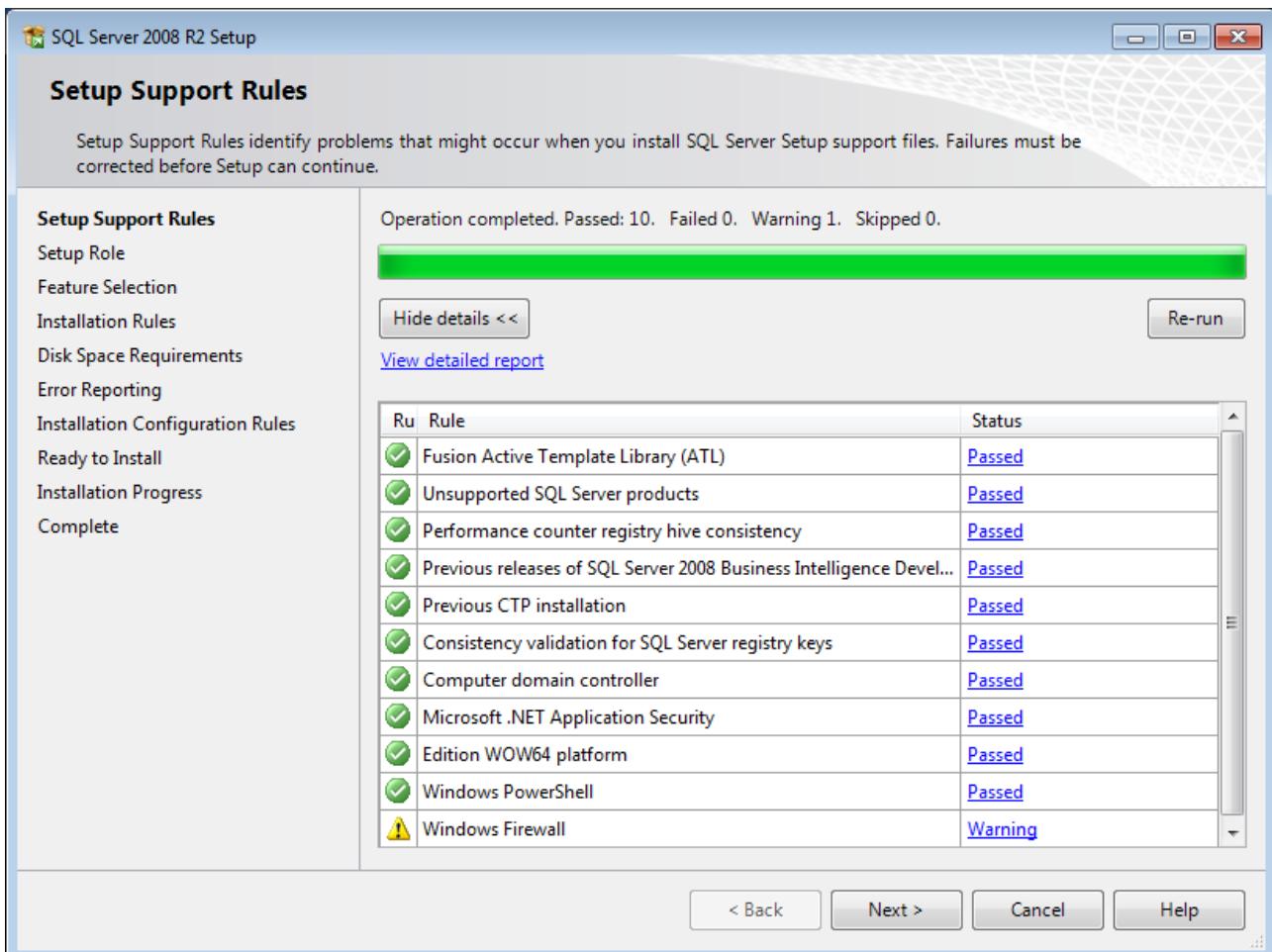


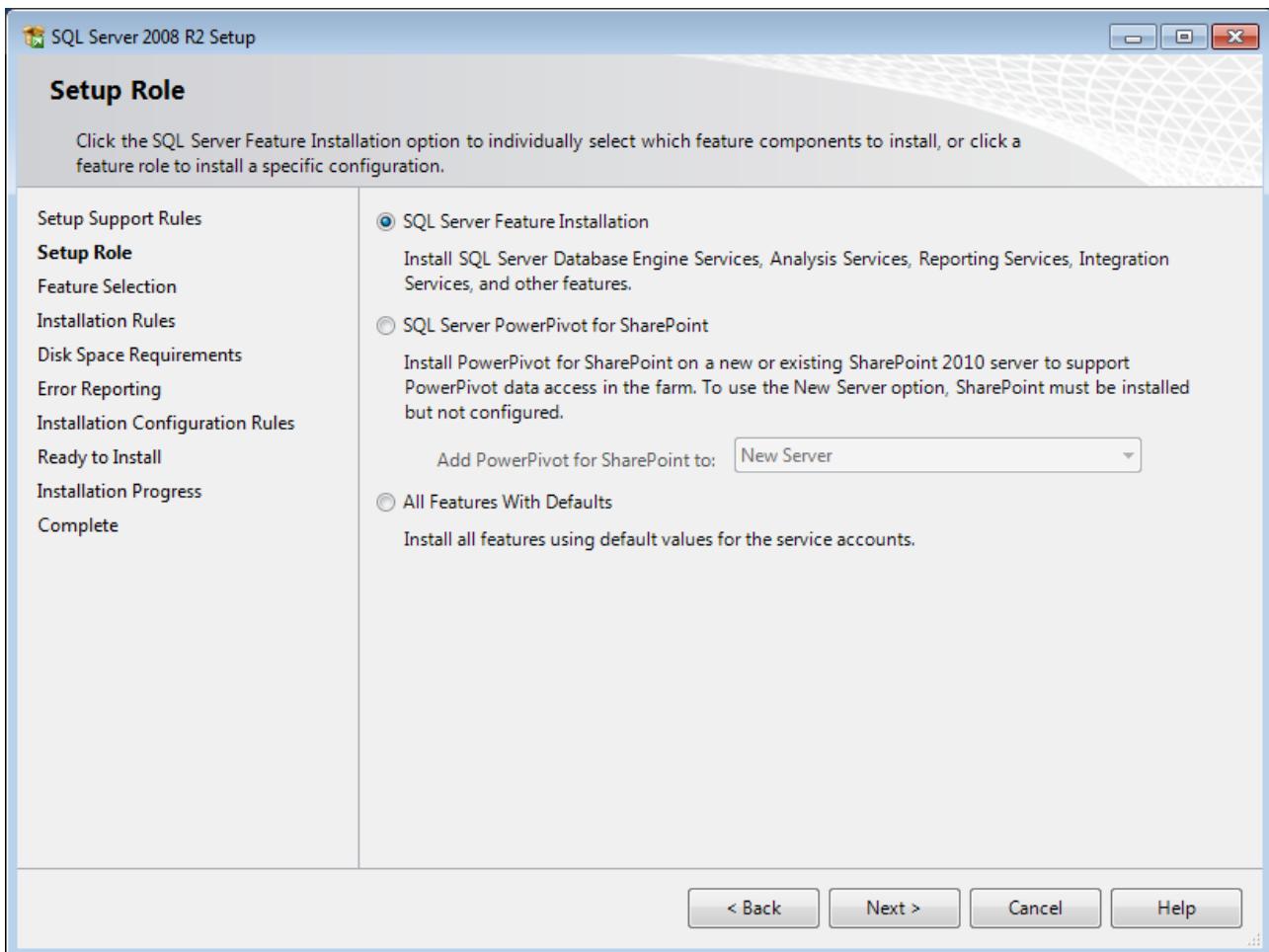


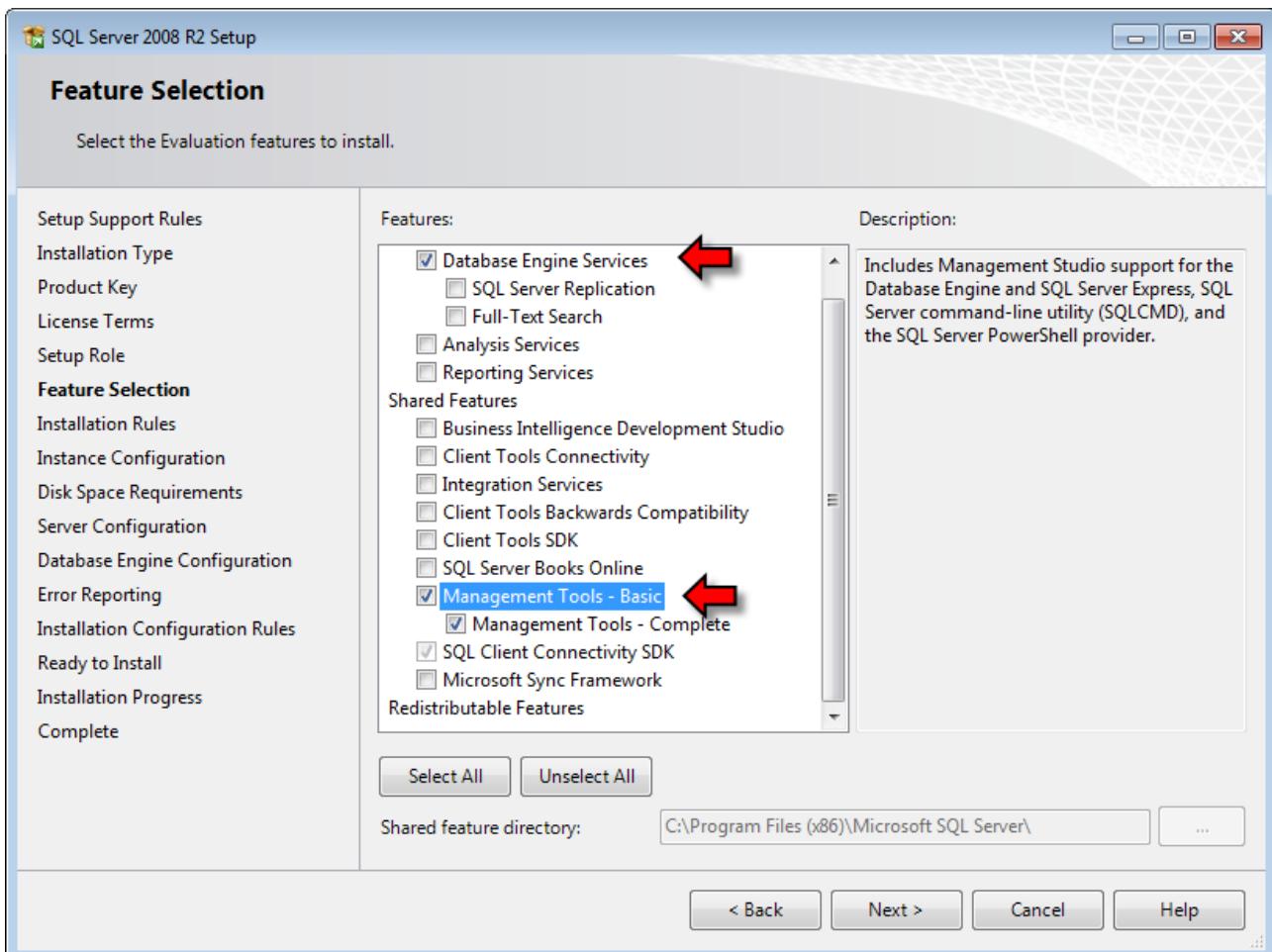


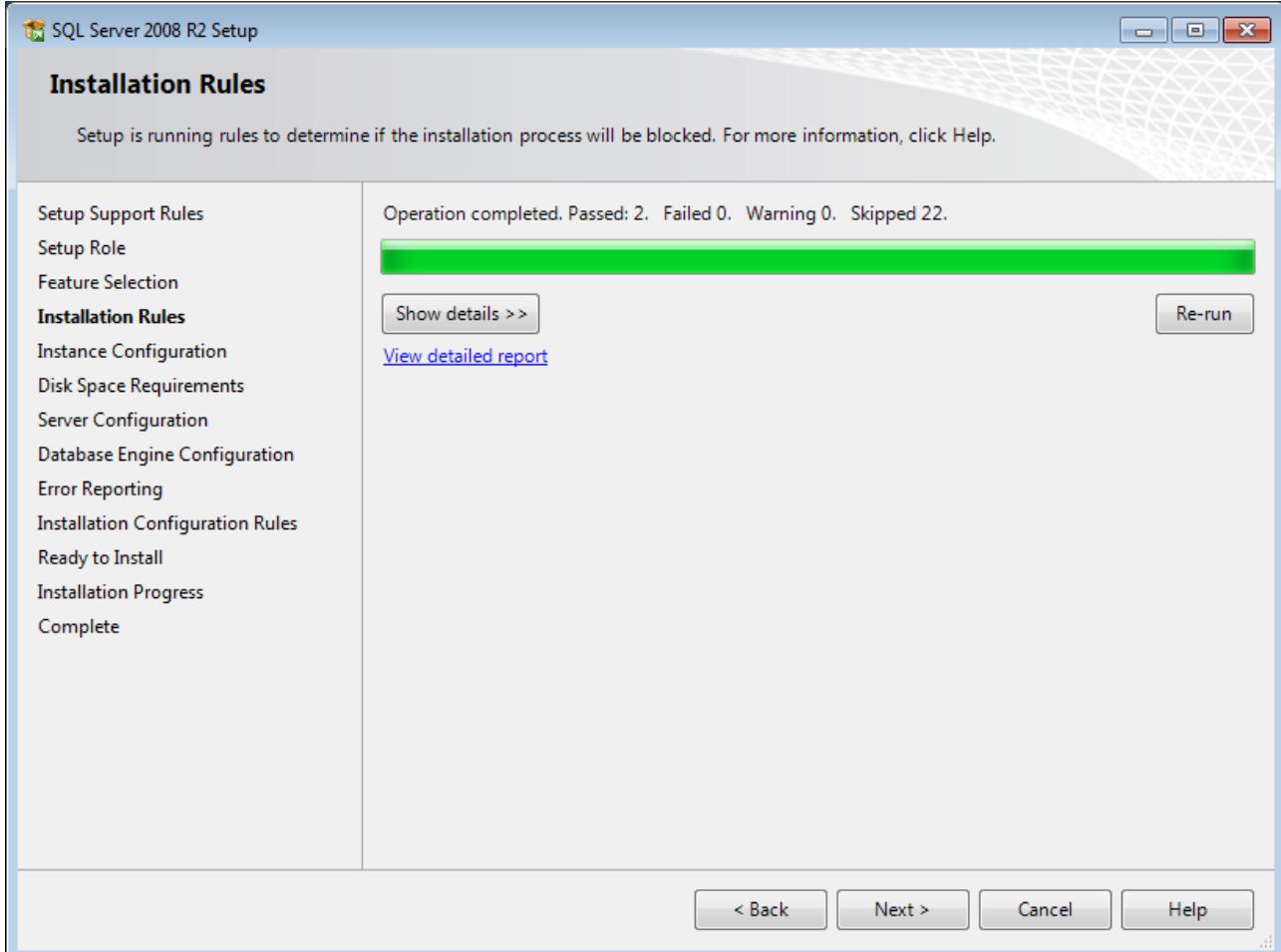


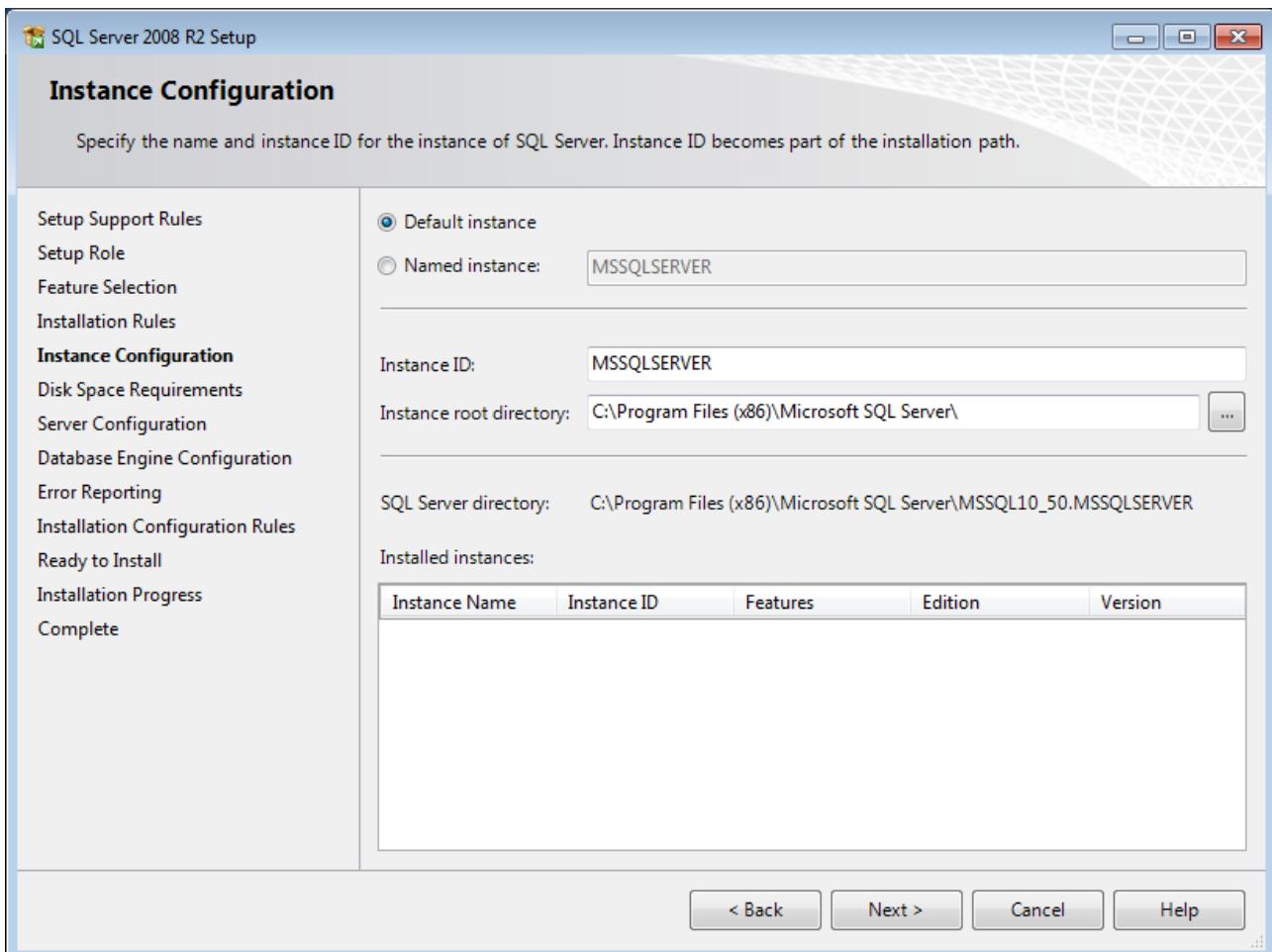


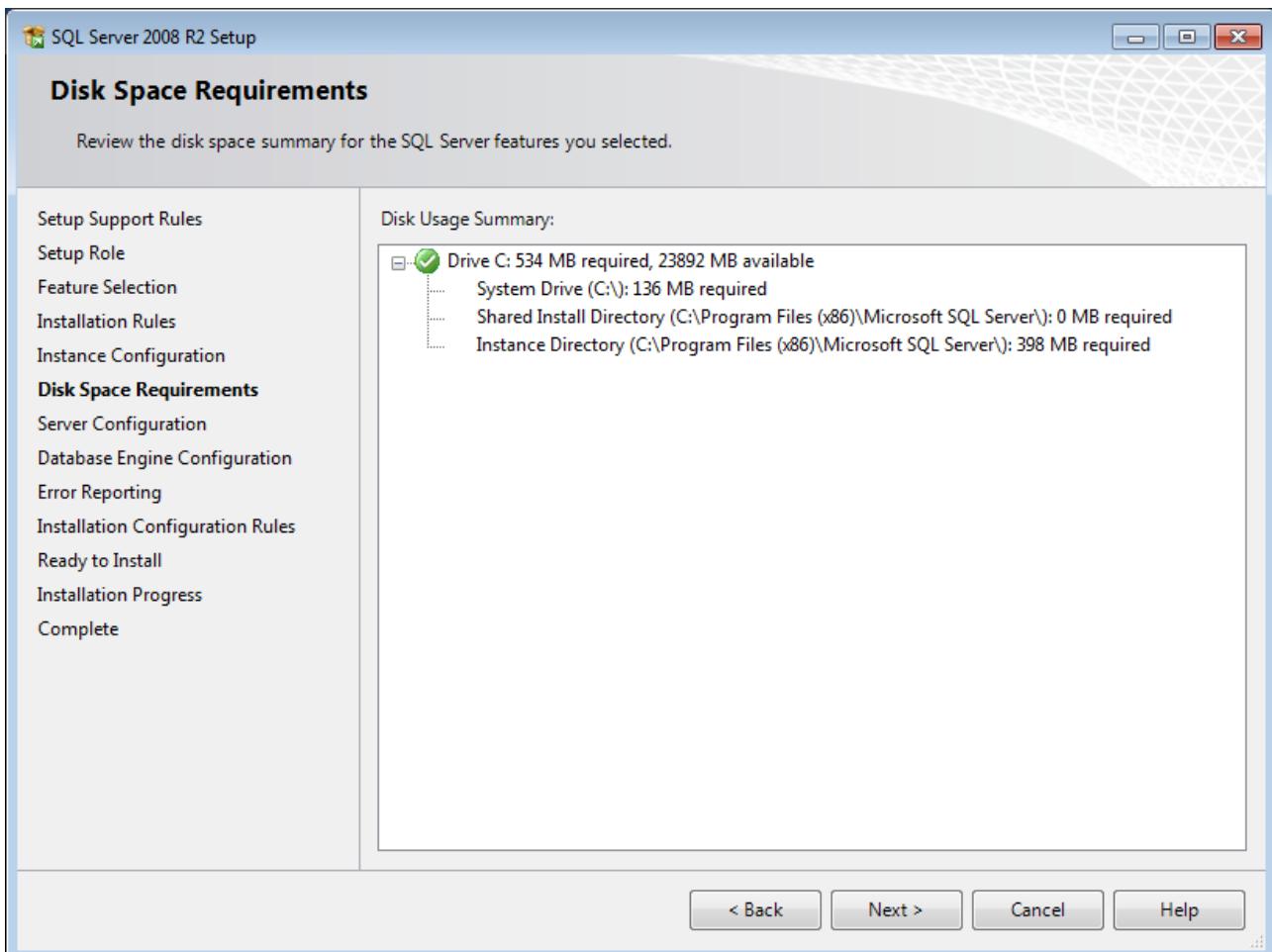


