



# Progress<sup>®</sup> Rollbase<sup>®</sup> User's Guide



---

# Table of Contents

<b>Copyright.....</b>	<b>25</b>
<b>Chapter 1: Introduction to Progress Rollbase.....</b>	<b>27</b>
Supported Browsers and Secure Access.....	29
Obtaining an Account and Logging In.....	29
Basic Rollbase Concepts.....	30
Navigating the Rollbase Environment.....	31
The Main Screen.....	34
Setup.....	37
The Rollbase Application.....	39
Rollbase Calendar.....	40
Configuring the Calendar.....	41
Day, Week, and Month Views.....	42
Calendar Notifications.....	42
Search.....	43
Wildcard Character Searches.....	43
Boolean Operators.....	44
Escaping Special Characters.....	45
Object Search.....	45
Metadata Search.....	47
Language-Specific Search and Indexing.....	49
Printing and PDF Generation.....	50
Sample Applications.....	51
Updating Applications.....	52
<b>Chapter 2: What is New in Version 3.X.....</b>	<b>53</b>
3.1 Rollbase New Features and Changes.....	53
3.1 Rollbase Mobile New Features and Changes.....	56
3.1 Mobile App Builder New Features and Enhancements.....	57
3.0 New Features and Enhancements.....	60
3.0 Changed Behavior.....	67
3.0 Rollbase Mobile Changes and Improvements.....	68
3.0 Private Cloud Features and Changes.....	74
Upgrading Rollbase Private Cloud.....	76
Upgrading to Version 3.1.....	76
<b>Chapter 3: Designing a Rollbase Application.....</b>	<b>79</b>
Distribution Options.....	80
Web Application Foundation.....	82

---

Object Definition Overview.....	82
Object Attributes.....	84
Business Logic and Customizing the User Experience.....	85
Rollbase UI Components.....	86
Application Page Types.....	87
Application Tabs and Menus.....	92
Tabs on Pages.....	96
Page Cells.....	99

## Chapter 4: Laying the Foundation.....101

Getting Started with the Quick Create Wizard.....	102
Creating and Managing Applications.....	104
Creating an Application.....	104
Editing Applications.....	105
Application Actions.....	107
View Diagram.....	107
Application Tree.....	108
Generate XML.....	109
Update from XML.....	110
Performance Audit.....	110
Custom Sidebar.....	110
Header and Footer.....	111
Set Mobile-Web Options.....	112
Translation.....	113
Attach String Tokens.....	113
Application Permissions.....	113
Deleting an Application.....	113
Installing and Updating from the Application Directory.....	113
Installing and Updating Applications from XML.....	114
Creating and Managing Objects, Fields, and Relationships.....	117
Creating a New Object Definition.....	117
Viewing and Editing an Object Definition.....	121
Adding Fields.....	124
Field Integration Name.....	125
Enabling Field-Level Help.....	126
Field Actions.....	126
Cloning Fields.....	126
Deleting Fields.....	126
Replacing a Picklist.....	127
Converting Field Types.....	127
Field Level Permissions.....	128
Field Validation.....	128
JavaScript Event Handlers.....	129
Adding Fields to Pages, Views, Reports.....	129

---

Deleting an Object Definition.....	129
Views.....	130
Relationships Between Objects.....	132
Page Lookup Fields.....	132
Global Lookup Field Properties.....	133
Related Views.....	134
Related Grid Controls.....	135
Working with Records.....	136
Cloning and New Record Creation.....	136
Cloning Control.....	138
Attaching Related Records to a Newly Created Converted Record.....	138
Protecting Records.....	139
Locking Records.....	139
Condition Formulas.....	139
Record Conversion.....	140
Converting Records.....	141
Converting Records to a Different Type.....	142
Find and Merge Duplicates.....	143
Compare.....	143
Finding Duplicates.....	144
Merging Records.....	144
Auditing.....	145
Send Email.....	146
Changing the Owner of an Object.....	147
Updating Multiple Records.....	147
Tagging Records.....	148
Orphan Records.....	149
Recurring Calandar Events and Tasks.....	150