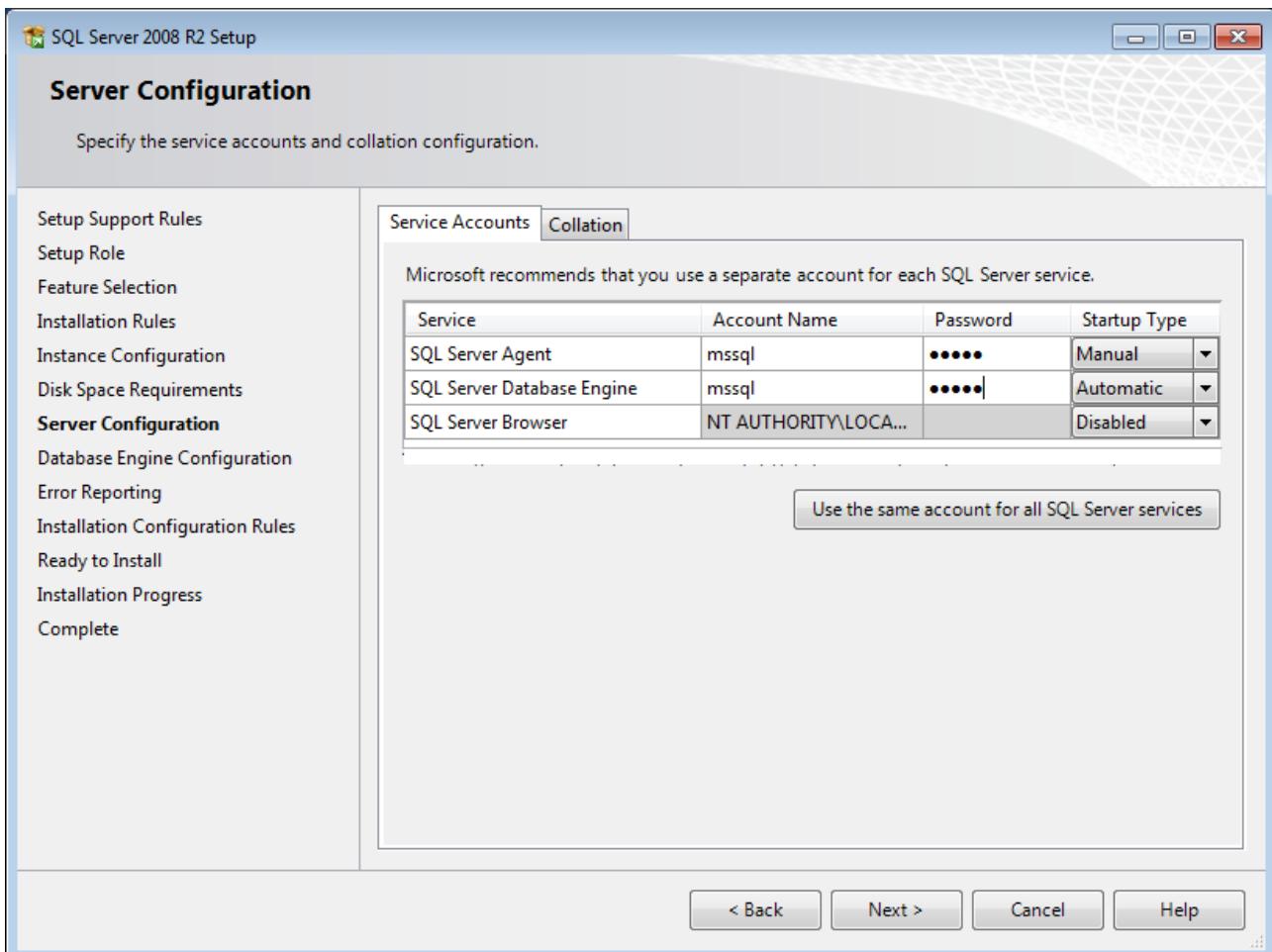


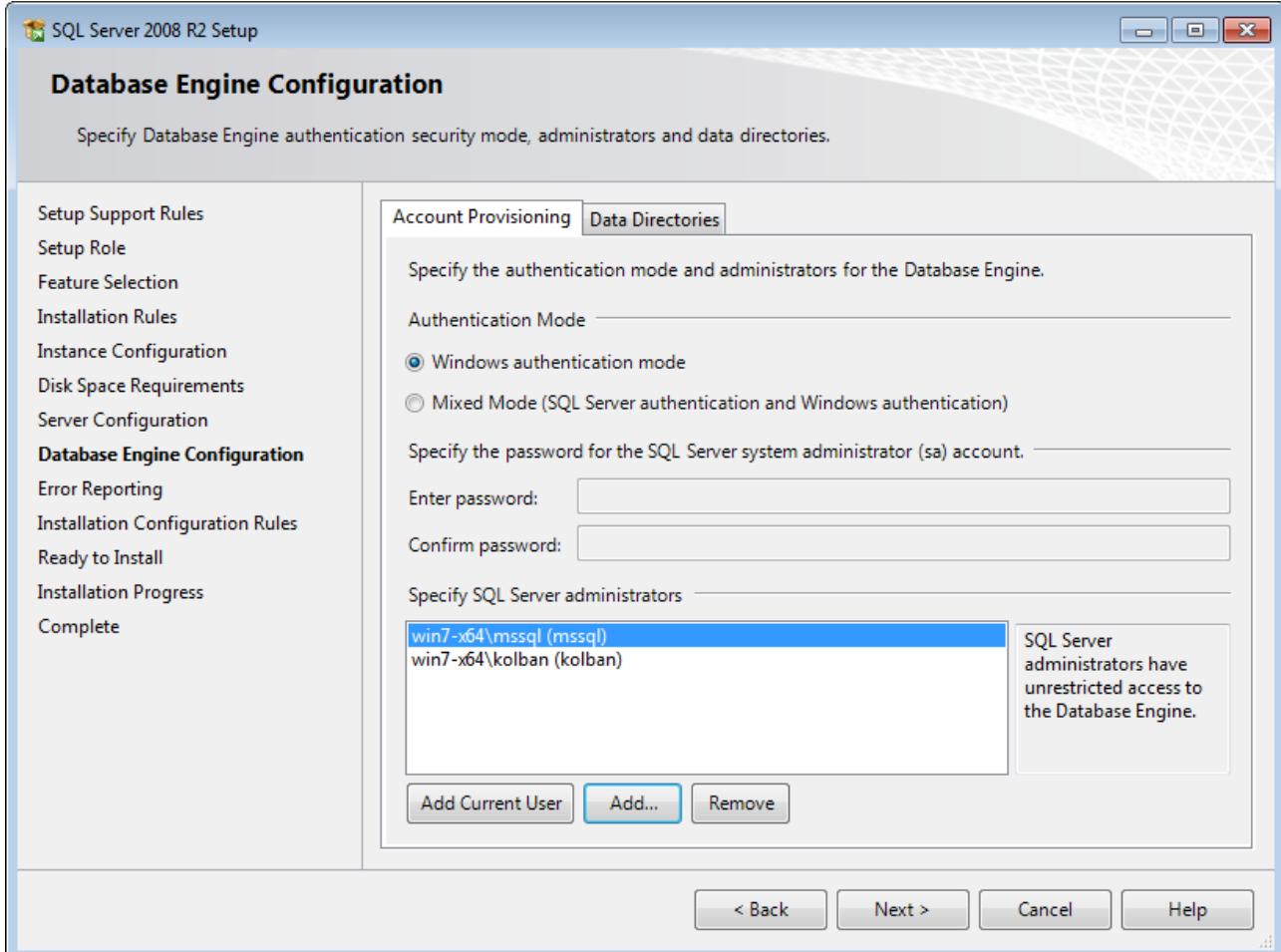


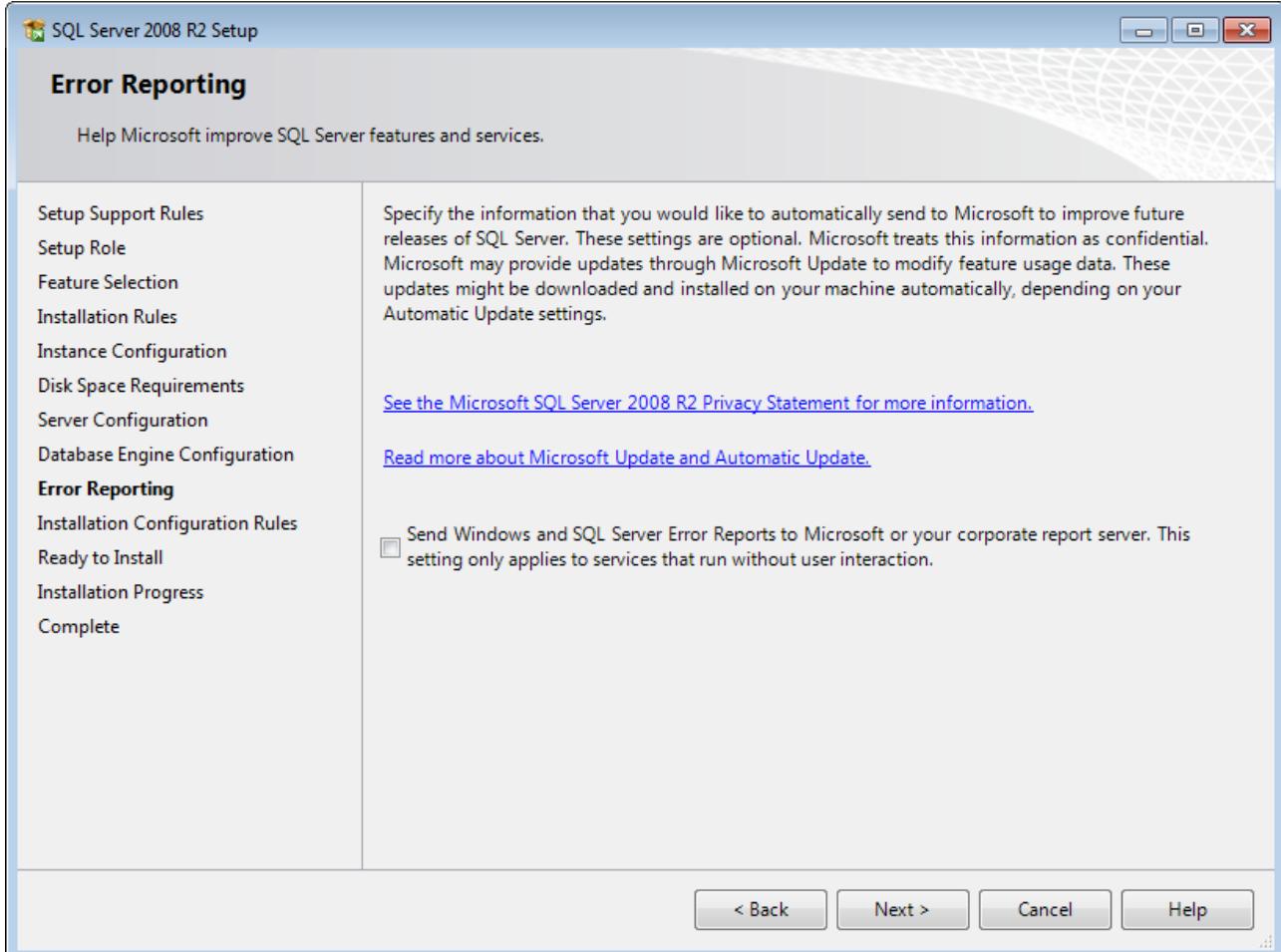


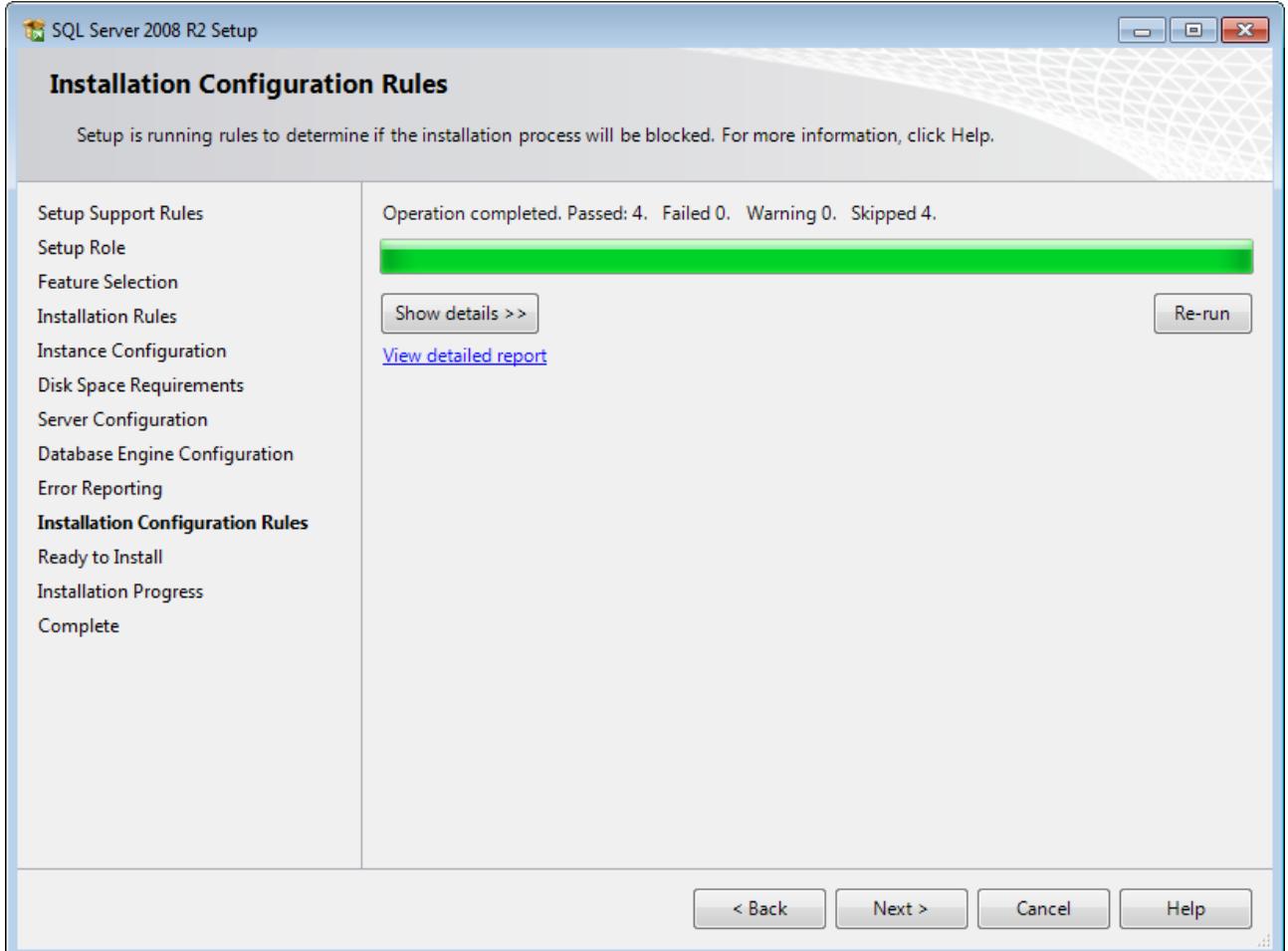


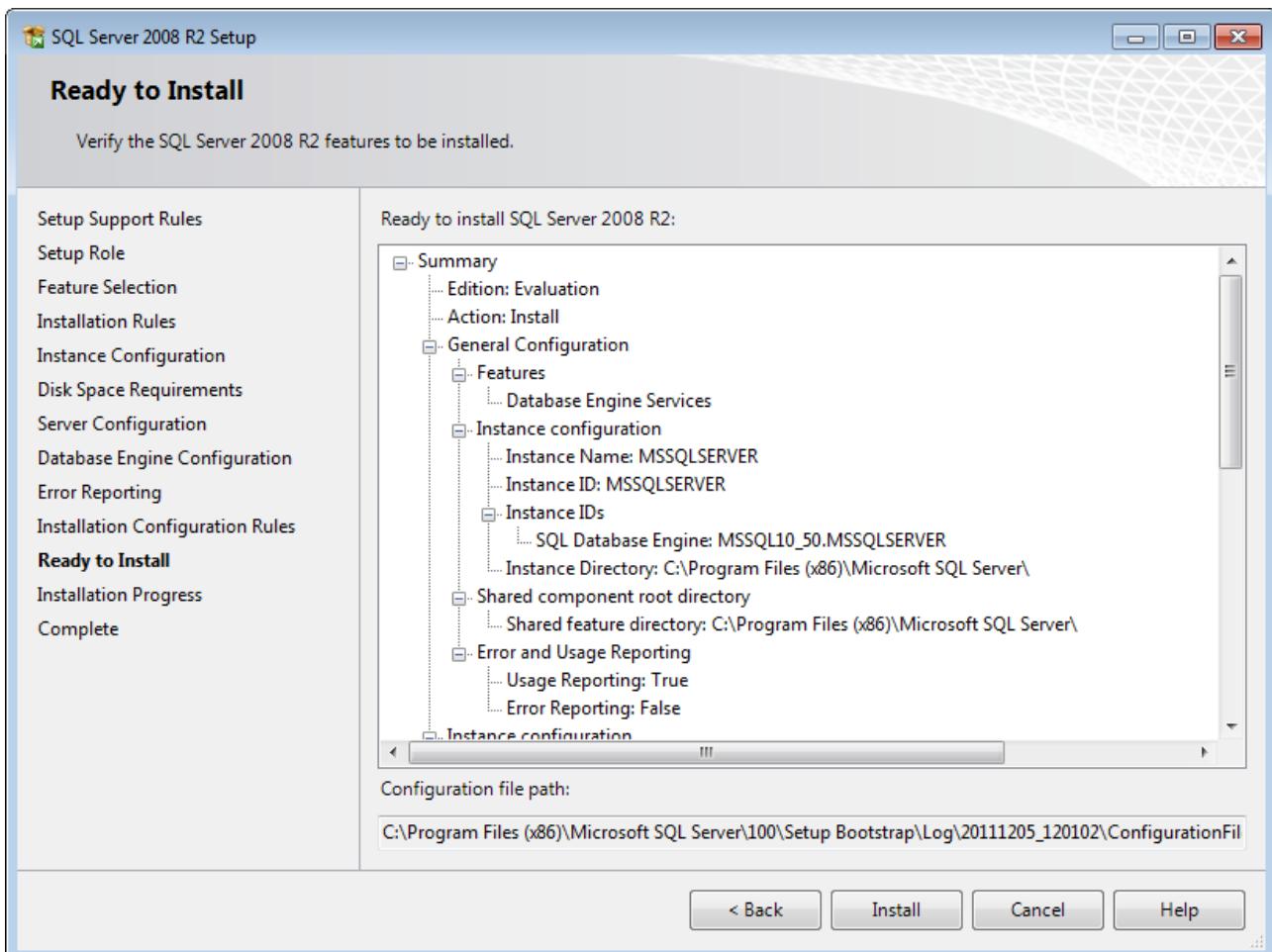


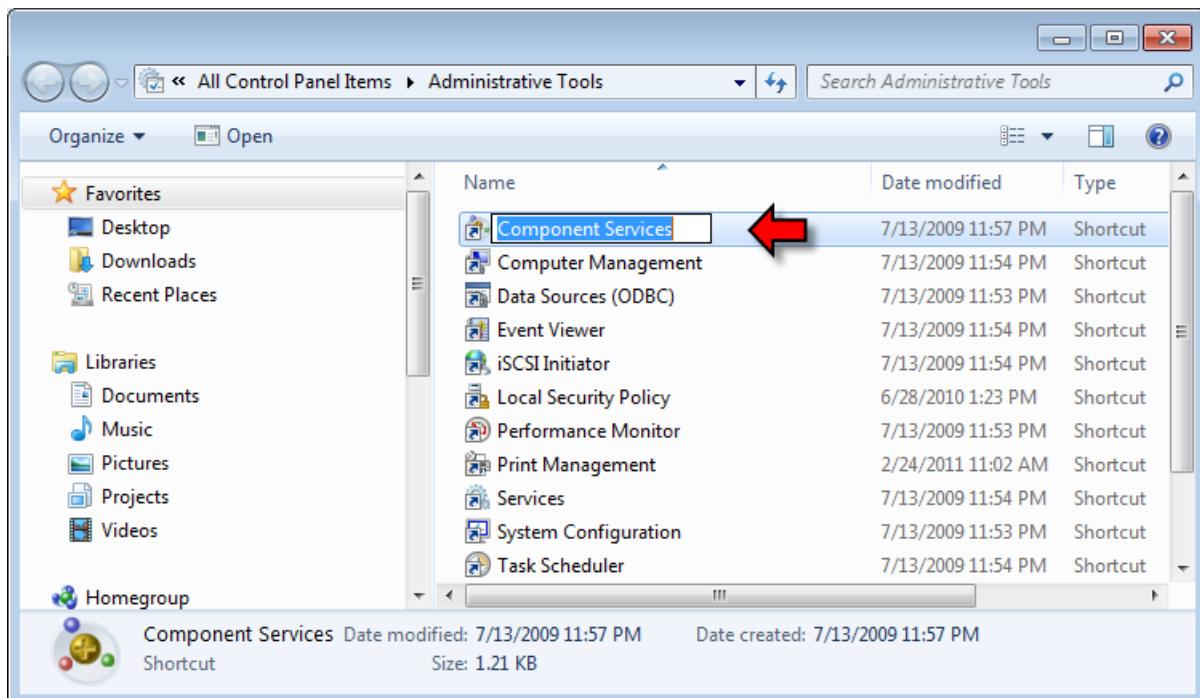
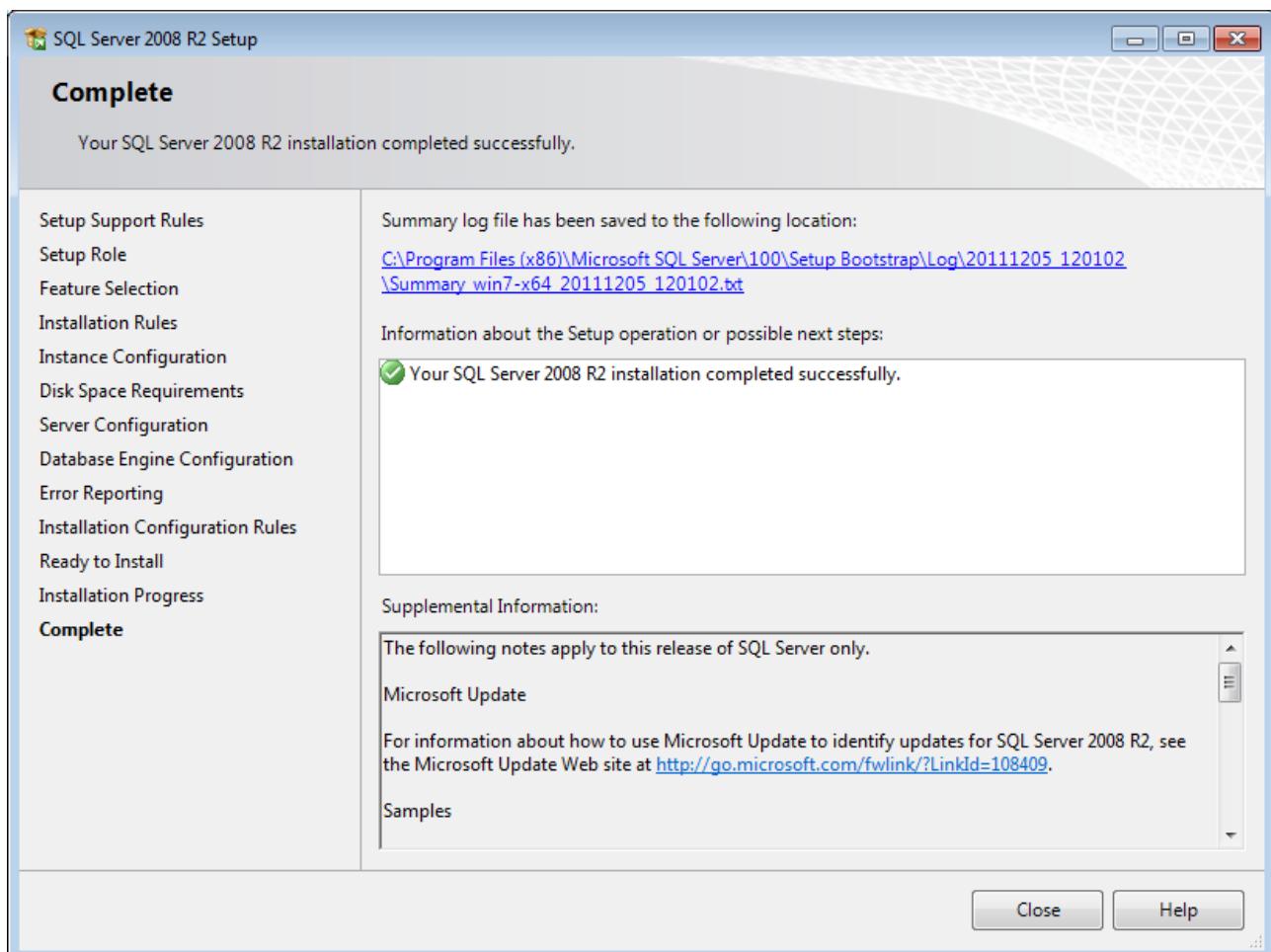


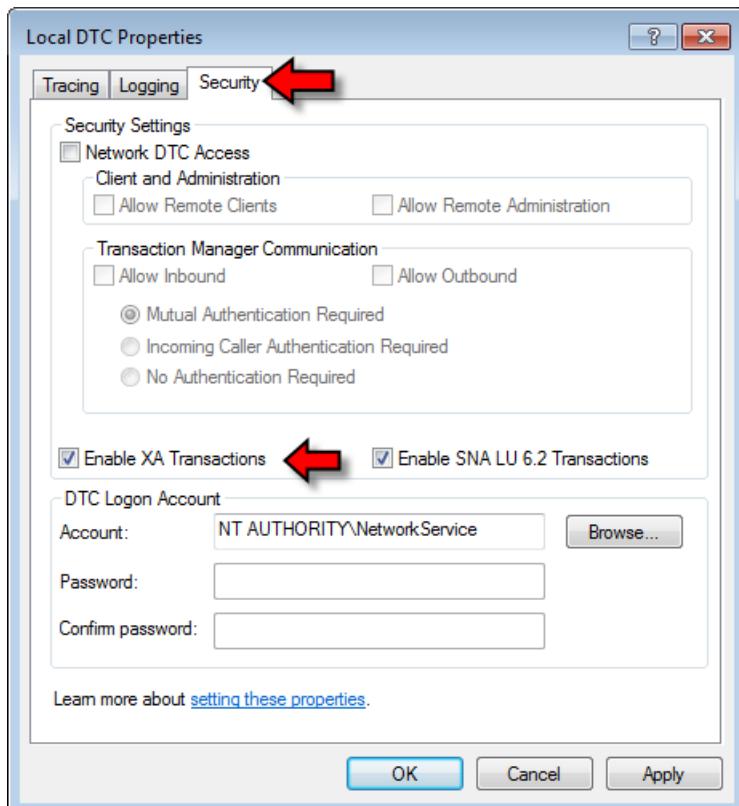
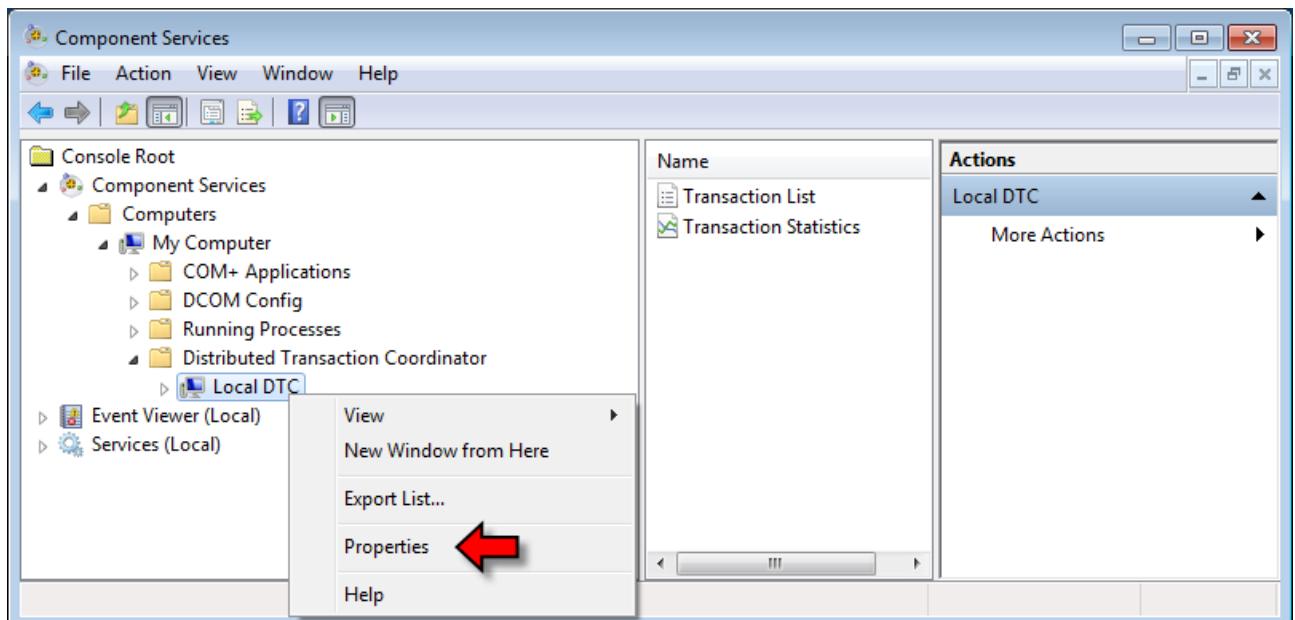






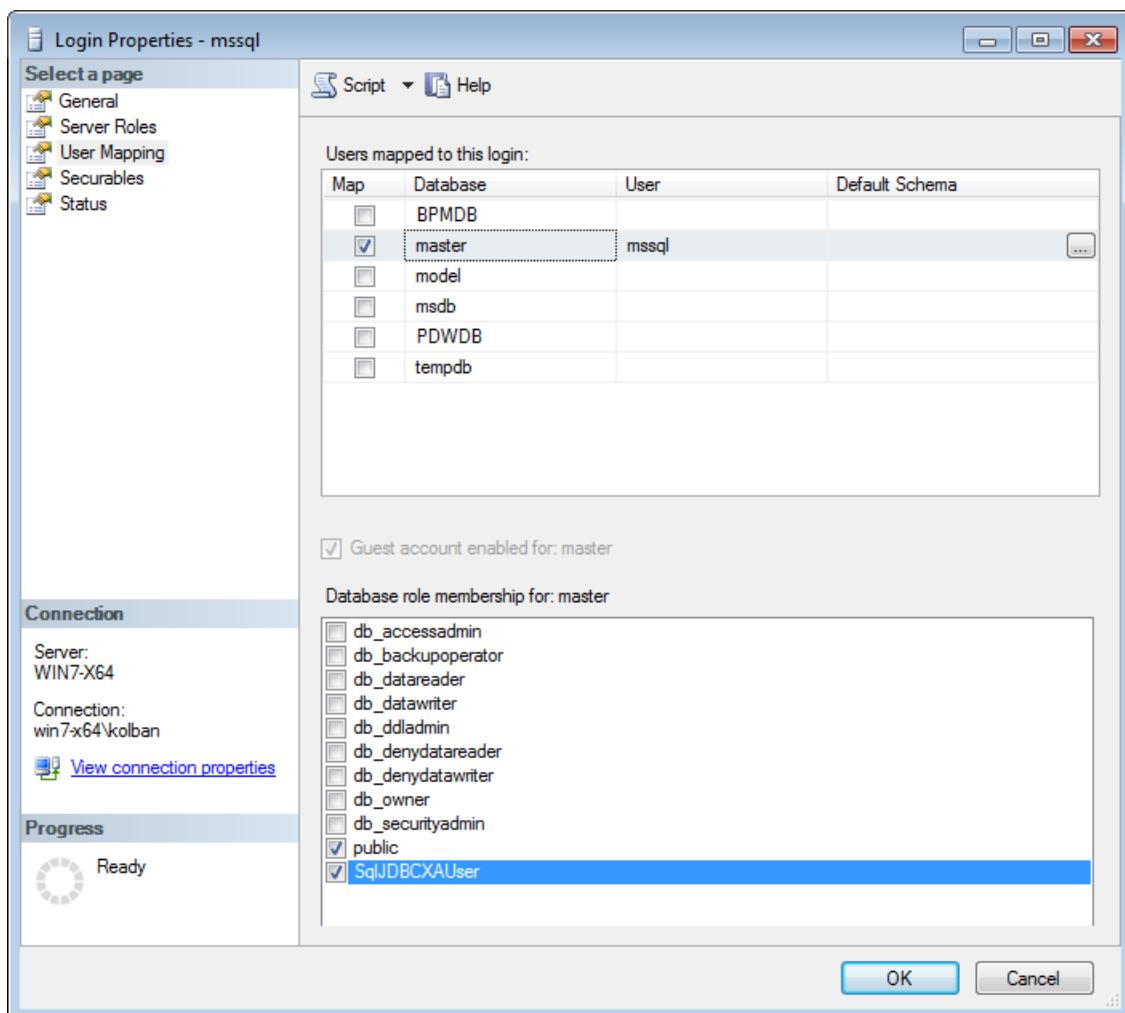






<http://www.microsoft.com/download/en/details.aspx?id=2505>

C:\Program Files (x86)\Microsoft SQL Server\MSSQL10\_50.MSSQLSERVER\MSSQL\Binn



# **Java Programming**

When working on a WebSphere Application Server environment, sooner or later some Java code will need to be written by someone. There are thousands and thousands of books and articles written on general Java and its use and this book will not be another. It simply would not belong in such a book. However, that said, there are sometimes Java oriented topics that may crop up in conjunction with IBPM that are of value. This section of the book is where these will be captured.

## ***Writing XML documents***

We all know what an XML document is and there is no need to go into that again. What if we wish to generate an XML document from Java code? There are many ways to achieve that task. This section talks about one one them ... the `javax.xml.stream` package.

First we will look at `javax.xml.stream.XMLOutputFactory`.

This has a static method called `newInstance()` which will return an instance of an `XMLOutputFactory`.

Given one of these, we can now obtain an `XMLStreamWriter` object using the `XMLOutputFactory.createXMLStreamWriter()` method.

Now that we have an `XMLStreamWriter` object, we can invoke its methods to write XML parts ... part by part.

## ***Working with Apache POI***

Apache POI is a set of Java libraries that provide access to Microsoft Office based documents including Excel and Word. The home page for this project can be found at:

<http://poi.apache.org/>

### **Reading an Excel Spreadsheet with POI**

### **Writing an Excel Spreadsheet with POI**

## Demo – Book Order Fulfillment

This demonstration is a comprehensive solution of a fictitious company (Books-2-U) that sells books over the internet. The scenario was chosen not to represent any particular industry but rather to be illustrative of how to use the IBM BPM product to solve problems using a story that should be familiar to all readers.

Most of us will have ordered books over the internet by now. The usual pattern is that we visit a web site, browse or search for books, create an order for the books we wish, provide shipping and billing information and then hit the submit button. Within a few weeks, a box arrives on our doorstep containing the books we ordered.



What this story considers is what happens when an order arrives at our Books-2-U company?

# Books-2-U

What do we have to do to fulfill our customer's order?

!!

Think through what you might steps you might imagine to be involved in fulfilling orders before you continue. See if what you imagined is close to what we imagined. There isn't necessarily a right or wrong answer.

When designing a solution, gather your staff together and have them start to describe what happens today. A process capturing tool such as IBM's Blueworks Live might be a good candidate for this step but if that is not available, consider a white board or a generic drawing package such as

Microsoft's Visio.

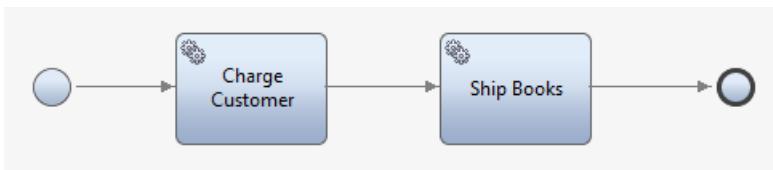
After a short while, we came up with the following:

- Receive the order
- Charge the customer
- Ship the books

This is about as simple and understandable as it can get. Much later on we will find that there is more to the story than this, but for now, let us assume this is what we have.

## ***Part I – A simple start***

Now we create a Business Process Definition (BPD) called *Book Order Fulfillment* that looks as follows:



These activities mapped exactly from our thinking of our process. We can easily see the flow of work. First we charge the customer and then we ship the books. That was pretty easy. Our thinking here is that if we now implement the two activities, we will have completed our task.

The next step we need to consider is the data associated with our process. All processes have data associated with them. They have to. It is the difference in data content that will differentiate one instance of a process from another. In our scenario, what data items are we working with?

After some thought, we decided on the following:

- A Book
  - Title
  - Author
  - Price
- Shipping
  - Recipient name
  - Address
- Billing
  - Card number
  - Card expiry
  - Card holder name
  - Billing address
- An Order
  - A list of Books
  - Billing details

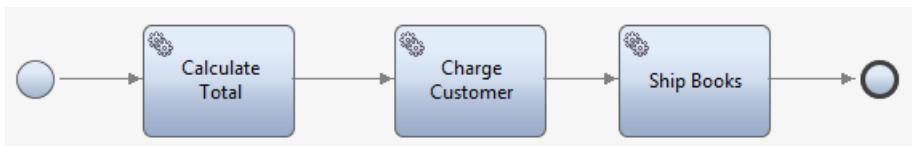
- Shipping details

The last data type (order) is the core. An order is considered to be a list of books desired by the customer plus the details of how they are going to pay and finally to where they want the books delivered.

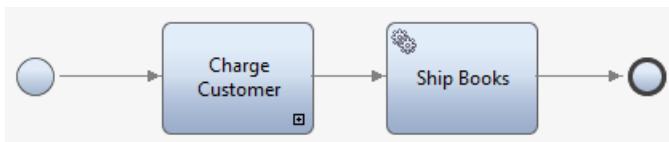
By having made this list, we can again map this to capabilities of the BPM product. The following BPM Business Object data types were built. Business Objects are data types that are created in the Process Designer.

Book	Address	ShippingDetails
• title (String)	• street (String)	• recipientName (String)
• author (String)	• city (String)	• address (Address)
• price (Decimal)	• state (String)	
	• zip (String)	
BillingDetails	Order	
• cardNumber (String)	• books (Book) (List)	
• cardExpiry (String)	• billingDetails (BillingDetails)	
• cardHolderName (String)	• shippingDetails (ShippingDetails)	
• address (Address)		

Now we can start looking at the implementation of the activities. The first activity we will examine is *Charge Customer*. This activity is meant to charge an amount against a customer's credit card. We will assume that this is a new implementation and we don't have an existing similar service. Thinking about what such a service needs in order to perform its work, we determine that it needs the amount of money to bill plus the details of where to bill via a credit card. Looking at the data already available to us, we find that we do *not* have a total amount to charge. Instead we have an order which contains a list of items where each item has an associated price. This means that we would like to calculate the total order amount. However, adding a technical step to perform that task would seem to clutter our process diagram. What we do **not** want to end up with is the following:

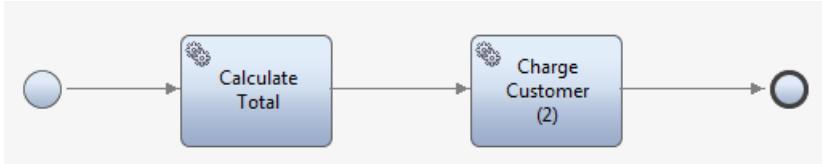


This intermixes clean and high level business steps with technical necessities which is not what we want. A solution to this type of issue is the creation of a sub-process which will encapsulate technical steps while leaving the top level BPD pristine:



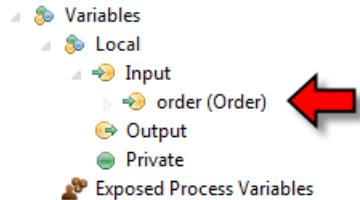
A sub-process is a sequence of BPD activities that are contained within the parent BPD but which are not immediately visible. Think of a sub-process as a "box" that owns other activities. Notice in the diagram above that the *Charge Customer* activity has a small icon in its lower right that indicates that it is a sub-process holder.

Notice that the *Charge Customer* step was changed from a system task to a sub-process. Drilling down into the sub-process, we can now define the following steps within that sub-process:

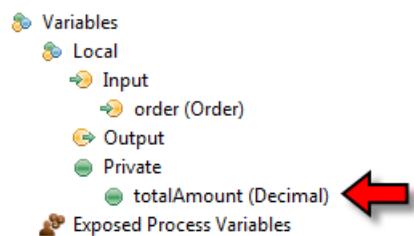


The true benefit of this is that we maintain our high level story while at the same time defining the next level down to have more details.

So far we have not defined any variables in our process. Thinking about its nature, we see that all process instances start with the receipt of an `Order`. We model this by defining an input variable on the process. This variable is called `order` of type `Order`.



Now we turn our attention to calculating the total amount to be charged against the customers credit card. The `Order` contains a list of books the customer wishes to purchase and each book has a price associated with. We define a local variable to the sub-process called `totalAmount` of type `Decimal`.



Note that because this is defined as private on the sub-process, it will not be visible on the parent process. This is goodness as it is only needed locally to the sub-process and doesn't dirty our parent process space of variables.

The implementation of the *Calculate Total* activity will be an in-line script. We could have chosen to create a separate service but since there appears to be no other obvious consumers, we don't see a need for this to be implemented in a re-usable fashion. The script itself is a simple loop over the books in the order.

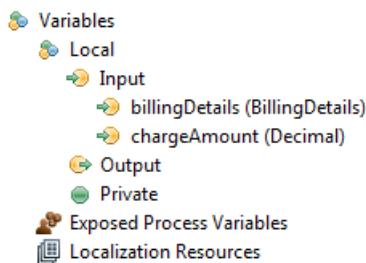
```

tw.local.totalAmount = 0;
for (var i=0; i<tw.local.order.books.listLength; i++)
{
 tw.local.totalAmount =
 tw.local.totalAmount + tw.local.order.books[i].price;
}

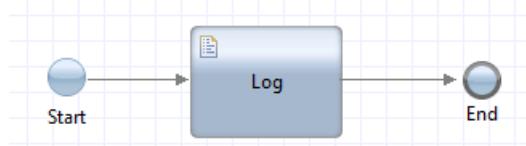
```

Making a charge against the customer's credit card will be a service. As a current place holder, we will define a general service to achieve that task but first we must decide what its input will be. To make a charge against the customer, we need the customer's billing details as well as the amount to charge.

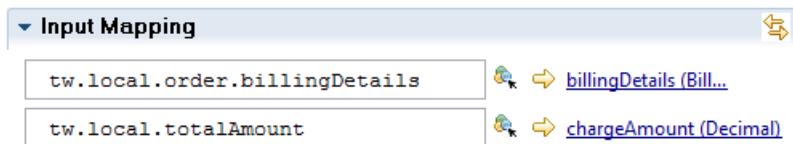
The service we create is called *Charge Card* and has the following interface:



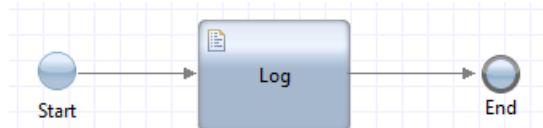
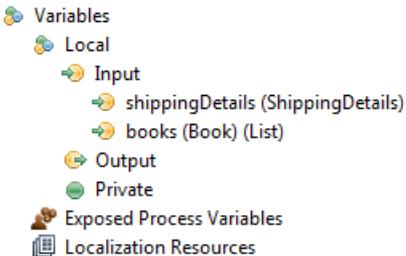
For now, its implementation is nothing more than a log that it was called:



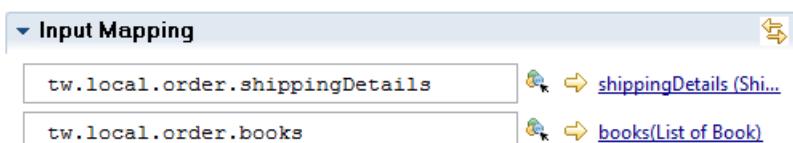
In the BPD, the activity called *Charge Customer (2)* is implemented using the *Charge Card* general service and the data mappings mapped as follows:



Now we return our attention to the activity called *Ship Books*. Again, this will be a technical service that will cause the books to be shipped. For now we will create a general service to implement this step. We will call this general service *Ship Books*. Its interface will look like:



The *Ship Books* BPD activity will be implemented by the *Ship Books* general service. The mapping from the variables to the input parameters will look as follows:



We are close to completing the first pass of our initial build. We see that the process needs some initial input in order to run. It is extremely good practice to create some default content for input variables for services and processes. This way we can unit test each one easily. When creating default/test content, always make sure that it looks like test content and couldn't accidentally be used as real content that would cause confusion in the future. At this point, defining some sample data for the order variable will be needed for testing.

**Variables**

- Variables
- Local
- Input
  - order (Order)
- Output
- Private
- Exposed Process Variables

**Details**

Name: order  
Documentation: (Edit)  
Is List:   
Variable Type: Order [Select...](#) [New ...](#)

**Default Value**

Has Default:

```

1var autoObject = new tw.object.Order();
2autoObject.books = new tw.object.arrayOf.Book
3autoObject.books[0] = new tw.object.Book();
4autoObject.books[0].price = 12.34;
5autoObject.books[0].author = "Terry Pratchet"
6autoObject.books[0].title = "Sample - The col
7autoObject.billingDetails = new tw.object.Bil
8autoObject.billingDetails.cardHolderName = "J
9autoObject.billingDetails.cardExpiry = "2099-
10autoObject.billingDetails.cardNumber = "9999"
11autoObject.billingDetails.address = new tw.o

```

We are now able to run our process through Process Designer and look at its result in the inspector. After a run, we should see:

Process Instances		Services in Debug						
Instance Name	Snapshot	Process	Status	Owner	Subject	Priority	Due Date	Task Id
Book Order Fulfillment... Tip	Book Order	Closed	Closed	tw_admin	Step: Charge Customer (2)	Normal	Jul 2, 2012 10:5...	804
				tw_admin	Step: Ship Books	Normal	Jul 2, 2012 10:5...	805

In the WAS console, we should also find the information logged by our stub services:

```

>>> Charge Card
Charge amount: 12.34
--- Billing details
Card holder name: John Q Sample
Card number: 9999
Card expiry: 2099-09-09
- Address
Street: 123 Anystreet
City: Anytown
State: AA
Zip: 9999
<<< Charge Card
>>> Ship Books
--- Books
Sample - The color of magic, Terry Pratchet
--- Shipping details
Street: 123 Anystreet
City: Anytown
State: AA
Zip: 9999
<<< Ship Books

```

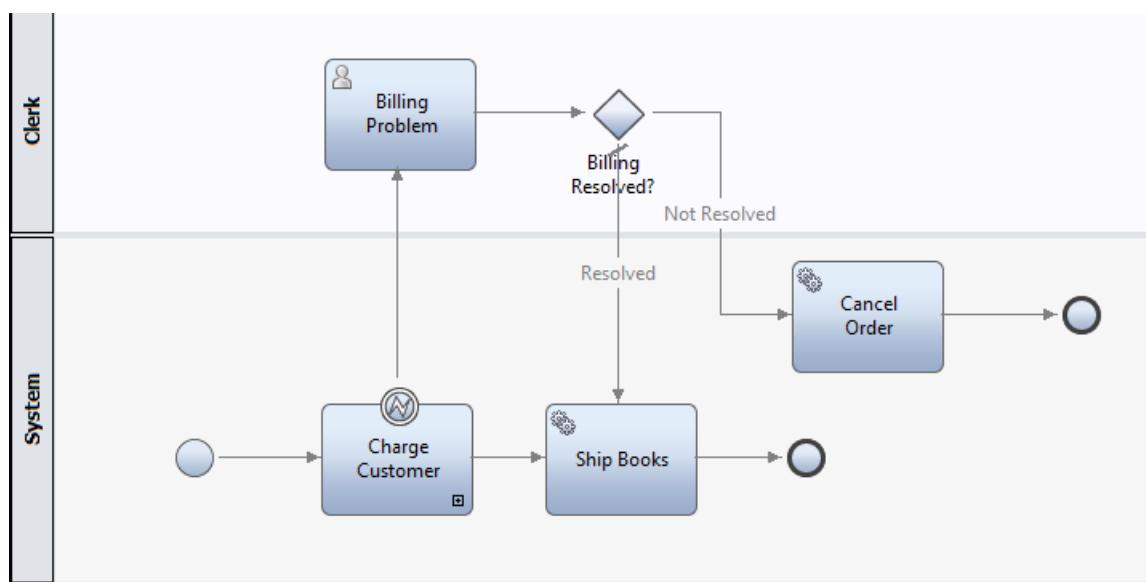
This completes our first pass at a basic process.

## Part II – Billing problems

Our solution so far seems to work nicely but is about as trivial as it gets. Unfortunately, really simple scenarios rarely reflect real world situations. Let us imagine that we put our process into production and almost immediately we find our first problem. If a customer orders books and they provide in-valid billing information, the situation is not handled. At best, the process terminates and the order is abandoned. We think we can do better than this. If the billing fails, we want to try and re-contact the customer and inform them of the problem. It could be they made a typo when they entered their card number or they temporarily had insufficient funds. These are correctable situations and, if we take new billing information, we can recover the process. This benefits the customer as they will get their desired order and it also benefits us as we get the desired revenue.

When we detect a problem with an attempt to charge the customer's credit card, we will take an alternate path. We will ask one of our clerks to examine the order and contact the customer. This may resolve the issue. It may also result in the order being canceled if we don't hear from the customer or else they wish to explicitly cancel their order.

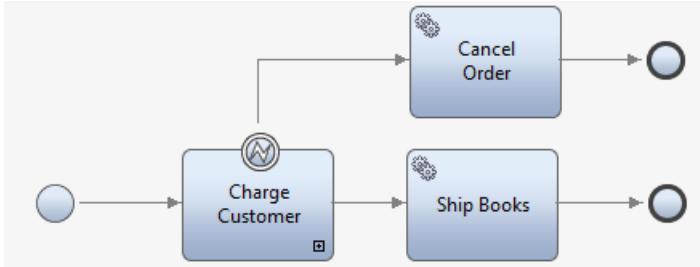
Our first pass at a design for this looks as follows:



We add an exception handler to our Charge Customer activity. This now implies that if there is a problem with billing, an exception will be thrown indicating this. A new activity called *Billing Problem* is defined which will handle the billing issue. This will have an outcome indicating whether or not the billing issue was resolved. If it was resolved, then we progress to the shipping of the books, if it wasn't resolved, we perform some action to cancel the order.