## **Chapter 5: Adding Business Logic .....151**

Working with Templates.....	152
Adding Templates to a Page.....	153
Adding HTML Components to a Page.....	153
Adding Script Components to a Page.....	155
Adding Template Fields and Integration Links to an Object.....	157
Creating a Record Name Template.....	157
Template Token Syntax.....	158
Common Tokens.....	158
Advanced Tokens.....	160
Iterating through Records.....	162
Loop Through Specific Number of Records, Comments, and Activity Trails.....	163
Loop through All Records .....	163
Using EVAL Blocks.....	164
Mail Templates.....	164

---

Document Templates.....	166
Communication Logs.....	170
Localization.....	171
New Record Template.....	171
Formulas.....	172
Writing and Debugging Formulas.....	173
Formula Return Types.....	174
Examples of Valid String Tokens.....	175
Using Dates in Formulas.....	175
Example Using Images to Represent Record Status.....	176
Formula Execution Limits.....	176
Group Functions.....	177
Typical Mistakes in Formulas.....	178
Triggers and Workflows.....	179
Trigger Overview.....	179
Trigger Rules and Restrictions.....	182
Trigger Timing Options.....	182
Best Practices for Trigger Formulas.....	183
Delayed and Repeating Triggers.....	184
Creating a Trigger.....	185
Trigger Types.....	186
Send Email.....	186
Create Audit Trail Record.....	187
Validate Record Data.....	187
Unique Fields Combination.....	187
Update Field Value.....	187
Change Workflow Status.....	189
Create New Record.....	189
Attach Related Record.....	189
Create Template Document.....	190
Run Triggers on Related Record.....	190
Object Script.....	190
HTTP Triggers.....	191
Send HTTP Post.....	191
Send HTTP Get Request.....	192
Send SMS Message.....	192
Example: Set Field Based on a Workflow Status Change.....	193
Debugging Complex Triggers.....	193
Debugging Delayed Triggers.....	194
Workflow Overview.....	194
Workflow Status.....	195
Workflow Actions.....	195
Workflow Action Properties.....	197
Group Action Properties.....	198
Workflow Processes.....	198

---

---

Creating a Workflow Process.....	199
Editing and Viewing a Workflow Process.....	200
Approvals.....	200
Record Queues.....	203
Reports, Charts, and Gauges.....	203
Working with Reports.....	204
Tabular Reports.....	204
Document Template Reports.....	206
HTML Template Reports.....	206
JavaScript Reports.....	207
Running Reports.....	207
Merging Reports.....	209
PDF Report Options.....	209
Report Batch Jobs.....	210
Charts.....	210
Creating Charts.....	210
Using Charts.....	212
Debugging Charts.....	213
Gauges.....	213
Creating and Configuring Gauges.....	213
Using Gauges.....	214
Multi-Currency Support.....	214
Surveys and Quizzes.....	217
Creating a Survey.....	217
Survey Questions Library.....	217
Adding Survey Questions.....	219
Survey Pages and Links.....	220
Taking a Survey.....	220
Using Surveys on Portals.....	221

## **Chapter 6: Customizing the User Experience.....223**

Pages, the Page Editor, and Grid Controls.....	223
Editing Pages.....	224
Adding Columns and Changing Alignment.....	226
Buttons.....	227
Using Grid Controls to Manage Multiple Records.....	227
Adding a Grid Control with the Page Editor.....	228
Configuring a Grid Component.....	229
Using a Grid Component.....	231
Embedded Quick Create.....	232
Detailed Search.....	233
Customizing the Header and Footer.....	234
Working with Views.....	235
Creating and Editing Views.....	236