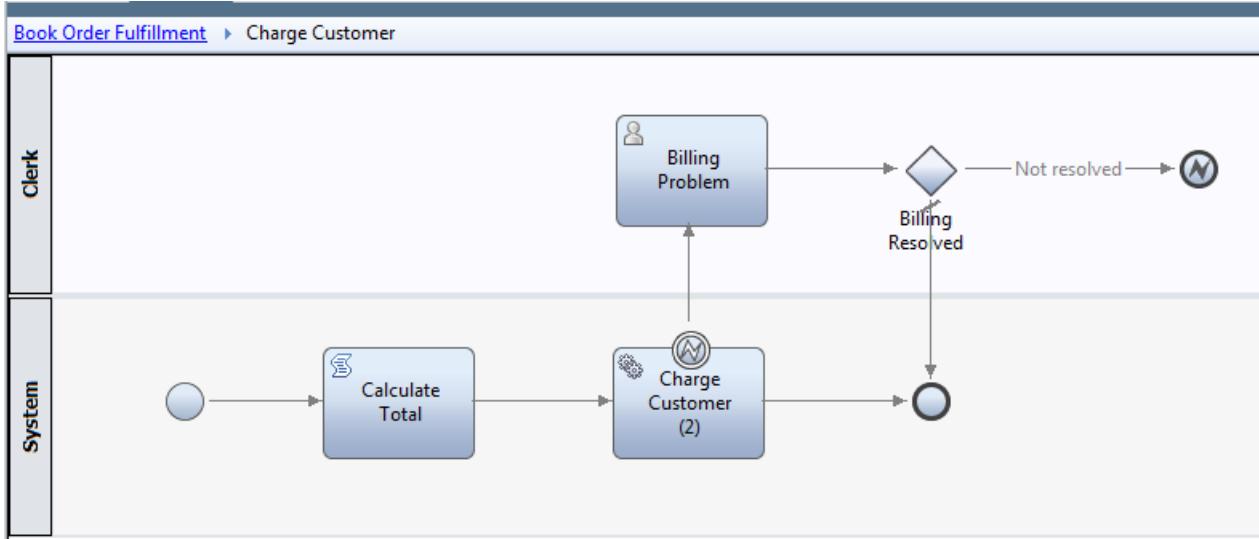
With this in mind, let us now turn our attention to what is needed to implement these steps. The first activity we examine is the *Billing Problem* step. We can see that it will be a human task as a clerk must work upon it. The input to the clerk must be the complete order which contains all the details needed to contact the customer and talk with them about their order. The output will be an indication on whether or not the billing issue was resolved. If we spend time looking further at this design, a thought should strike us ... a side effect of resolving the billing issue must include actually charging the customer's credit card but that capability is owned by the *Charge Customer* activity. In addition, our nice clean top level business process is starting to look a little complex. Maybe there is an alternate design.

Here is a second pass at our design.



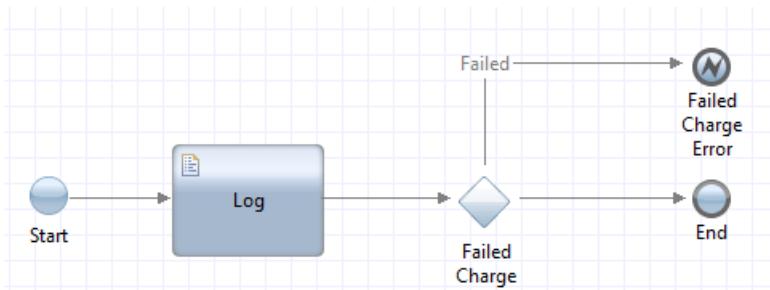
In this story, the *Charge Customer* will either succeed or fail and, if it fails, the order will be canceled. The handling of an invalid card is now *owned* by the *Charge Customer* sub-process.

The following shows a possible implementation within the *Charge Customer* sub-process.

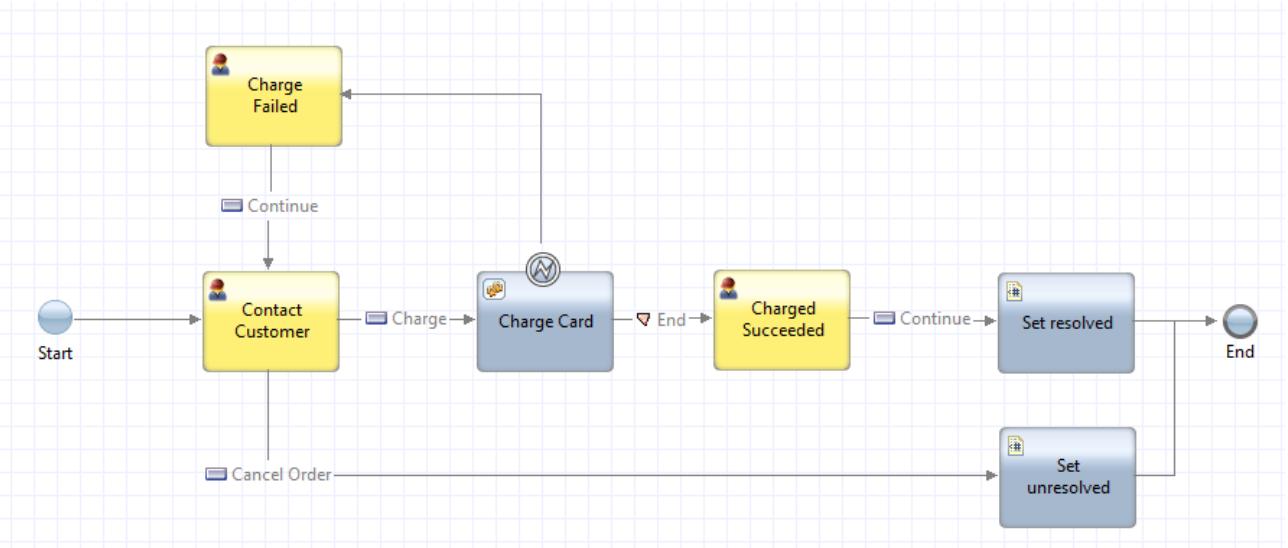


Again we assume that the *Billing Problem* activity will also charge the customer's card if all is well. There is rarely an absolute correct design for a business process and in many cases, the quality of the design will be subjective. The thought behind this, our final design, is that billing is now contained solely within the billing sub-process. Let us now turn our attention to the implementation of some of these steps.

First we notice that *Charge Customer (2)* throws an exception. Currently, our implementation does not do that. For testing purposes, we will say that if the account number on the card is 9999 then that will be our indication that the card is invalid. The new *Charge Customer (2)* implementation now looks like:



Now we get to look at the *Billing Problem* human task activity. Its input will be the Order and the Total billing amount. Its output will be an indication of whether or not the charge was resolved. An initial pass at the implementation of *Billing Problem* looks as follows:



There appears to be a lot to this so let us take some time to walk through it. First we reach a coach called *Contact Customer*. When displayed, it looks as follows:

### Billing Problem

Charge Amount

12.34

Card Holder Name

John Q Sample

Card Number

9999

Card Expiry

2099-09-09

**Cancel Order**

**Charge**

Here we will contact the customer and discuss new charging information. We can either cancel the order or charge the card. If we select Charge, we attempt to charge the card with the newly entered information. This can again fail and, if it does, we tell the clerk that it failed again. The clerk can again talk with the customer. If the charge succeeds, we tell the clerk that the charge has been made. A variable is set to indicate whether or not the charge has been processed before the end of this human service. The *Charge Card* step is an instance of a nested service that invokes the *Charge Card* service we built earlier.

Back in our process diagram, we can now associate the *Billing Problem* activity with its implementation of *Billing Problem* human service.



Having completed these steps, go back and re-run the solution and validate that it now does what you expect.

Do you see a problem with the solution as currently described? It appears that we missed gathering the customer's contact information such as email and phone number. We can go back to the Order data structure and add these missing fields.

- 
- email (String)
  - phone (String)
  - books (Book) (List)
  - billingDetails (BillingDetails)
  - shippingDetails (ShippingDetails)

The Billing Problem Human Service was also updated to reflect the new fields.

**Billing Problem**

**Card Holder Name**

**eMail address**  
noone@nowhere.com

**Telephone**  
(555) 123-4567

**Charge Amount**

**Card Number**

**Card Expiry**

**Cancel Order** **Charge**

### **Part III – Starting process instances**

So far we have been testing our processes by running them from within Process Designer and using the default values of the variables to be the input to the process. In reality, we will be starting these instances as a result of orders submitted through the Internet. In our story, we also are able to receive orders that are called in to us via the phone. This means that we need to create a mechanism where clerks can enter the orders manually. We will start by building a human service that allows us to enter the details of an order. We will call this human service *Order Entry*.

The *Order Entry* human service must do three things for us. It must allow us to select the books the customer wants, it must allow us to enter the shipping information and it must allow us to enter the billing information. Once this information has been entered, it must be able to start an instance of the process.

A first pass at the screens in the solution looks as follows:

## Enter Order

	Title	Author	Price
<input type="radio"/>	About Love	Anton Chekhov	12.34

## Book Search

Search Title

	Title	Author	Price
<input type="radio"/>	About Love	Anton Chekhov	12.34
<input checked="" type="radio"/>	Around the world in 80 days	Jules Verne	12.34

In this screen, the clerk is talking to the customer who is providing titles of books to order. The clerk searches for these by entering the title in the search box and clicking Search. A list of titles that match the search is shown. These can be added to the order. When the desired books have been entered, the clerk clicks Next to move to the next page which is the *Shipping* coach.

## Enter Order

### Shipping Details

Name

Phone

eMail address

Street

City

State

Zip

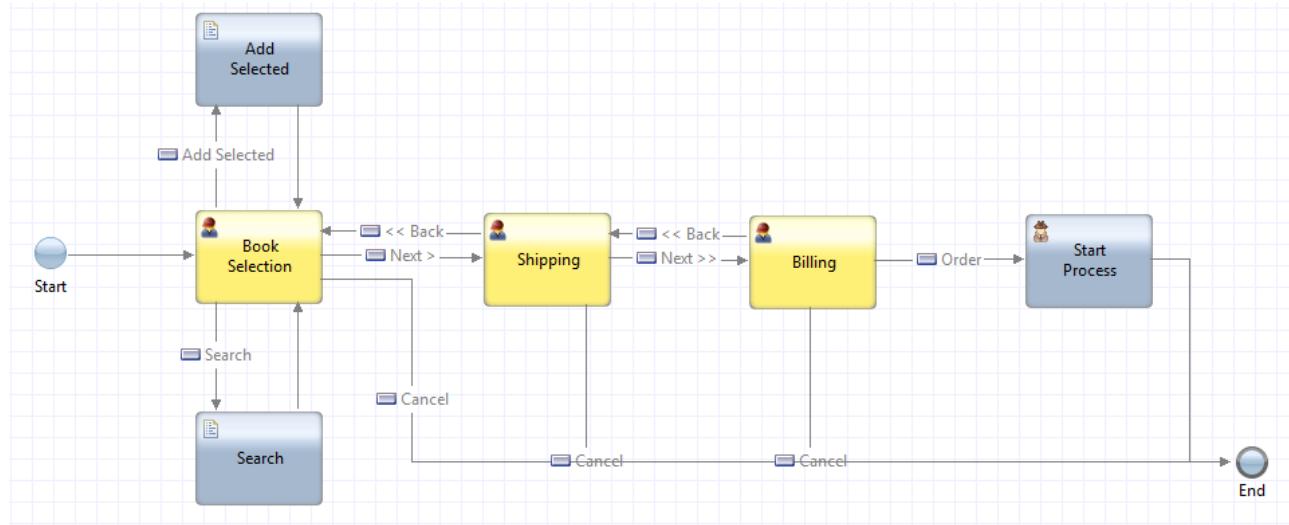
Here the clerk captures the shipping details of the customer. When done, Next is selected which takes us to the Billing coach.

## Enter Order

Billing Details

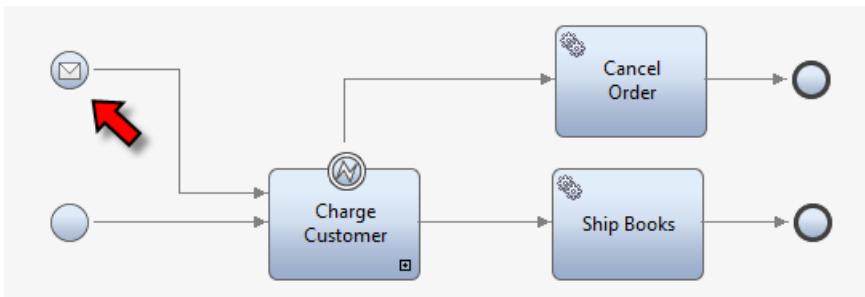
<b>Card Holder Name</b>	John Doe
<b>Card Number</b>	9876
<b>Expiry Date</b>	2013-08
<b>Street</b>	123 Elm Street
<b>City</b>	Fort Worth
<b>State</b>	TX
<b>Zip</b>	76182

Finally the billing information is requested. This is the last step before the order is ready. Once done, the clerk clicks Order and an instance of the process is started. The Human Service called *Order Entry* that accomplishes these steps is shown next:



Let us spend some time on this. Notice the three coaches colored in yellow. These correspond to the three previous screens. The primary component of interest to us now is the one labeled *Start Process*. This is an `Invoke UCA` step. By the time it is reached, an *Order* business object has been constructed and we have all we need to start an instance of the process.

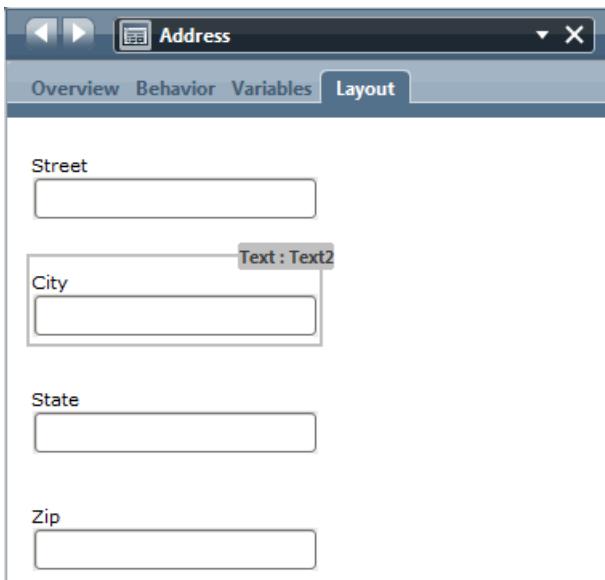
The process itself has to be augmented to allow it to be started by the corresponding UCA. The top level *Book Order Fulfillment* process now looks like:



By flagging the *Order Entry* as Exposed to be a startable service, we can now launch the *Order Entry* from the Process Portal.

The screenshot shows the IBM Process Portal interface. At the top, there's a navigation bar with "IBM", "WORK", "DASHBOARDS", and "Administrator". Below the bar, a dropdown menu is open under "Exposed As:" with "Startable Service" selected. The main area is titled "My Tasks" and shows a search bar and links for "Open Tasks" and "Completed Tasks". A message says "No tasks were found." On the right side, there are buttons for "Launch", "Following", and "@Mentions". A red arrow points to the "Order Entry" link under the "@Mentions" button.

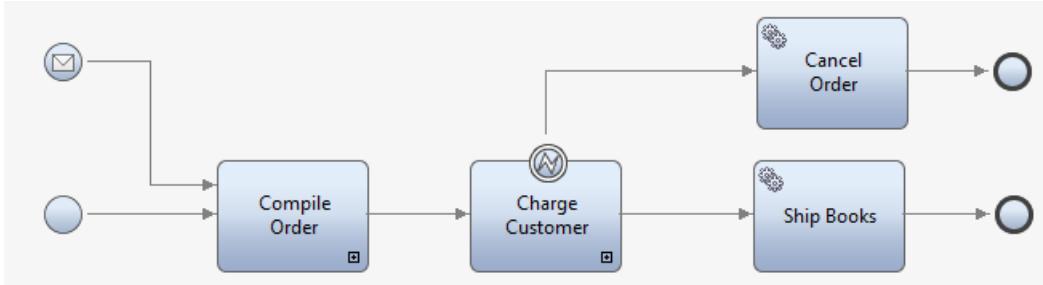
In the shipping and billing steps, we had the opportunity to enter an address in both screens. Rather than re-code the interface twice, this became a good opportunity to create a reusable visual called a Coach View. We created a Coach View that was called *Address* that looked as follows:



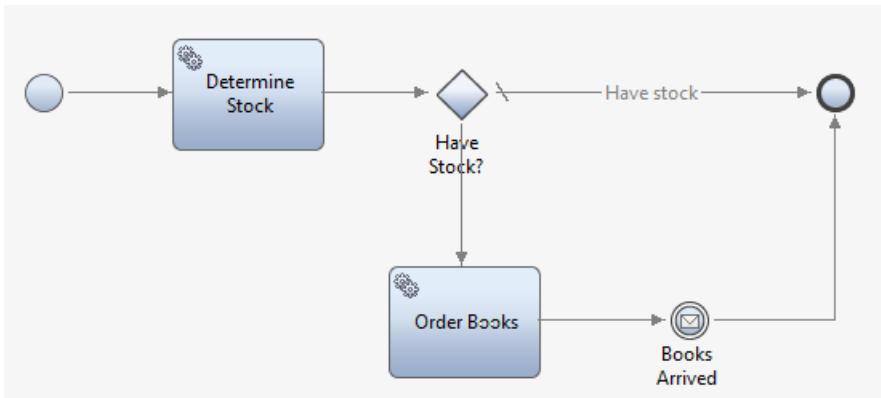
## **Part IV – Check Stock**

As our business carries on, we find that we are not managing the stock of books on hand very well. Specifically, when a customer orders a book and we don't have it in stock, we don't know quickly enough that we have to order more from our supplier. In addition, when the new supplies arrive, we are slow in fulfilling the order. To this end, we now introduce a new component in our solution that we call *Compile Order*. This step will determine that we have the books on hand to fulfill the order.

and, if not, order more stock from the supplier. We will only exit the *Compile Order* step when we have all the books necessary to ship. Looking at our new BPD below, we see the location of the *Compile Order* sub-process.



The refinement of the *Compile Order* sub-process looks as follows:



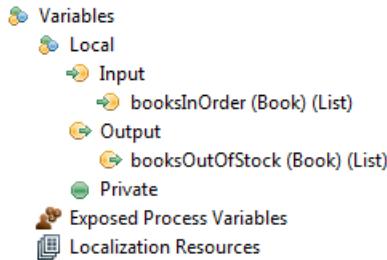
Here we take the list of books that the customer desires and compare those to the inventory of books on hand. If we have stock of all of our books, we have completed our step. In the event that we are missing some books from the order, then we execute the *Order Books* step which will contact our supplier requesting the books be sent to us. This is an asynchronous step. It could take some time for the supplier to provide us our books. Because of this, we must wait for the books to be delivered. Here we use a Message Event to achieve that task. A Message Event is an event indication sent to us by some other activity in the BPM system. That message will be generated by an un-specified separate process that is handling the inventory of books received from suppliers.

Let us not forget that there could be many parallel instances of these processes in existence. There could be multiple separate orders for different customers all waiting on different books from different suppliers. Because of this we need to ensure that the correct instance of the process is informed when books associated with it become available and not when books for a different order become known. To achieve this, we introduce the concept of an *OrderID*. An *OrderID* is a unique identifier for each order that can be used to distinguish one order from another. We will assume that the *OrderID* has been generated when the initial order was provided by the customer. We will add this as a new property of the Order data structure which now looks as follows:

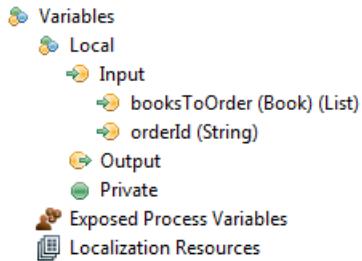
- orderId (String)
- email (String)
- phone (String)
- books (Book) (List)
- billingDetails (BillingDetails)
- shippingDetails (ShippingDetails)

The *Determine Stock* step will again be a technical step. This will compare the current order with available in-house stock and build a new list of out-of-stock books. For our purposes, we will again stub this out. We will simply say that any book that has a title beginning with "Z" is considered out of stock. We will implement *Determine Stock* as a general service. The interface to the *Determine*

*Stock* service will look as follows:



The *Order Books* step will execute a request to order books from the supplier. The inputs to this step look as follows:



Again we will implement this service as a stub merely logging the fact that we are requesting books.

The *Books Arrived* step is interesting. It blocks the process until a corresponding message has been delivered. The message will come from a UCA. We will match the incoming message content's *orderId* value to the one contained in the process. If the matching message arrives, we now have the books in stock and can proceed.

A UCA called *Books Arrived* was built to be associated with this message event.

## ***Part V – Advanced UI***

**<TBD>**

## ***Part VI – Business Monitoring***

**<TBD>**

# POC Considerations

## *Choice of server hosting environment*

IBM BPM is typically installed and executed on a Windows environment although Linux can be used as an alternative. The amount of RAM should be 4GBytes or better. 8GBytes is ideal. During solution development, the logs of the WAS server will need to be consulted frequently. This can cause problems if the developer's workstations hosting PD are remote from the Process Center/Process Server. The ideal solution is to export the directory hosting the logs on a network share so that they can be mounted/accessed remotely. This is easier if Process Center is installed on Windows but much more difficult if the Process Center is installed on Linux. Technology options exist to share files on Linux such as Samba but they need to be explicitly set up and we need to ensure that the evaluator has the skills to do this.

## *Choice of developer environment*

The developer's environment should be a Windows machine for installation of Process Designer and/or Integration Designer. 4GBytes of RAM is sufficient for this task. Screen resolution should be as large as possible. Process Designer assumes high screen resolutions and the larger the better. The developer's environment should *mount* the directory from the server to show the logs from the server. This is also affected by the choice of hosting environment.

## Software choices

Even if IBM BPM Standard is the edition of the product being evaluated, install IBM BPM Advanced. The reason for this is to recognize that even though IBM BPM Advanced is being installed, we can trivially create profiles for IBM BPM Standard. Although having Advanced installed will consume additional disk space that is a small price to pay. Experience has shown that as a PoC progresses, the evaluator may wish to see Advanced or, alternatively, a decision that Standard would suffice turns out to be incorrect. The penalty of installing Standard and then wishing to move up to Advanced is huge if Standard was what was installed in the first place.

When it is time to install the software, ensure that all possible packages are in your possession before starting out. Never assumes that additional packages or parts can be download while at the evaluators location. This includes the latest maintenance. Don't assume that Installation Manager can "reach out" to the Internet to download additional parts.

If all things are equal, use IBM DB2 as the database. Although the product is supported on a number of databases, it is a good bet that the developers who build the product are using DB2 on a constant basis. It is extremely likely that DB2 has been more thoroughly tested as a database than the other database types.

Bring any additional software packages that *may* be used with you. This includes any "tail tools" such as BareTail or LogExpert. If Web Services or REST APIs are to be used, bringing soapUI is also strongly recommended.

## Environment Checklist

### DB

Item	Example	Reality
DB Type	DB2, Oracle ...	
DB Hostname	myhost.ibm.com	
DB Port	50000	

DB Admin Userid	db2admin	
DB Admin Password	db2admin	
PDW DB Name	PDWDB	

## Process Center

Item	Example	Reality
Hostname	myhost.ibm.com	
Admin userid	admin	
Admin password	admin	

# Kolban's Projects

## *Apache Velocity Integration*



Apache Velocity is an Open Source template engine that takes as input a text string and a set of *variables* and returns the original text string with variable references replaced with their values. In addition to simple name/value mapping, Velocity has many other features and options which makes it an extremely versatile template language including conditionals and looping.

Seeing the capabilities of velocity, the goal of this project was to see how it could be integrated into a IBPM solution. A Java Connector was written which *wraps* the Velocity template library and exposes it as a trivially invocable service from a BPD activity or as a nested call within the another service component.

The input to the Velocity Service are two parameters.

One parameter is a text string that is an instance of a velocity template. This can be hard coded, retrieved from a managed file or retrieved from a URL. The second parameter is a IBPM Map variable. The keys in the map are the velocity variable names and the values are the IBPM variables.

This component is *an alternative* to the JavaScript functions supplied as core as part of the product. It has advantages in a number of circumstances such as repeating iterations, conditions as well as having the definition of macros and other powerful capabilities. The details and value proposition of the Apache Velocity project can be read about here:

- [Apache Velocity Project](#)
- [Apache Velocity User's Guide](#)

Making use of this wrapper removes the need for anyone to have to know Velocity coding APIs and makes its functions immediately available for use.

## **The Velocity Template Language**

The Velocity Template Language (VTL) is a simple yet powerful language that is used to process text. Here is a quick summary

Variable references

A variable name prefixed with a \$ symbol will be expanded during rendering.

Here is some text \$name will replace the name variable

An alternative style is to place the variable name in curly braces

\$name

and

`${name}`

are identical

Function invocation

```
$classVariable.methodName()
```

Directives

```
#if (expression)
...
#elseif (expression)
...
#else
...
#end
#set($variable = "value")
```

Comments

```
This is a comment
/*
This is a multi line comment
*/
#foreach ($variableName in $listVariable)
...
#end
```

These are only some of the more commonly used capabilities. The VTL is much richer and one should examine the VTL user guide and reference for more details. The reference for VTL can be found at the "[VTL Reference](#)".

In order to run, velocity needs the following mandatory pre-reqs

- [Commons collections](#) – Supplied with WLE (eg. <WLE>/twinit/lib/commons-collections.jar)
- [Commons Lang](#) – Supplied with WLE (eg. <WLE>/twinit/lib/commons-lang.jar)

Unfortunately, with the latest levels of Velocity, these won't work. The latest versions of these classes need to be added to the packaging.

Links

[The Apache Velocity Project](#)

## **XSLT DataHandler**

Although not supplied with WPS by IBM, this section describes a project that constructed an XSLT DataHandler.

XSLT can be used to construct output from an input XML document and an XSL template that describes how the transformation should work. This can be used to convert one XML document format to another. If we had an XSLT DataHandler, then the output data generated by the DataHandler could be generated by the XSLT processing.

If the incoming data were an XML document but **not** of a format suitable for converting to a Business Object, the XSLT DataHandler could be used to transform the incoming XML to BO format XML before building the BO.

The XSLT DataHandler has one configuration parameter which is the name of an XSL Template file. This file should be packaged as part of a deployed project.

At a technical level, and usually not seen by consumers of the datahandler, the input formats supported by the DataHandler can be:

- InputStream
- String
- StringReader
- DataObject

To use the DataHandler, obtain the XSLTDataHandler.jar file. This contains the implementation of the solution.

Add the JAR into the WID project or library that is to use it.

<Info Needed>

In WID, add the new DataHandler in the BindingRegistry.

## Converting a BO to a plain XML

The following stylesheet can be used to transform a BO to a plain XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
 <xsl:output standalone="omit" encoding="UTF-8"
 omit-xml-declaration="yes" media-type="text/xml" method="xml" indent="yes"
 version="1.0" />
 <xsl:template match="/child::element()">
 <xsl:element name="{local-name(.)}">
 <xsl:copy-of copy-namespaces="no" select="child::*" />
 </xsl:element>
 </xsl:template>
</xsl:transform>
```

Author: Neil Kolban

Date: 2011-01-02

EMail: kolban@us.ibm.com

## ***IBM BPM 7.5 Coach Auto-save***

Consider a coach screen being shown to a user. The user enters a bunch of data and then goes for lunch. The system then crashes or his browser crashes ... and he has lost his work. The question becomes ... how could we have saved his work?

At a high level, let us imagine that periodically we wish to save the current state of data entered into fields. We will assume that we have some external mechanism to do just that. Next we should assume that if a page is re-loaded, we have a way to determine the uniqueness of the page and re-load any saved data that was auto-saved.

The saving of data will be accomplished by an Ajax call passing in the name/value pairs of the fields we wish to save.

## ***Process and Task Data Explorer***

This is a project that I flag as "experimental". I have chosen to make it available now just in case anyone may have some use of it. The project is a BPM v8 Process App that, when installed, provides the user with a table of all the process and tasks found in the system. Clicking on a row in the table shows the details of that particular task.

IBM Process Portal

https://localhost:9443/mum/enabler#list%20%20

IBM WORK DASHBOARDS Administrator ?

### Process and Task Data ...

Refresh Change Columns Search Type: Instances

Instance Name	Task Subject	Task Activity Name	Instance ID	Task ID	BPD Name	Instance Status	Task Status	Instance Due Date
Book Order Fulfillment:356	Step: Billing Problem	Billing Problem	356	875	Book Order Fulfillment	Terminated	Closed	2012-07-17T17:55:
Book Order Fulfillment:357	Step: Billing Problem	Billing Problem	357	878	Book Order Fulfillment	Terminated	Closed	2012-07-17T17:56:
Book Order Fulfillment:358	Step: Billing Problem	Billing Problem	358	881	Book Order Fulfillment	Terminated	Closed	2012-07-17T17:56:
Book Order Fulfillment:359	Step: Billing Problem	Billing Problem	359	884	Book Order Fulfillment	Terminated	Closed	2012-07-17T17:56:
Book Order Fulfillment:360	Step: Billing Problem	Billing Problem	360	888	Book Order Fulfillment	Terminated	Closed	2012-07-17T17:57:
BPDVT1:439	Step: ABC	ABC	439	944	BPDVT1	Terminated	Closed	2012-07-25T21:36:
BPD1:441	Step: HS	HS	441	945	BPD1	Completed	Closed	2012-07-27T10:10:

**Process Details**

Process Instance:	360	Task Instance:	888	resolved - null
Name:	Book Order Fulfillment:360	Name:	Billing Problem	chargeAmount - 24.68
Description:		Description:		order
Process App Name:	Book Orders	Display Name:	Step: Billing Problem	books
Process App Acronym:	BK2	Kind:	KIND_PARTICIPATING	billingDetails
Process Template Name:	Book Order Fulfillment	Originator:	tw_admin	shippingDetails
State:	STATE_TERMINATED	Owner:		
Execution State:	Terminated		All Users_T_da7e4d23-78cb-4483-98ed-b9c238308a03.86104947-982f-4aae-a9c6-56aa773f1b08	
Due Date:	2012-07-17T17:57:58Z	Assigned To:		
Last Modification:	2012-07-24T16:28:53Z	Process Identifier:	360	
		Service ID:	1.81faf03a-bc0a-4044-91ea-0966115e20bb	
		Priority:	30	
		Process Instance Name:	Book Order Fulfillment:360	
		State:	STATE_FINISHED	
		Status:	Closed	

**Task Details**

- resolved - null
- chargeAmount - 24.68
- order
  - orderid - 0:123
  - books
  - billingDetails
  - shippingDetails

The application is launched from Process Portal within the Dashboards section:

The screenshot shows the IBM Process Portal interface. At the top, there's a navigation bar with links like 3D, Games, IBM, News, Weather, Wikis, Search, Projects, and others. Below that is a main header with 'IBM' and 'WORK'. A 'DASHBOARDS' dropdown menu is open, showing options: My Performance, My Team Performance, Process Performance, My SLA Overview, Process and Task Data Explorer (which has a red arrow pointing to it), and Ad Hoc Reports. The main content area displays a table titled 'Process and Task Data' with columns: Instance Name, Task Subject, Step, ISK ID, and BF. There are three rows of data, each representing a 'Book Order Fulfillment' task with a 'Billing Problem' step and an ISK ID of 875, 878, or 881.

Instance Name	Task Subject	Step	ISK ID	BF	
Book Order Fulfillment:356	Step: Billing Problem	Billing Problem	356	875	Bc Fu
Book Order Fulfillment:357	Step: Billing Problem	Billing Problem	357	878	Bc Fu
Book Order Fulfillment:358	Step: Billing Problem	Billing Problem	358	881	Bc Fu

There are some context menu options available by right clicking on a row but these haven't been fully polished yet.

The solution is implemented in a combination of Coach Views and Dojo.

It is available for download from here:

[http://www.neilkolban.com/IBM/DataFiles/Process\\_and\\_Task\\_Data%20-%202012-07-24.twx](http://www.neilkolban.com/IBM/DataFiles/Process_and_Task_Data%20-%202012-07-24.twx)

## *Out of Office support*

Consider a user being out of the office or away on vacation or being out sick. He may wish any new tasks that were routed to him to be re-routed to someone else for a period.

One way of achieving this is to have a demon process that runs every period (say every 15 minutes). This process could look for new tasks that have been assigned to our user. If found, they could be re-assigned to another user.

Turning this into a generic story ... we could have an algorithm that reads as follows:

Every 15 minutes, run the following:

Find the set of tasks that have been created since last time we looked and are not yet in the Closed state;

For each unique user owner of the tasks

```
{
 Retrieve the out of office instructions for the userid (if present)
 Execute the out of office instructions for the userid for each of the tasks owned by this userid
}
```

# Tutorials

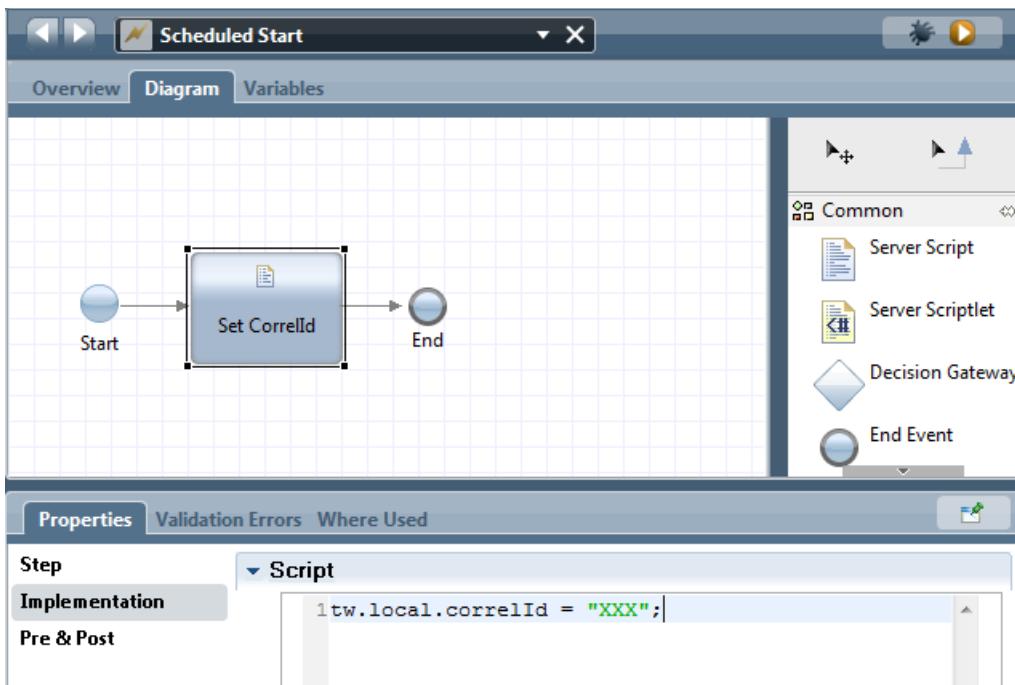
In the following sections we work through some illustrative tutorials demonstrating some functional areas of the product.

## ***Starting a business process on a schedule***

There are times when we want a Business Process to be started on a repeatable schedule. Perhaps we want to run a process to start re-stocking at the end of each day. Fortunately, IBPM provides an elegant solution for just this capability. UCAs can be triggered to start at defined intervals and using a Start Message Event activity in a BPD allows a BPD to be associated with a UCA such that when the UCA is fired, the process instance can run.

### 1. Define a General Service to be triggered by the UCA

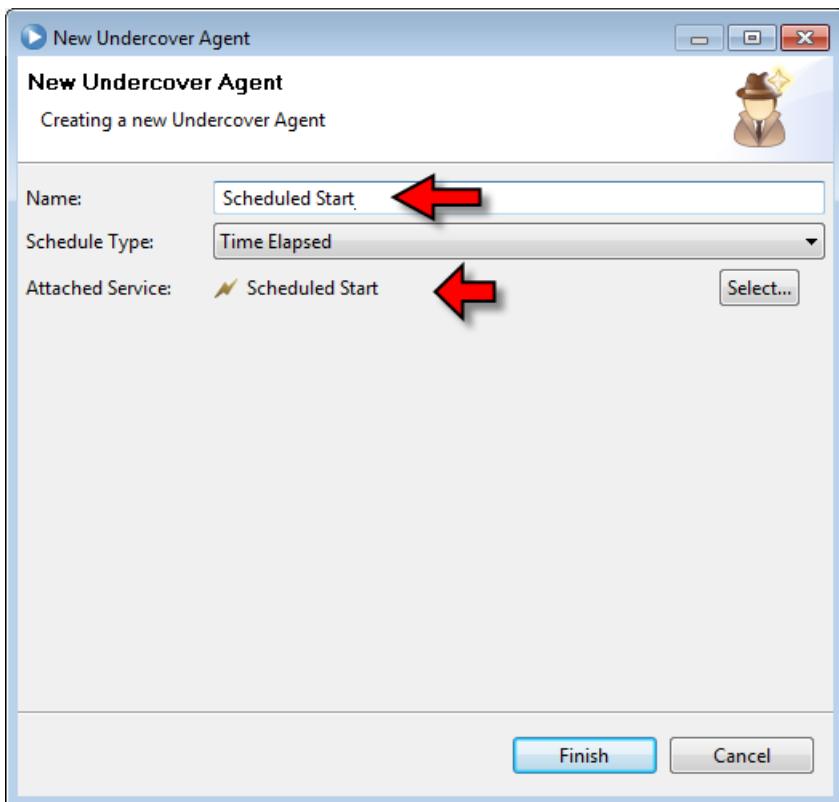
Create a new General System service called Scheduled Start. Define an output variable called "correlId" of type string. Define a Server Script node to set the value of the output variable to be "XXX".



This will be the General Service that is started by the Scheduled UCA.

### 2. Define a scheduled UCA to be fired

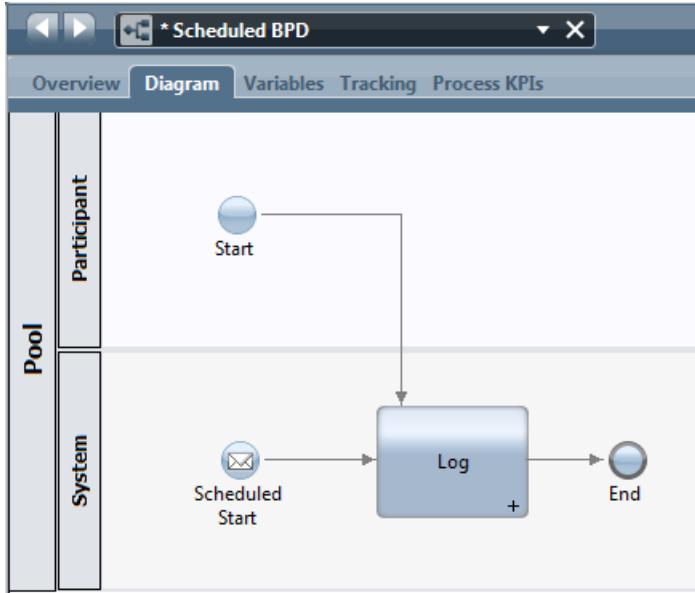
Add a new Under Cover Agent (UCA) called Scheduled Start and associate this with the General Service that we just created (also called Scheduled Start)



In the UCA implementation, selected all the entries in the right column that says that the start time for the UCA will be every 5 minutes:

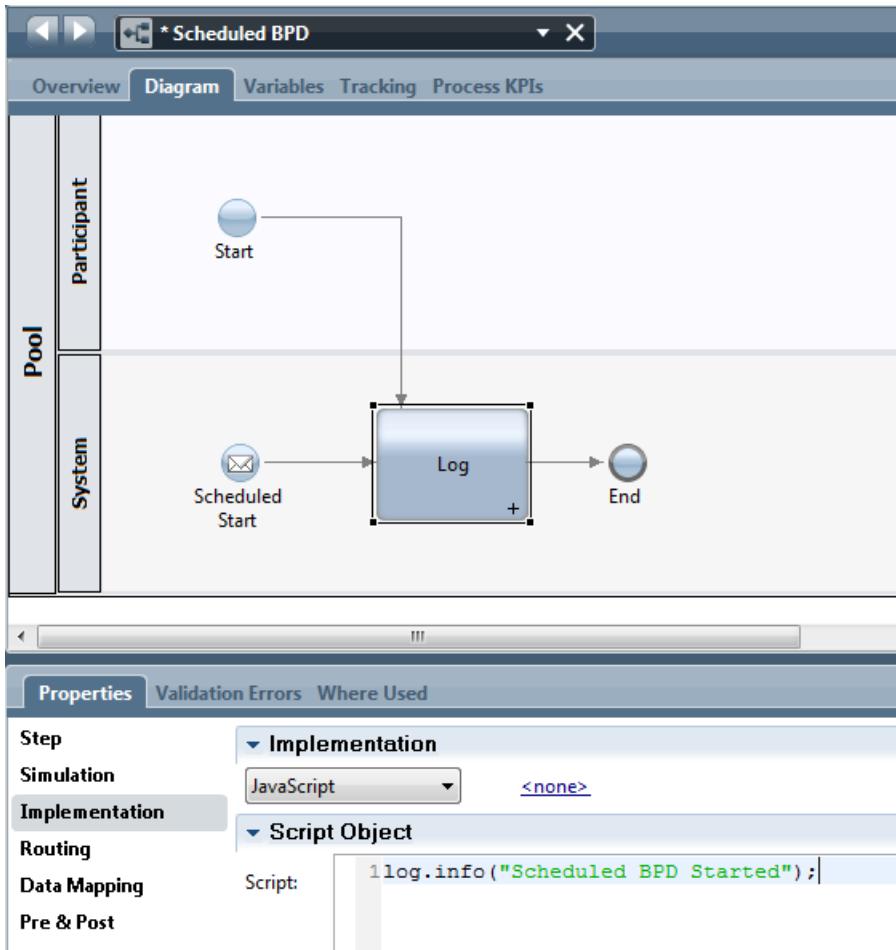
3. Define a BPD to be triggered by the UCA

Create a new BPD called Scheduled BPD. Build and wire the BPD as follows:



Add a private variable called "correlId" of type String.

For the implementation of the Log activity, define it to be a piece of in-line JavaScript that logs that the BPD was started:



For the Start Message Event, associate it with the Scheduled Start UCA. For the UCA output mapping, map to the "correlId" variable.

**Properties** Validation Errors Where Used

**Step**

**Simulation**

**Implementation**

**Pre & Post**

**Message Trigger**

Attached UCA: [Scheduled Start](#) [Select...](#) [New](#)

Condition:

Consume Message:

Durable Subscription:

**UCA Output Mapping**

[correlId \(String\)](#) [tw.local.correlId](#)

In the Pre & Post settings of the Start Message Event, give the correlId variable an initial value of "XXX":

**Properties** Validation Errors Where Used

**Step**

**Simulation**

**Implementation**

**Pre & Post**

**Pre Assignments**

tw.local.correlId ["XXX"](#)

#### 4. Deploy the application to a Process Server

Create a snap shot of the Process App and deploy to a Process Server (not the Process Center).

#### 5. Watch/test the solution

If we now start a Process Admin session and visit Event Manager → Monitor, we will see a scheduled entry for our new Process App:

Event Manager > Monitor

Scheduler ID	Status	Connect expiration	# Jobs Executing						
win7-x64		Jul 10, 2011 2:34:42 PM	0						
Total Jobs Executing: 0 Total Jobs: 2									
<a href="#">Refresh</a>	<a href="#">Pause</a>	<a href="#">Resume</a>	<a href="#">Pause All</a>						
<a href="#">Resume All</a>									
Scheduler	Process App / Toolkit	Snapshot	Job Name	Job Queue	Scheduled Time	Last Scheduled Time	Last Execution Time	Next Scheduled Time	Job Status
	UCATests3	SN1	Execute UCA Scheduled Start, on set schedule	Async queue	7/10/11 2:35:00 PM			7/10/11 2:40:00 PM	Scheduled
	Process Portal	7.5.0	Execute UCA Periodic SLA Update, on set schedule	SYNC_QUEUE_1	7/10/11 2:45:00 PM	7/10/11 2:30:00 PM	7/10/11 2:30:00 PM	7/10/11 3:00:00 PM	Scheduled

This shows the next time it will fire.

BUG!!! It seems that I have to create a new snapshot and deploy twice before it works.

## **Building a Monitor Model for BPMN events**

In this tutorial we are going to walk through the construction of a Monitor Model from scratch to handle the processing of incoming Tracking Group events. This techniques does not start with a generated monitor model. It is assumed that you have skills in:

- BPD modeling
- Tracking groups
- Monitor model editing
- Business Space

This is not a tutorial for newbies to either IBPM or Business Monitor.

### 1. Build the BPD

In our story, we want to generate tracking events. To start we assume a new process application. Into this we create a new Tracking Group that, in this story is called "sale":

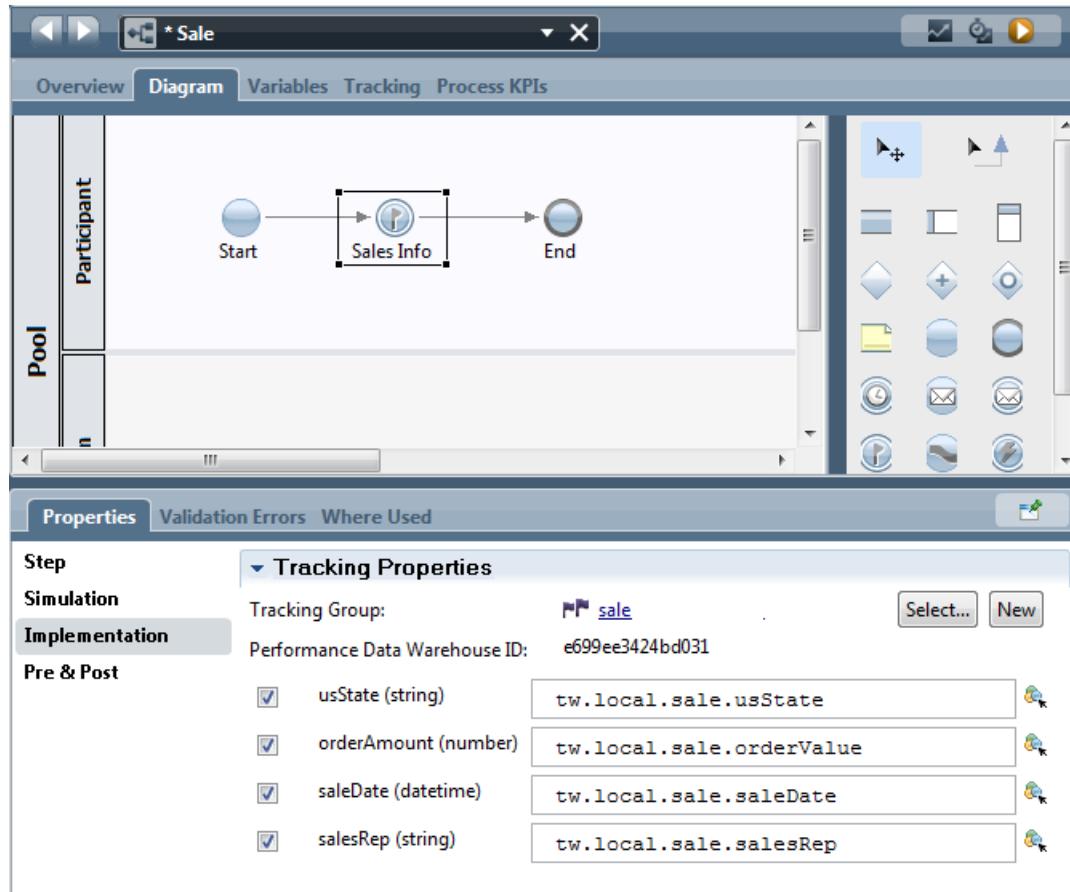
The screenshot shows the 'Tracking Group' configuration window. At the top, the title bar says 'sale'. The main area is divided into two tabs: 'Common' and 'Details'. Under 'Common', the 'Name' is set to 'sale', 'Enabled' is checked, and 'System ID' is 'guid:56d35d2f221c8d0e:79f0aded:130'. Under 'Details', there is a 'Performance Data Warehouse ID' field with value 'a6da5c4905bd031'. Below these tabs is a 'Documentation' section with a large empty text area. At the bottom left is a 'Tracked Fields' section containing four fields: 'usState (string)', 'orderAmount (number)', 'saleDate (datetime)', and 'salesRep (string)'. There are 'Add' and 'Remove' buttons next to this list. On the right side, under 'Tracked Field Details', there is a single entry for 'usState' with type 'String' and a documentation text area.

The fields that we want to monitor when the process instances run are:

- `usState` – The state to which the order is to be shipped
- `orderAmount` – The sales amount in dollars
- `saleDate` – The date of the sale

After having constructed this Tracking Group, we next build a BPD that includes an Intermediate

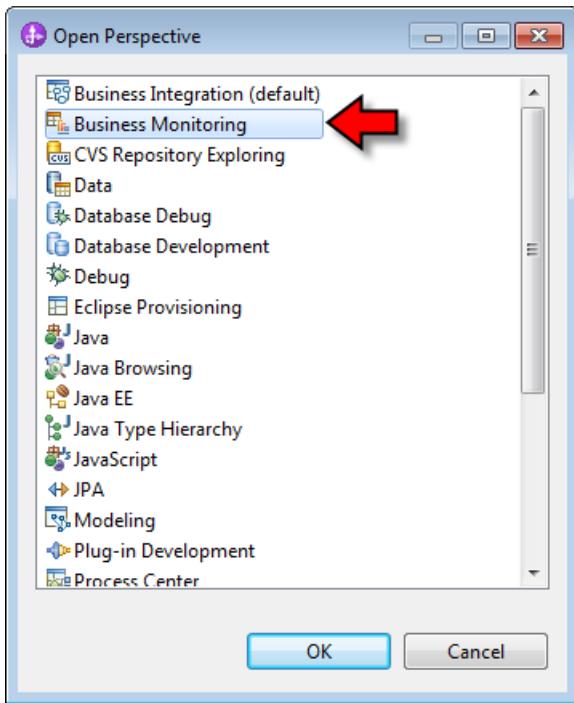
Tracking Event to externalize the event for monitor consumption:



The fields of the tracking group are populated from a variable (not shown in this diagram).

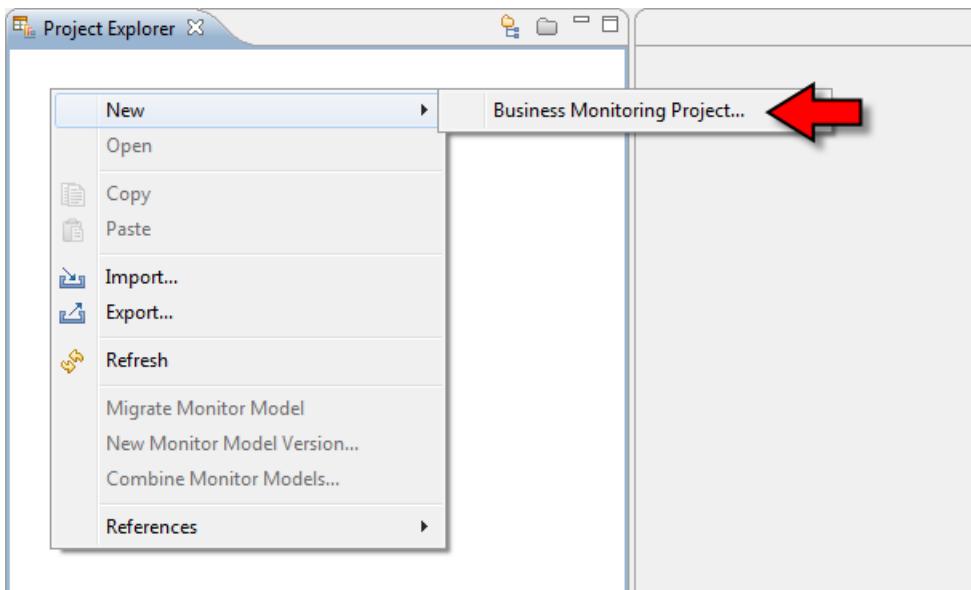
## 2. Start Integration Designer and open the Monitoring Perspective

The construction of the Monitoring Model is performed in Integration Developer. Start a copy of that and open the Business Monitoring Perspective. This perspective owns all the editors and views needed to build the model.

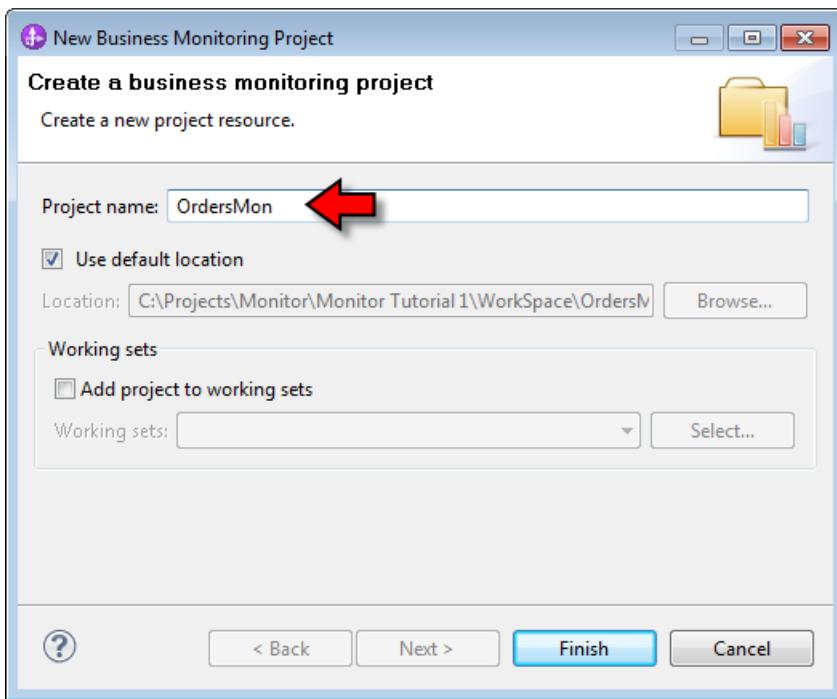


### 3. Create a new Business Monitoring Project

We start by creating a Business Monitoring Project to hold all our artifacts.



We choose to call the project "OrdersMon".

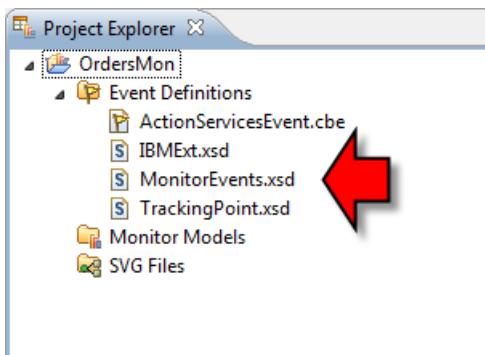


#### 4. Build and import the three XSD files

The incoming events that arrive from IBPM are in an XML data format. In order for Monitor to be able to work with these events, we need to describe the format of the events through XSD descriptions. The XSD descriptions are surprisingly not supplied with the products. Instead they are found in the InfoCenter here:

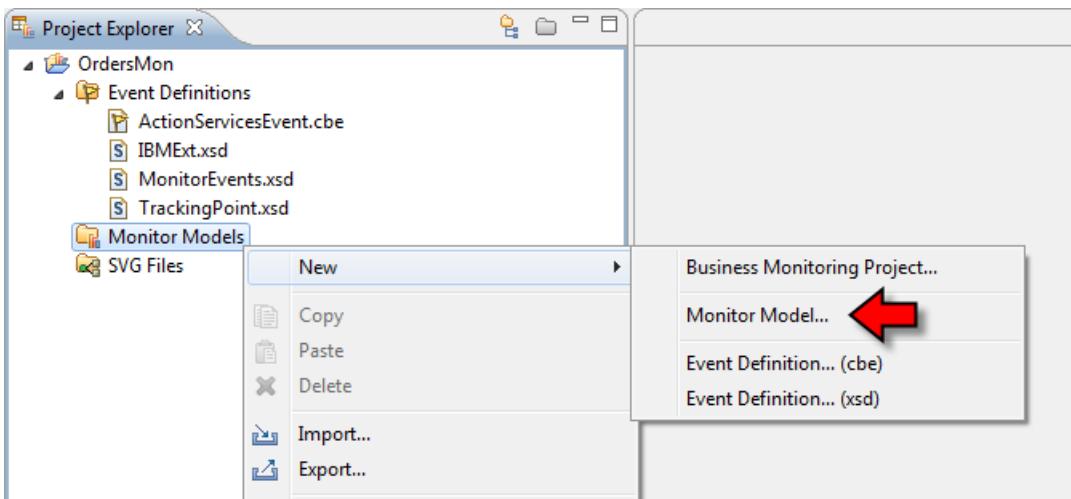
[http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.admin.doc/topics/rmon\\_eventschemaext.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wbpm.admin.doc/topics/rmon_eventschemaext.html)

These three XSD files should be added into the Event Definitions section.

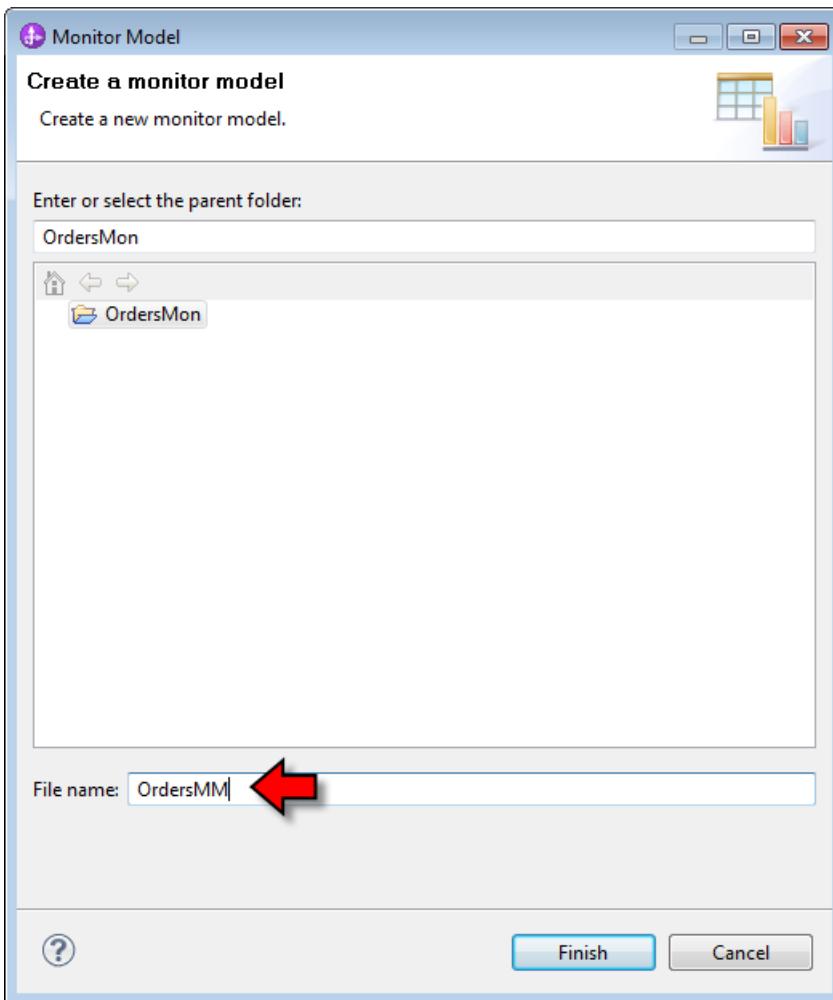


#### 5. Create a new Monitor Model

An individual Monitor Project can contain multiple Monitor Models. We have now created our project and now need to build our model. In the Monitor project, we create a new Monitor Model.

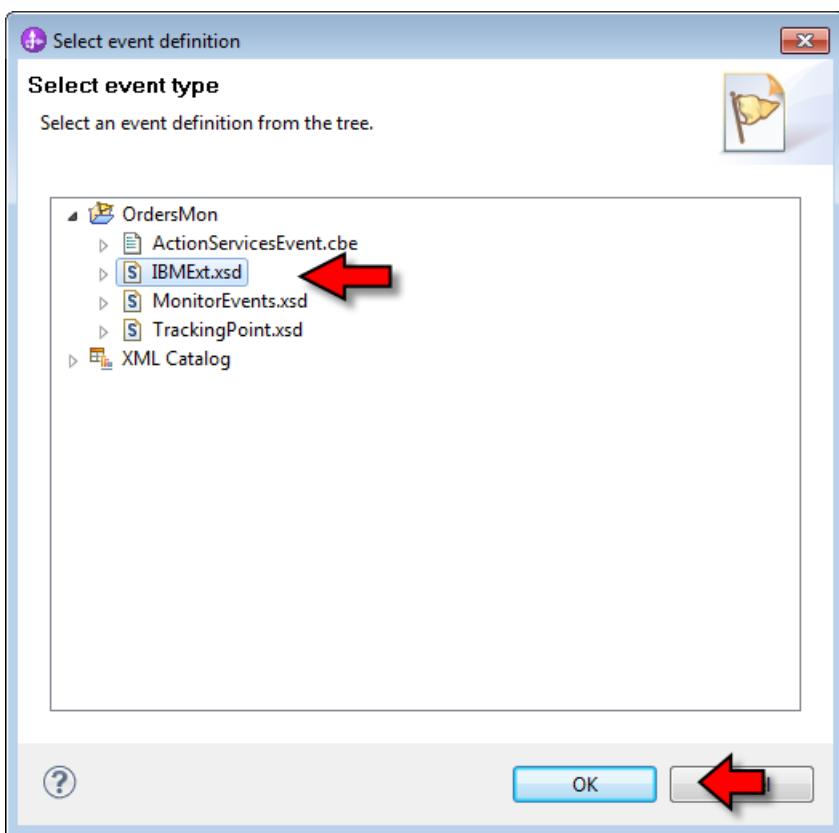
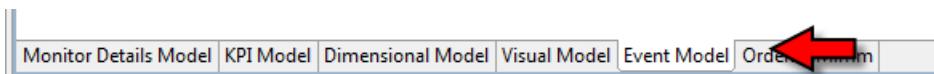


We call our new model "OrdersMM".



## 6. Add the event definition references

Although we have added our event descriptions to our project, we have not yet said that they are to be associated with the model. In the model editor, select the Event Model tab and add each of the XSDs to the model. It appears that order of addition is important so add them in the order shown:

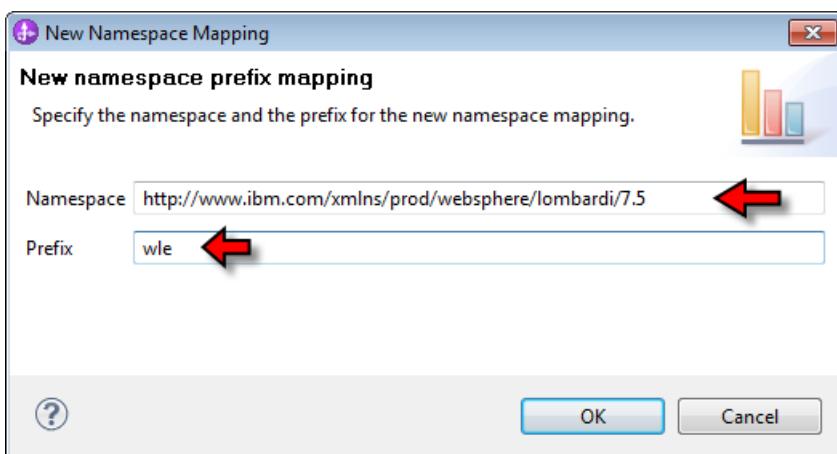


Import Location or Namespace	Event Definition Type Name
http://www.ibm.com/AC/commonbaseevent1_0_1	
IBMExt.xsd	
MonitorEvents.xsd	
TrackingPoint.xsd	

## 7. Add the namespace prefixes

As we build out the XPath entries later on in the model, we will refer to some namespaces that need to be explicitly defined with their own prefixes.

Prefix	Namespaces
--------	------------



### 8. Rename the context key

When the model was created, it was given a default key and name. We want the key of this monitoring context to be the Process ID instance. We rename the ID of the key to be "processId"

OrdersMM

Monitor Details Model

Key Details

ID: OrdersMM\_Key

Name: OrdersMM Key

Description:

Change Element ID

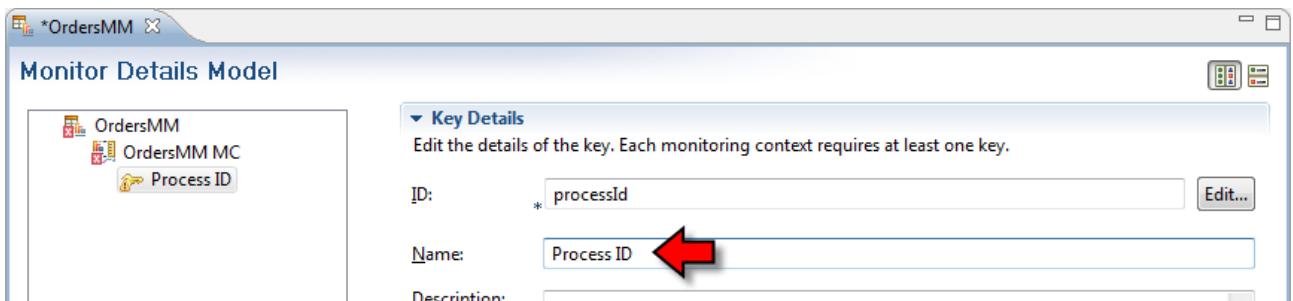
Refactor ID

To change the ID of this element, type a new ID and click Finish to proceed or click Next to preview the list of ID refactoring changes.

ID: processId

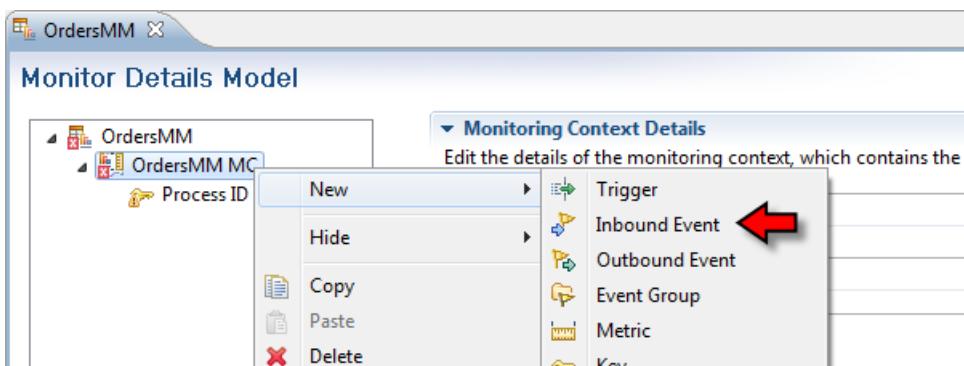
Finish Cancel

And then rename the Name field to be "Process ID".

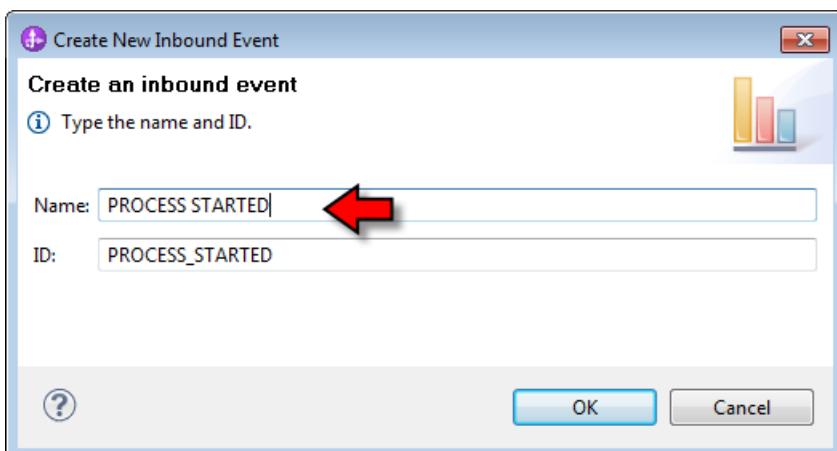


### 9. Add the Process Started inbound event

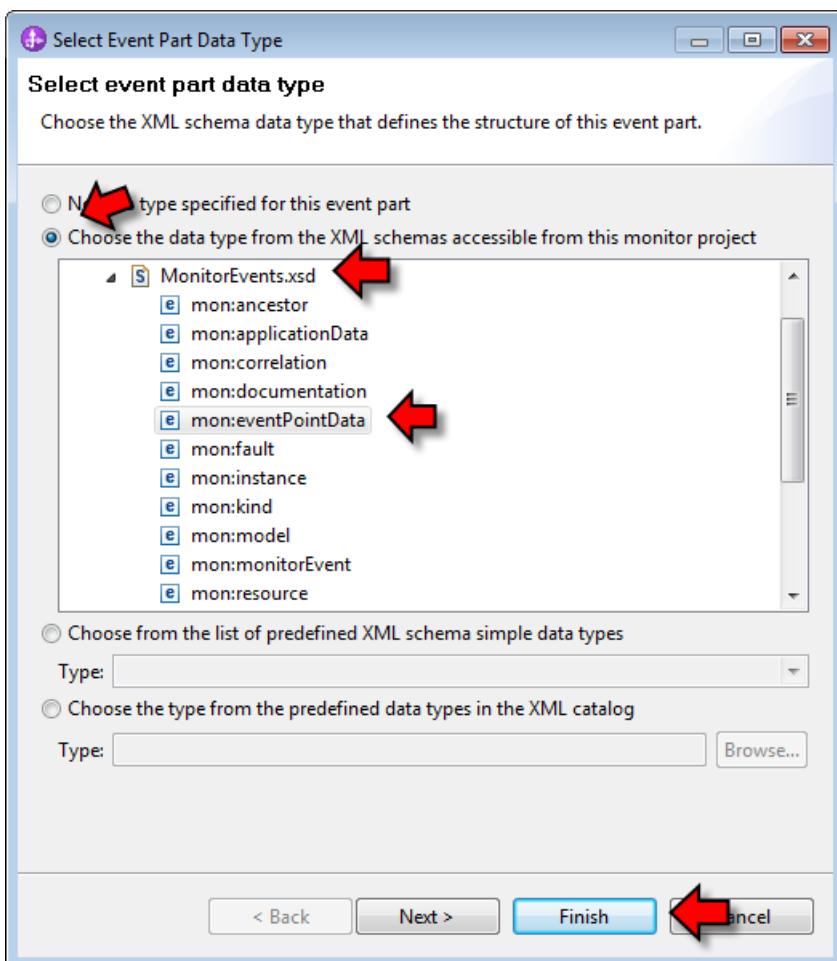
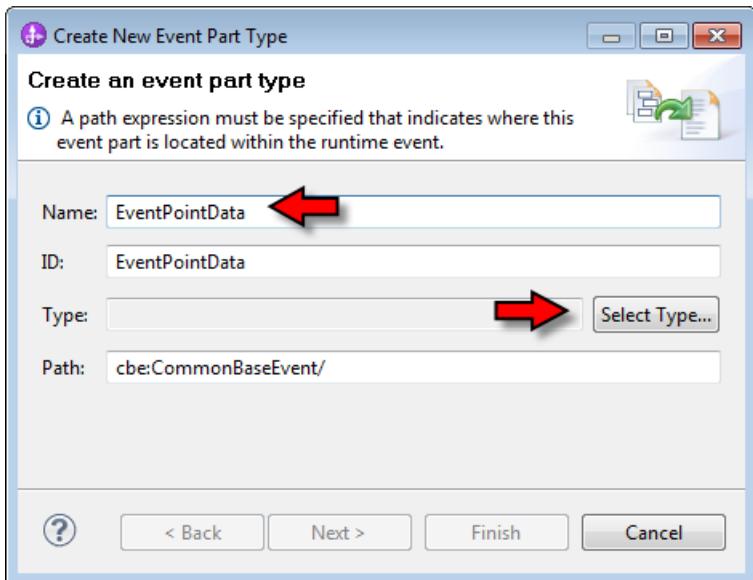
We are now at the point where we start to add the core artifacts into the model. We start with an inbound event that signifies a new instance of a BPD process has started. We add a new inbound event.



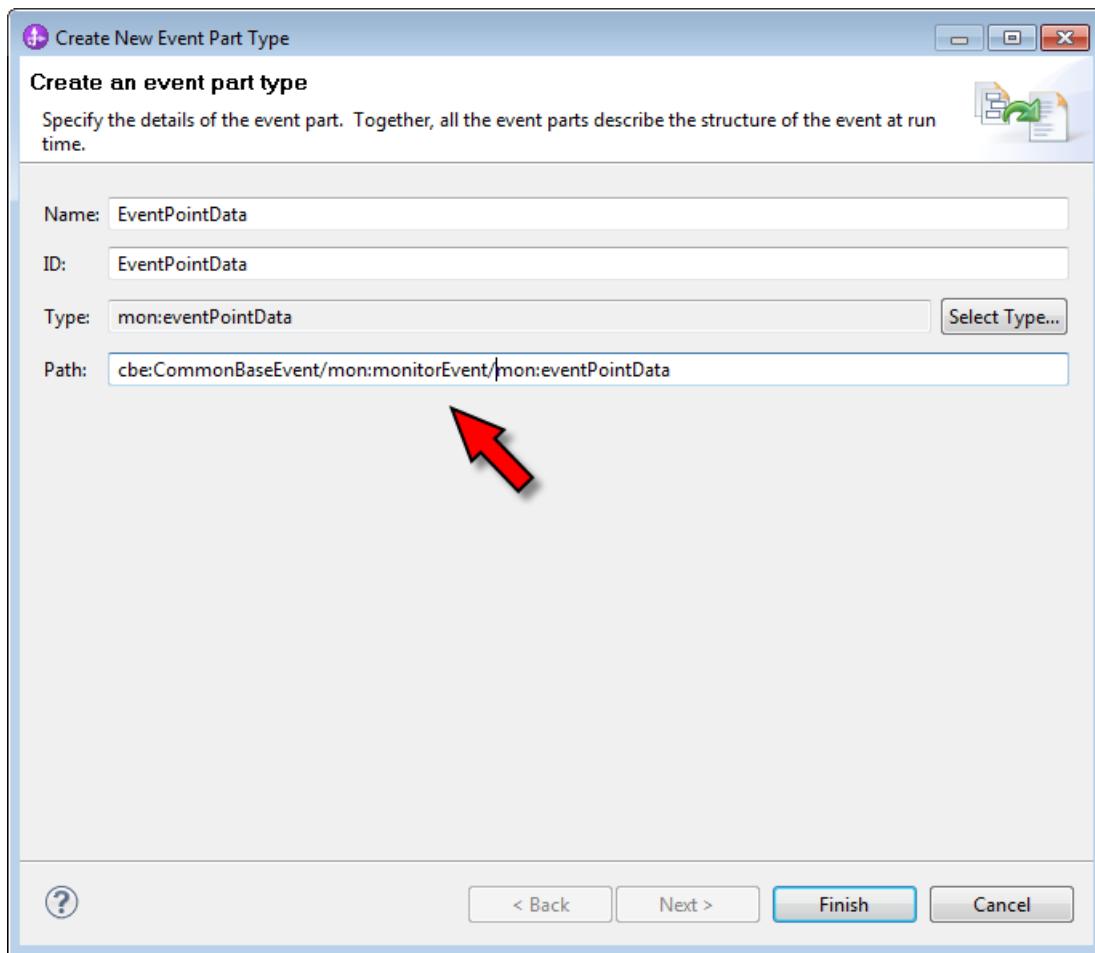
And call it "PROCESS\_STARTED".



The input data is an "eventPointData" object



and we need to take care to explicitly say where this eventPointObject can be found within the incoming event. Notice the addition of the "mon:monitorEvent" part as shown in the following figure.



The final event types definition looks as follows:

**Event Type Details**  
Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both.

Extension name:	Browse...	Clear																					
Event parts:	<table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Type</th> <th>Path</th> </tr> </thead> <tbody> <tr> <td>EventPointData</td> <td>EventPointData</td> <td>mon:eventPointData</td> <td>cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			ID	Name	Type	Path	EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData												
ID	Name	Type	Path																				
EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData																				
	Add	Remove																					

Next comes the Filter Condition. It is the Filter Condition which specifies whether or not the incoming message is an instance of this kind of event.

We look at the mon:kind field of the EventPointData and check to see if it is a PROCESS\_STARTED event. If it is, we have a match.

```
xs:string (PROCESS_STARTED/EventPointData/mon:kind) =
'{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS_STARTED'
```

▼ Filter Condition

Define a condition based on the event attributes to identify whether to accept an event of this type.

xs:string(PROCESS\_STARTED/EventPointData/mon:kind) =  
'{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS\_STARTED'

Next comes the correlation. Here we ask if we have seen this event before for an instance of a process. If all is well, we will NOT have an existing instance and we create a new monitoring context as a result.

PROCESS\_STARTED/EventPointData/mon:correlation/wle:starting-process-instance = processId

▼ Correlation Expression

Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

PROCESS\_STARTED/EventPointData/mon:correlation/wle:starting-process-instance = processId

If no instances are found

Create new instance

If one instance is found

Treat as error

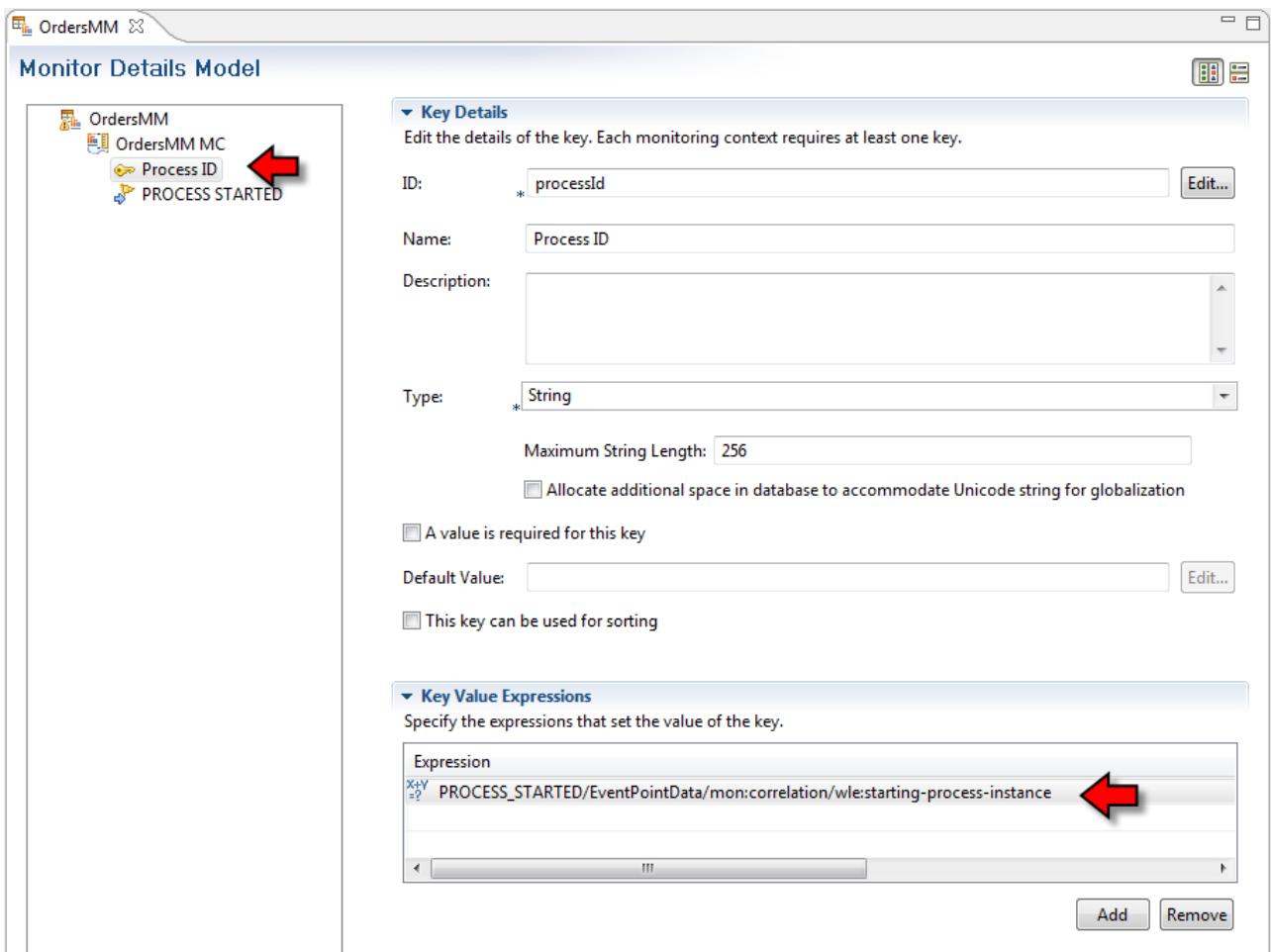
If multiple instances are found

Treat as error

## \_\_\_ 10. Update the Process ID key

In the preceding definition, we added a new inbound event type. Now we update the key metric to when an instance of this event is found. The expression used to source the process instance will be:

PROCESS\_STARTED/EventPointData/mon:correlation/wle:starting-process-instance



## 11. Create the Process Completed inbound event

We have previously defined what constitutes a start process event and seen how this creates a new monitoring context when seen. Each creation of a monitoring context should have a mechanism to complete that context. Now we define an inbound event that will be an indication that the process has completed. We call this event `PROCESS_COMPLETED`. We define the event type details the same as we defined for the `PROCESS_STARTED` event.

**Event Type Details**  
Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both.

Extension name:	Browse...	Clear								
Event parts:	<table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Type</th> <th>Path</th> </tr> </thead> <tbody> <tr> <td>EventPointData</td> <td>EventPointData</td> <td>mon:eventPointData</td> <td>cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData</td> </tr> </tbody> </table>		ID	Name	Type	Path	EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData
ID	Name	Type	Path							
EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData							
	Add	Remove								

The filter condition which indicates that this is a Completion event is shown next.

```
xs:string(PROCESS_COMPLETED/EventPointData/mon:kind) =
'{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS_COMPLETED'
```

**Filter Condition**  
Define a condition based on the event attributes to identify whether to accept an event of this type.

```
xs:string(PROCESS_COMPLETED/EventPointData/mon:kind) = '{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}PROCESS_COMPLETED'
```

Finally, we need to correlate to the correct monitoring context.

PROCESS\_COMPLETED/EventPointData/mon:correlation/wle:starting-process-instance = processId

**Correlation Expression**  
Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

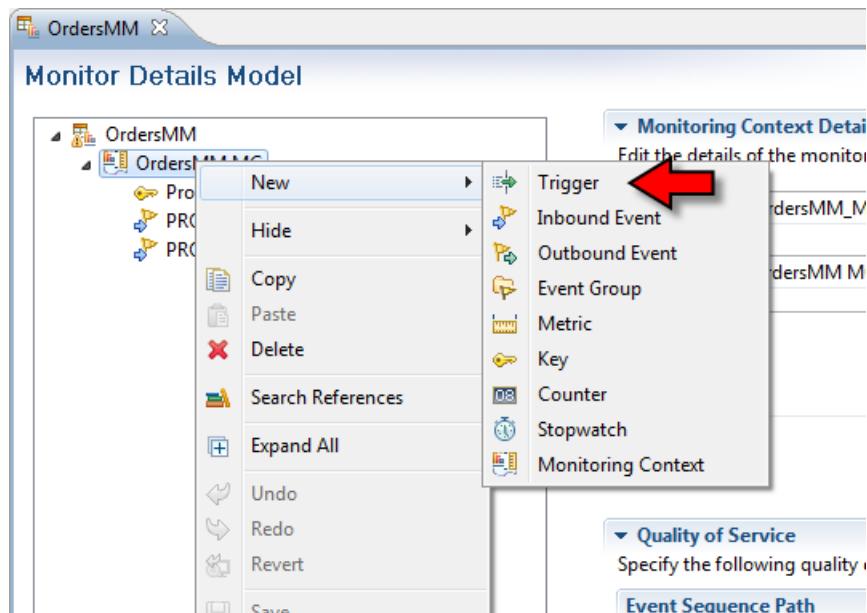
```
PROCESS_COMPLETED/EventPointData/mon:correlation/wle:starting-process-instance = processId
```

If no instances are found: Treat as error  
If one instance is found: Deliver to the instance  
If multiple instances are found: Ignore

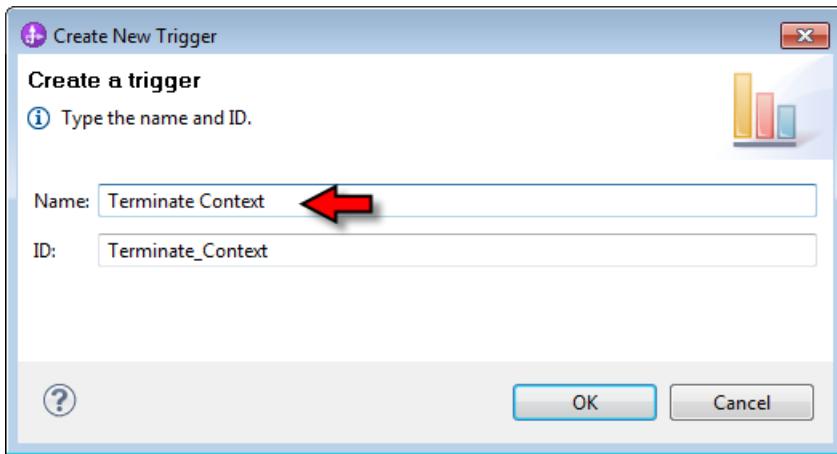
Now when a process completes, the monitor model will see it as a PROCESS\_COMPLETED event.

## 12. Create a termination trigger

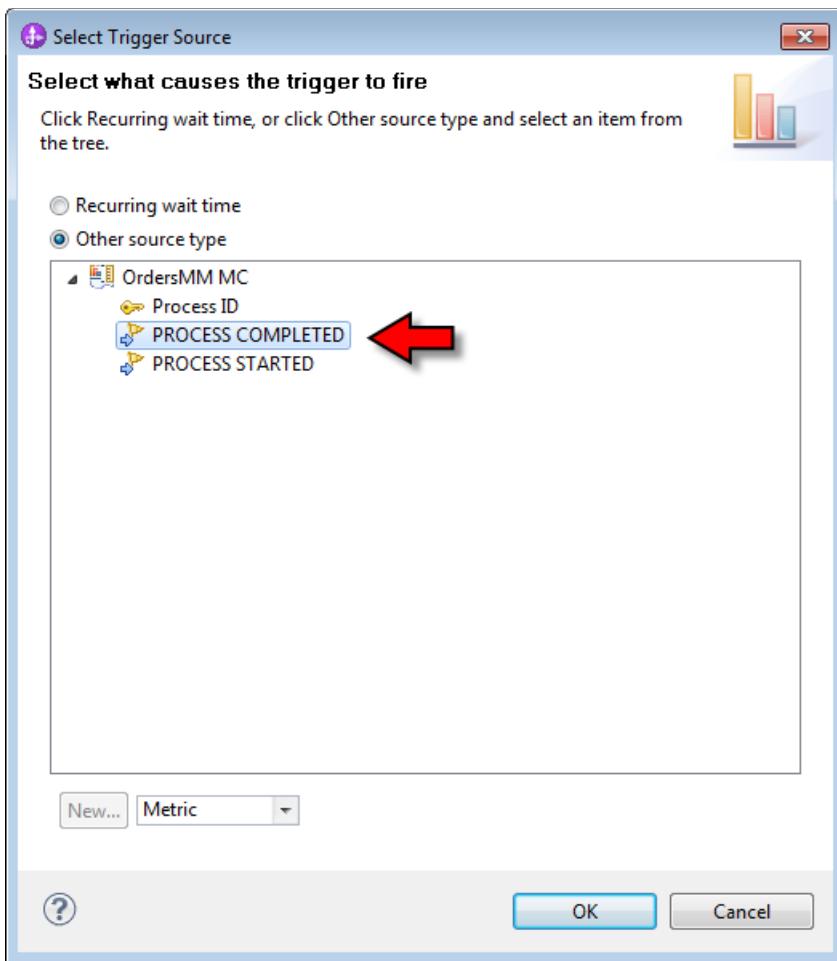
The way we terminate a monitoring context is through a trigger. We create a new trigger definition.



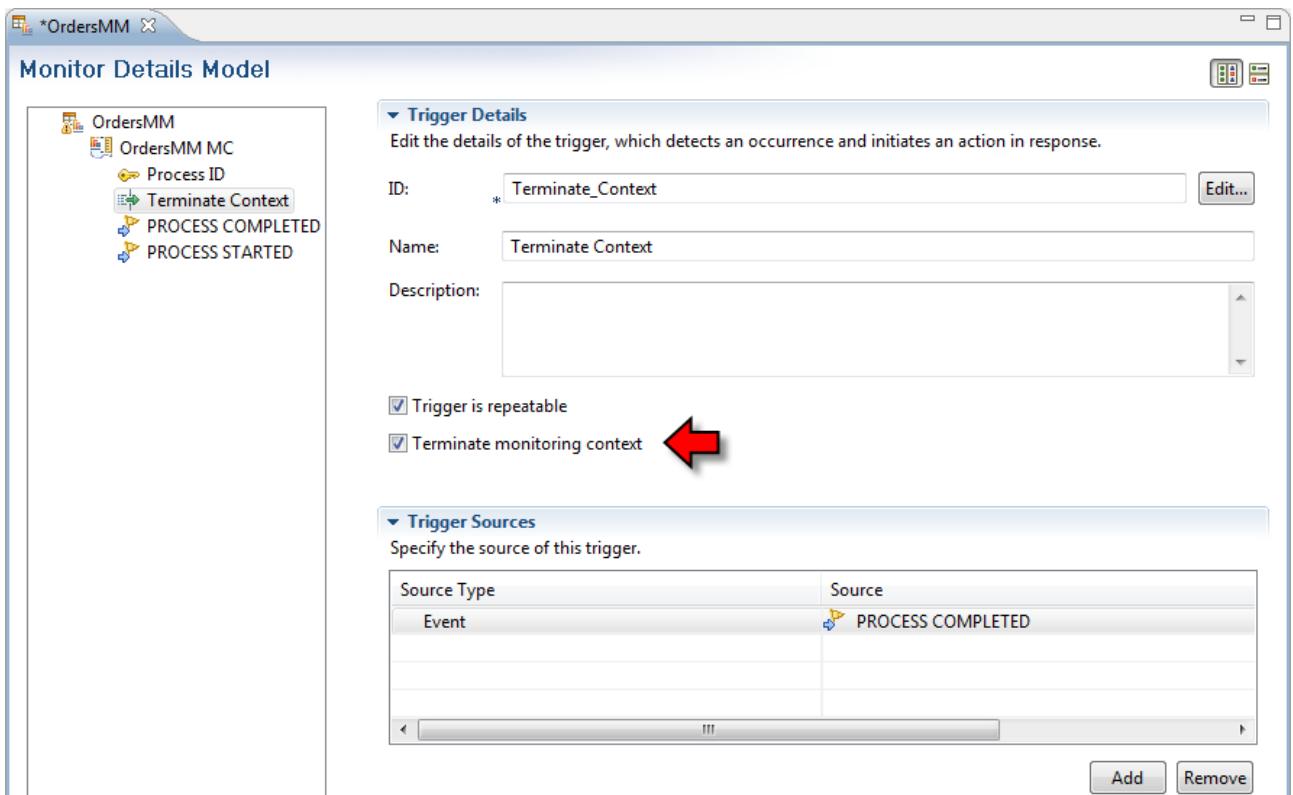
we call the new trigger "Terminate Context".



We next say that the trigger is fired when a PROCESS\_COMPLETED event is detected.



Finally, in the definition of the trigger, we flag it as causing the termination of the monitoring context.

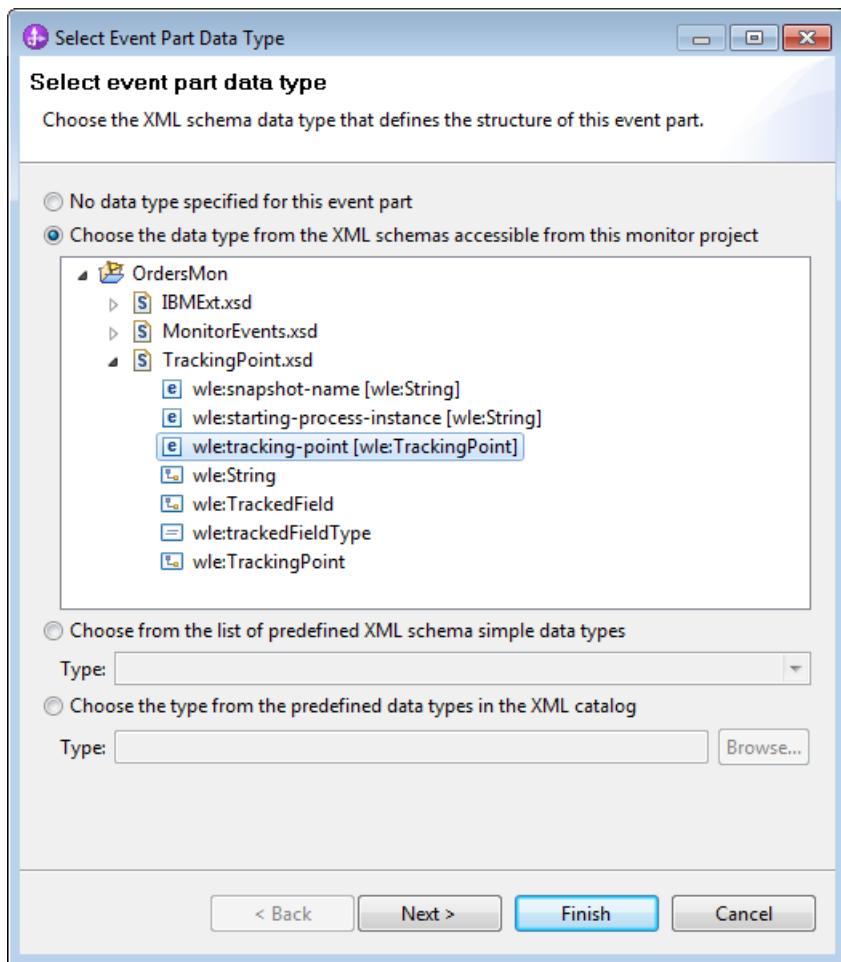


### 13. Create the Tracking Group inbound event

Now we are able to detect the start and end of the process in a monitoring model. It is possible that the recipe we have followed up until now will be repeated in many monitor models.

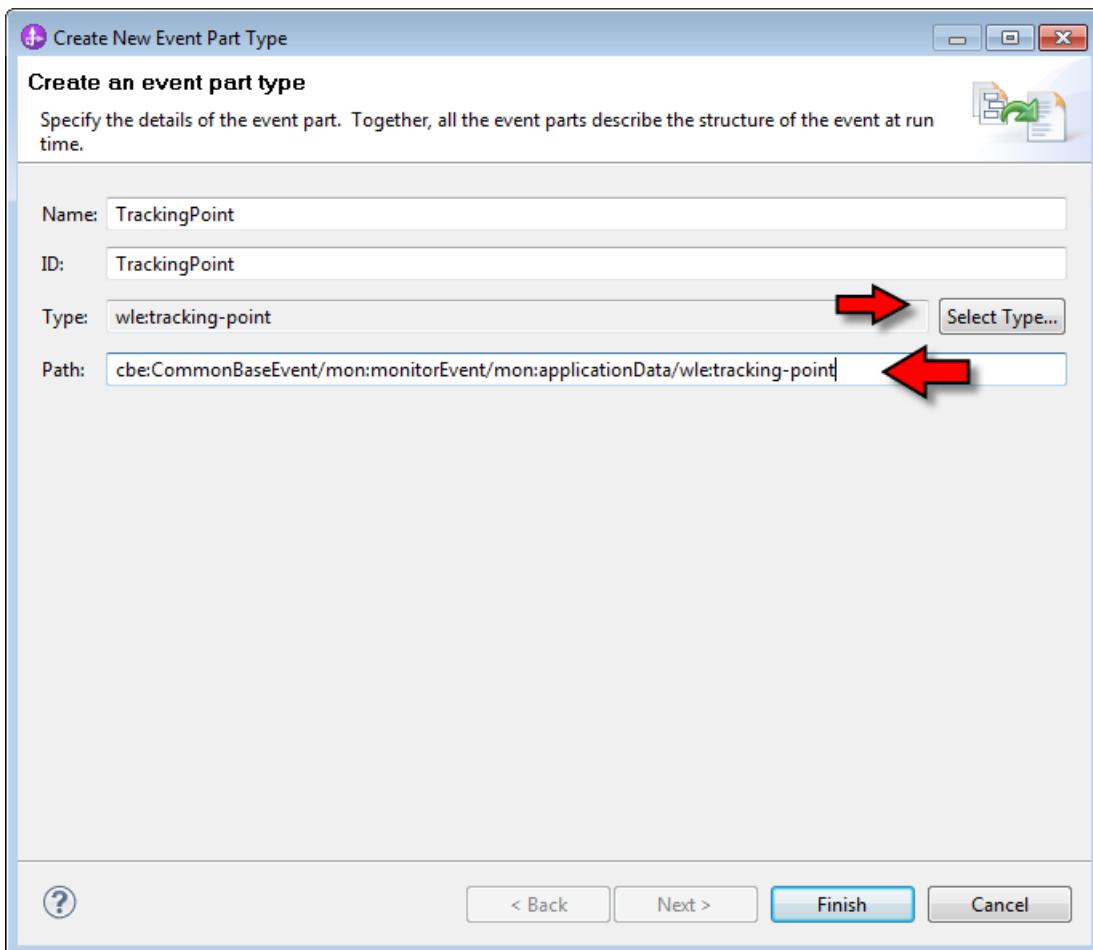
Now we build out more definitions specific for our sample. If we think back we have an intermediate tracking event in our process. We want to be able to catch this type of event so once again, we create a new inbound event definition. We will call this new inbound event "TRACKING GROUP SALE".

Just as before, we add an Event Part for the EventPointData but this time we add a second Event Part for the tracking point definitions. We call this event "TrackingPoint".



The path to this entry will be:

cbe:CommonBaseEvent/mon:monitorEvent/mon:applicationData/wle:tracking-point



The final result for the events parts looks as follows:

**Event Type Details**  
Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both.

Extension name:	Browse...	Clear													
Event parts:	<table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Type</th> <th>Path</th> </tr> </thead> <tbody> <tr> <td>EventPointData</td> <td>EventPointData</td> <td>mon:eventPointData</td> <td>cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData</td> </tr> <tr> <td>TrackingPoint</td> <td>TrackingPoint</td> <td>wle:tracking-point</td> <td>cbe:CommonBaseEvent/mon:monitorEvent/mon:applicationData/wle:tracking-point</td> </tr> </tbody> </table>			ID	Name	Type	Path	EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData	TrackingPoint	TrackingPoint	wle:tracking-point	cbe:CommonBaseEvent/mon:monitorEvent/mon:applicationData/wle:tracking-point
ID	Name	Type	Path												
EventPointData	EventPointData	mon:eventPointData	cbe:CommonBaseEvent/mon:monitorEvent/mon:eventPointData												
TrackingPoint	TrackingPoint	wle:tracking-point	cbe:CommonBaseEvent/mon:monitorEvent/mon:applicationData/wle:tracking-point												
	Add	Remove													

The filter condition that should be applied to an incoming event to see if this is an instance of our tracking event is more superficially complex but at a high level, it is true when:

- The event is an EVENT\_THROWN event
- The source of the event in the BPD is an intermediate tracking group
- The tracking group associated with the intermediate tracking group activity is 'sale'

```

xs:string(TRACKING_GROUP_SALE/EventData/mon:kind) =
'{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}EVENT_THROWN' and
xs:string(TRACKING_GROUP_SALE/EventData/mon:model[1]/@mon:type) =
'{http://schema.omg.org/spec/BPMN/2.0}intermediateThrowEvent' and
TRACKING_GROUP_SALE/TrackingPoint/@wle:groupName = 'sale'

```

**Filter Condition**  
Define a condition based on the event attributes to identify whether to accept an event of this type.

```
xs:string(TRACKING_GROUP_SALE/EventPointData/mon:kind) = '{http://www.ibm.com/xmlns/bpmnx/20100524/BusinessMonitoring}EVENT_THROWN' and
xs:string(TRACKING_GROUP_SALE/EventPointData/mon:model[1]/@mon:type) = '{http://schema.omg.org/spec/BPMN/2.0}intermediateThrowEvent' and
TRACKING_GROUP_SALE/TrackingPoint/@wle:groupName = 'sale'
```

As before, we want to determine which of the possible process instances and hence monitoring contexts the event should be associated with.

TRACKING\_GROUP\_SALE/EventPointData/mon:correlation/wle:starting-process-instance = processId

**Correlation Expression**  
Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

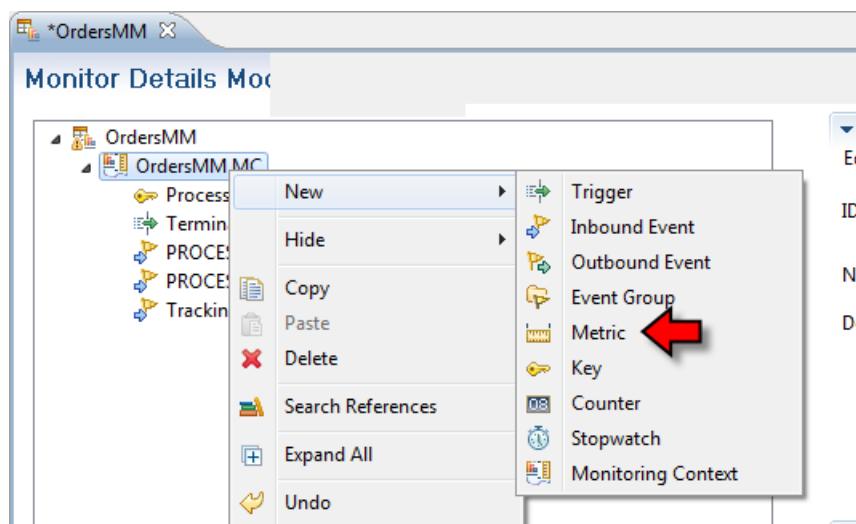
```
TRACKING_GROUP_SALE/EventPointData/mon:correlation/wle:starting-process-instance = processId
```

If no instances are found  If one instance is found  If multiple instances are found

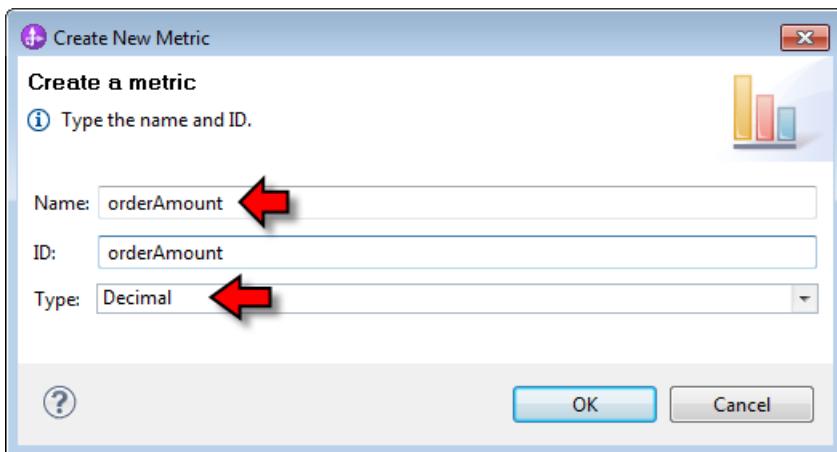
Now we are at the point where we can detect our tracking group event. This is the last of the inbound events we need to work with.

#### 14. Create the "orderAmount" metric

In our model, we wish to create a metric to hold the "orderAmount". We create a new metric.



and give the metric the name "orderAmount" and set its type to be "Decimal".



The metric's value will be taken from the following expression. Note that it names the Tracking Group event as the source and hence will only be evaluated when a Tracking Group event is detected.

```
xs:decimal(TRACKING_GROUP_SALE/TrackingPoint/wle:tracked-field[@wle:name='orderAmount'])
```

**Metric Value Expressions**  
Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.

Trigger	Expression
	xs:decimal(TRACKING_GROUP_SALE/TrackingPoint/wle:tracked-field[@wle:name='orderAmount'])

**Add** **Remove**

## 15. Create the "US State" metric

We create a second metric to hold the US state in which the sale occurred.

```
TRACKING_GROUP_SALE/TrackingPoint/wle:tracked-field[@wle:name = 'usState']
```

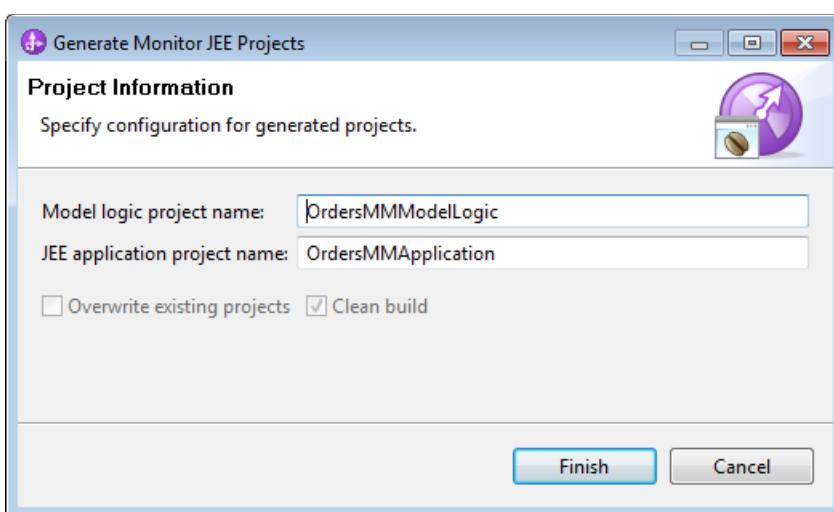
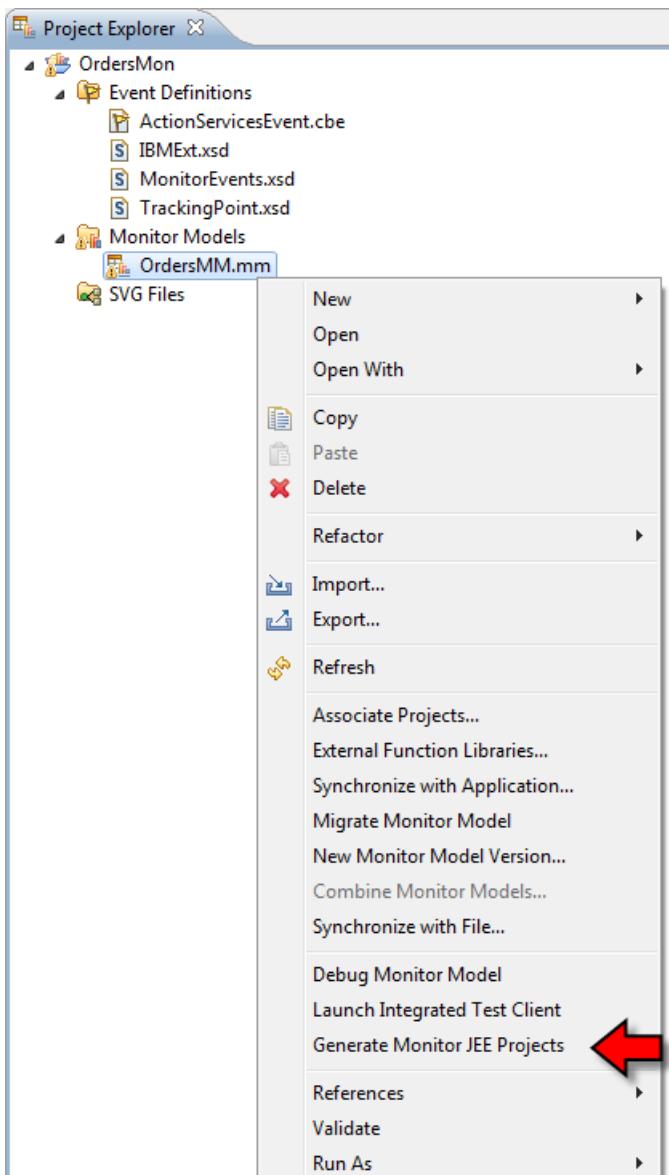
**Metric Value Expressions**  
Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.

Trigger	Expression
	TRACKING_GROUP_SALE/TrackingPoint/wle:tracked-field[@wle:name = 'usState']

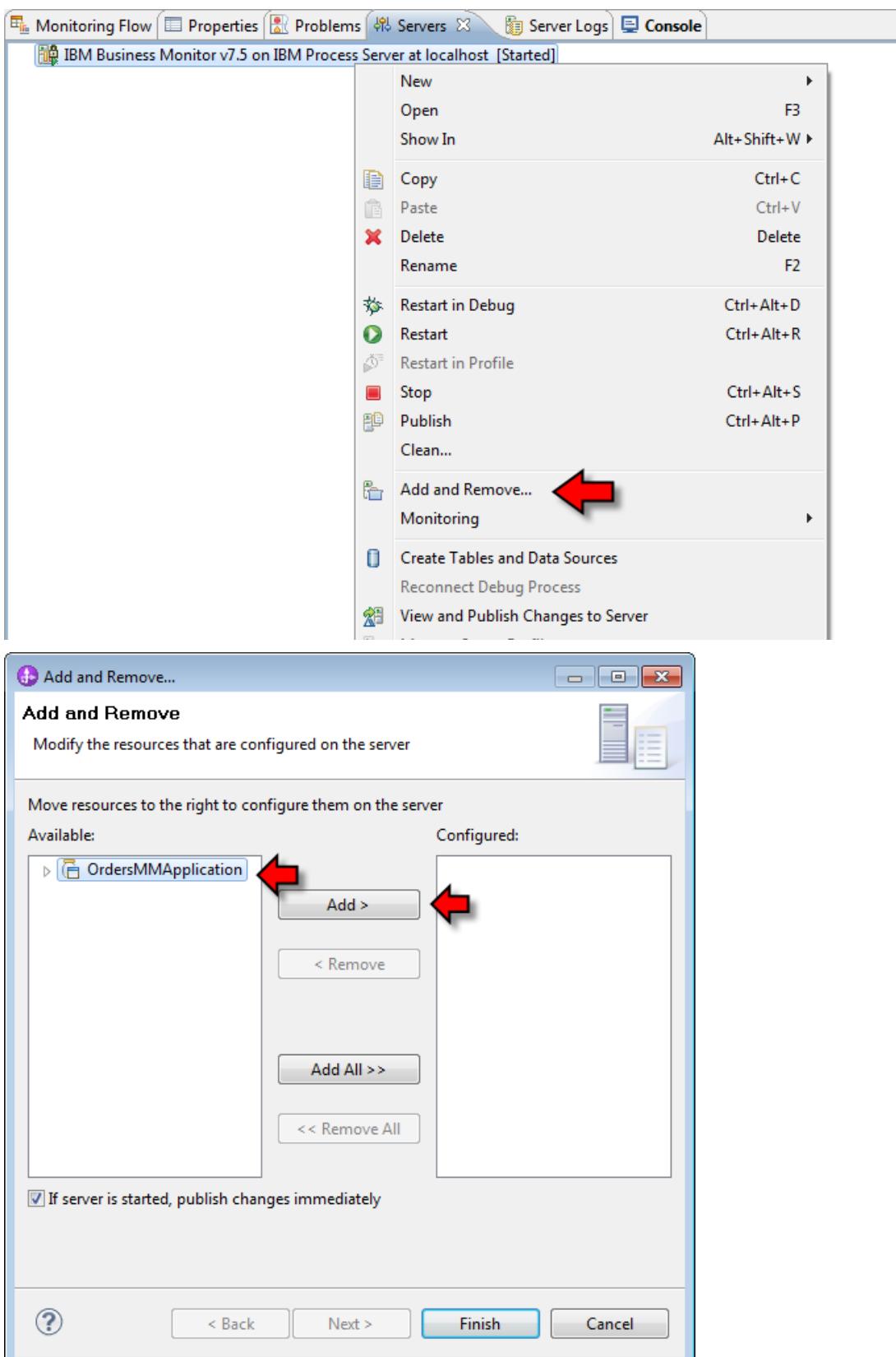
**Add** **Remove**

## 16. Generate and deploy the project

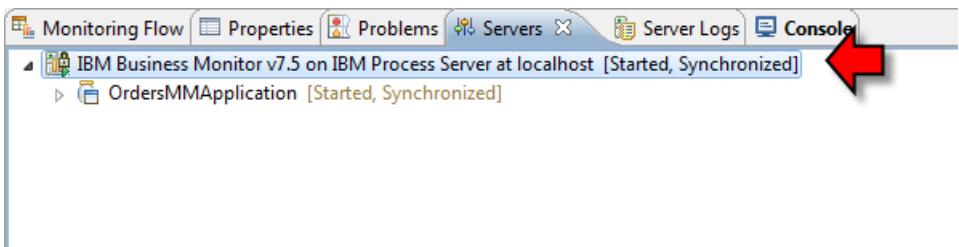
Having completed all the steps in our model's development, what remains for us to do is generate the JEE artifacts and deploy them to a Monitor server.



At the completion of this step, the JEE projects will have been built and will be ready for deployment. We then deploy the project to a Monitor server through the Servers view.

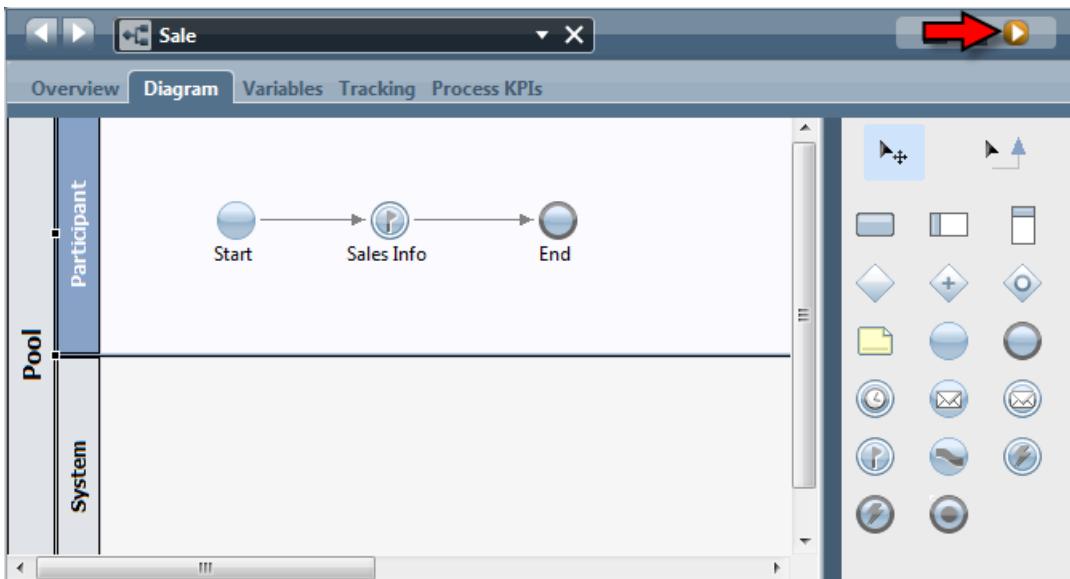


After deployment we will see the application representing the monitor model deployed to the server.



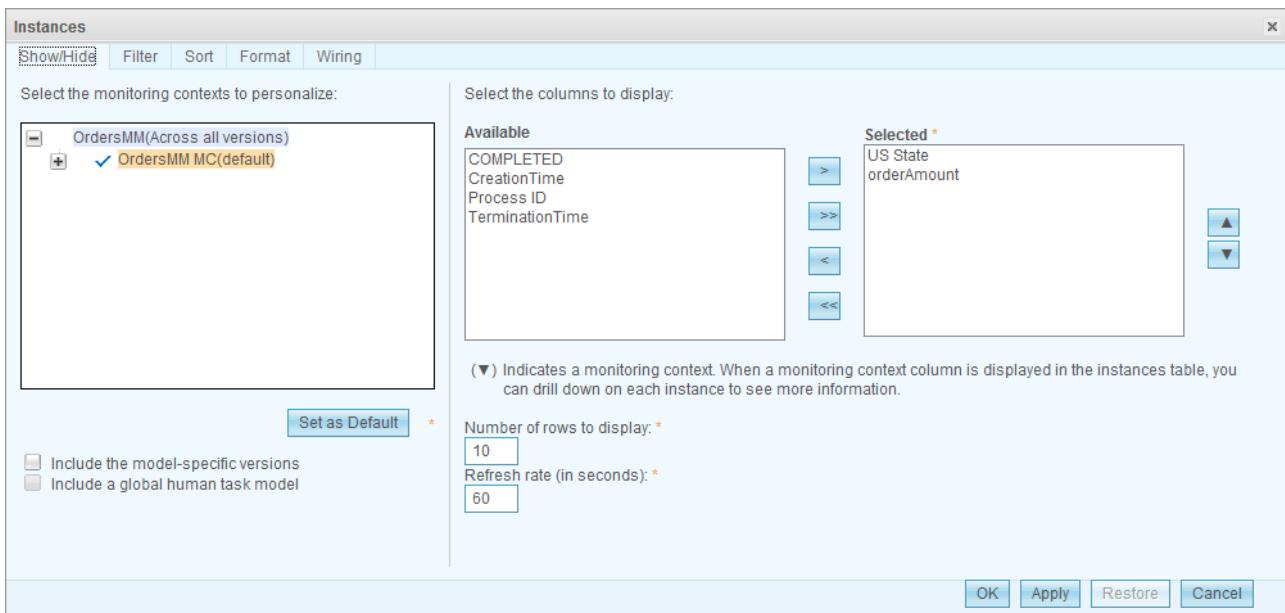
17. Run the process to generate some data

Our construction is finished and now what remains is for us to run an instance of the BPD process.



18. Examine the results in Business Space

After having run an instance of the process, we can bring up Business Space and an a monitoring instances widget to a page. In the configuration of the widget, select the metrics that we build in the model for display.



If all has gone as planned, we will see the following in the widget populated with data.

The screenshot shows the Business Monitor interface with a 'MySpace' workspace. The top navigation bar includes 'Home', 'Go to Spaces', 'Manage Spaces', and 'Actions'. Below the bar, the workspace title is 'page1 Model1'. The main area displays an 'Instances' table. The table header includes 'Export ...' and a dropdown for 'OrdersMM MC'. The table body shows two rows: one for 'US State' with value 'TX' and 'orderAmount' with value '12.34', and another row for 'TX' with the same values. The table has a light gray background with alternating row colors.

And this concludes the tutorial. At this point we have a Business Monitor model that is as clean as can possibly be and contains no superfluous processing. The construction of KPIs, diagrams, cube views and the other powerful capabilities of monitor usage are now as standard with the product with no involvement in its linking to IBPM.

## Glossary

- BPD – Business Process Definition
- BPMN – Business Process Modeling Notation
- EPV – Exposed Process Values
- UCA – Under Cover Agent

## Ideas

Here we have ideas for fun *projects* when otherwise idle:

- There are cases when a process step is to be "worked" offline. Write an application that can obtain an External Activity data, write it to a file ... let some off line app work it ... and then write the file back to the External Activity for completion.
- If a complex object has some of its fields changed, it might be useful to know which ones changed. What we need is a routine that will compare two complex Objects and see what if anything has changed in their content between each other. This could be a Java utility.
- Build a PDF report using the tracking data contained within the Performance Data Warehouse tables. Ideally this should be wizard driven allowing the user to select what kinds of data to output.
  - See: Forum post: [STEW Mailing List: Report on the process variables after the process completion](#)

## Areas to write about and work to do

- Build the "Work" Dashboard using the Coaches
- Run performance testers against Coaches including
  - Firefox Yslow
  - Chrome PageSpeed and Network
- Consider a Coach View widget that has no visuals but instead performs an Ajax call to retrieve data. This could then set its binding data which would then be picked up by another widget such as the the GridX Coach View.
- Create a launcher for Process Designer that uses a list of Process Centers
- How do we dynamically set the timer of a BPD activity? How can it subsequently be changed?
- Research "Groovy" and "Spring"
- Handling the close of a Coach Window – see  
<https://www.ibm.com/developerworks/forums/thread.jspa?threadID=426313&tstart=0>
- Imagine I add a field to a BO and then migrate an in flight process... what if I remove one?
- There is a REST Performance API ... what is that all about?
- Imagine a process has failed ... it is in the failed stated. Is there any way I can make it alive again?

- How do I change the Due Date of a process or task from within JavaScript?
- Fascinating notion on MIL Loops to be studied here:  
<https://www.ibm.com/developerworksforums/thread.jspa?threadID=425272&tstart=0>
- In this thread (<https://www.ibm.com/developerworksforums/thread.jspa?threadID=417448&tstart=0>) there is discussion on returning Java Lists or string[] from Java code. Play with this and see how it works.
- How do we get the content of an attached document through the REST APIs?
- What does it mean to attach a document to a task that has no Process ID???
- In coaches, we seem to be able to suspend/postpone a task and then loop back to it and the data is saved ... can we do something similar with REST APIs?
- There is talk of APIs to “Suspend” processes. What does this mean? What is the state of a process if it is suspended? What of tasks, can they be suspended differently?
- There are questions on generating XML directly from the BOs ... see  
<https://www.ibm.com/developerworksforums/thread.jspa?threadID=411762&tstart=0>
- Look into a tool to dynamically show documentation based on the new asset list technology
- The WAS server appears to have the ability to allow client applications to obtain console logs remotely. How is this done and how can it be used?
- Build a comprehensive Reporting sample that can be used as a tutorial as well as a demonstration that can be explained from soup to nuts
- How to use the SOAP Connector (raw) supplied service?
- How to handle page flow solutions using Custom UI?
- What free CSS editors are out there?
- What is HermesJMS and how can it be used with IBPM?
- How can I make certain rows in a Coach table not to be editable?
- It appears that if I have two tables in two separate sections, only one table's selection is honored, the other is ignored.
- What is involved in writing a VMM plugin
- When a BPD executes, how can I see the list of the activities in the BPD that were actually executed?
- Add into IBPMExplorer the ability to assign/re-assign tasks
- The sendData REST API allows us to change the data in a service. What has to be sent in to change the data for a date/time based variable?
- The Alert functions need re-researched ... are they or are they not part of the product?
- Look into what would be involved in creating a Java Object in one service and making that object available in another service. One possible idea would be the use of the WAS Caching technology:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tdyn\\_distmap.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.express.doc/info/exp/ae/tdyn_distmap.html)

- Date manipulation from JavaScript escaping through LiveConnect
- Sending and receiving help through Process Portal
- Configuring WLE with LDAP
- Using CDL to create a chart background image
- Talk about how scoreboards are visible only by role
- Write about showing work queues for users. We want to see how much work is associated with a particular user ... who is over used or under used. What if a user needs emergency leave, how do we re-own the tasks.
- There is a new feature in 7.2 called "Critical path management". What is that all about?
- We need to go back and add a LOT more flesh on using JMS with WLE
- We can invoke a service from a coach with tw.coach.callService. How can this be leveraged from Dojo? This appears also to be able to return a Java Object instead of XML ... how does it achieve this?
- Build a Dojo list with height
- What is the difference between a Role and a Participant Group
- WSDL Web Service fault handling
- What is a "Task Narrative"?
- Document the use of the interface called "Teamworks" for Java. See:  
<http://www.ibm.com/developerworks/forums/thread.jspa?threadID=360457&tstart=0>

## Quick Links

Here are some links to areas in the book that *I* find especially useful and need to get to in a hurry:

- REST Integration
- Google Web Toolkit – GWT

## The End?

This PDF is always a work in progress. There will always be more to come so check back regularly to see what updates may be found. The plan is to release a new version of the PDF each month usually on the 1<sup>st</sup> of the month.

Hope you found something useful in here.

Neil