---

Adding Columns.....	238
Grouping and Sorting.....	238
Totalling Columns.....	239
Filtering by Dates.....	240
Filter Criteria.....	240
Inline Editing.....	242
Filtering by Formula.....	243
Filtering by Search criteria.....	243
Dynamic Filtering.....	245
Filter Results Color Code.....	245
Programmatic Client-side Customization.....	247
HTML Event Handlers.....	247
Defining Event Handlers.....	248
Copying a Field's Value to Other Fields.....	250
Disabling Fields.....	250
Setting Default Values.....	251
Using System Settings to Define Values that Might Change.....	252
Rollbase AJAX APIs.....	253
Examples.....	253
Showing or Hiding a Page Section.....	253
A Simple Lookup.....	254
A Financial Calculation.....	255
Access Control.....	256
Rollbase Portals.....	257
Creating a Portal.....	258
Creating a Custom Header and Footer.....	260
Creating Portal Pages.....	261
Portal Page Properties.....	262
Portal Page URLs.....	262
Using an EVAL Block on a Portal Page.....	263
Portal Security.....	264
Creating Portals Without Authentication.....	265
Creating Portals With Authentication.....	265
Hosted Files.....	267
Managing Hosted Files.....	268
Hosted File Tokens.....	268
Using Hosted File Tokens.....	269

## **Chapter 7: Supporting Mobile Users.....**

**271**

Rollbase Mobile.....	272
Overview.....	273
The Mobile App Builder project.....	274
Hosting and deployment options.....	277
Mobile run-time architecture.....	277

---

Hybrid apps and browser apps.....	277
Mobile access to Rollbase objects using JSDOs.....	278
Mobile login sessions and user access.....	280
Working with Push Notifications and other Mobile features.....	280
Simple Rollbase Mobile App Example.....	281
Creating the Mobile App Definition.....	282
Viewing the Default UI.....	284
Editing the default pages.....	286
Adding components to display records.....	289
Adding data sources to retrieve records.....	293
Adding Events to invoke the Services.....	297
Testing the Example Mobile App.....	299
Creating and updating Rollbase Mobile Apps.....	300
Enabling Mobile App Development in Rollbase Private Cloud.....	300
Creating a Rollbase Mobile App project.....	302
Importing Mobile App projects created with OpenEdge Mobile.....	303
Editing Rollbase Mobile App projects.....	306
Adding objects to a Mobile App.....	309
Testing Rollbase Mobile Apps.....	309
Packaging and Deployment Options.....	310
Packaging and deploying hosted browser apps.....	310
Packaging and deploying Apps for iOS.....	310
Packaging iOS Apps.....	311
Testing the IPA directly on an iOS device.....	311
Deploying iOS Apps.....	312
Apple Developer.....	312
iOS Provisioning Portal.....	312
Certificates.....	312
App IDs.....	312
Devices.....	313
Provisioning Profiles.....	313
Packaging and deploying Apps for Android.....	313
Packaging Android Apps.....	313
Testing the APK directly on an Android device.....	315
Deploying Android Apps.....	315
Versioning your Mobile App.....	315
Package name.....	315
Signing your application.....	315
Mobile-Web Enabled applications.....	316
<b>Chapter 8: Integrating with Outside Sources.....</b>	<b>321</b>
Creating Rollbase Objects from OpenEdge Services.....	322
Limitations.....	322
Supported Data Types.....	323

---

---

Linking a Rollbase Object to OpenEdge Data.....	324
Creating an Application from OpenEdge Data.....	326
Enabling support for filtering options and sorting.....	328
Enabling object attributes for an OpenEdge Service Object.....	340
Using DataDirect Cloud to Access External Data.....	343
Creating Rollbase Applications from Microsoft Access.....	345
Upload the MDB File.....	345
Create Objects from MDB Tables.....	346
Create Fields and Records.....	347
Review Results.....	349
Creating Rollbase Applications from Salesforce Applications.....	350
Migrating the Application.....	351
Using External Tables as Rollbase Objects.....	352
Using an External Database and External Objects for Private Cloud.....	352
External Object Overview.....	353
External Object Fields and Attributes.....	353
External Relationships.....	354
SQL Queries for External Objects.....	355
Creating an External Object from an External Database Table.....	357
Importing Data.....	359
Compatible Import Data Types.....	360
Importing for Existing Objects.....	361
Importing to Create a New Object.....	363
Importing Related Objects.....	365
Deleting Multiple Records by Importing a Spreadsheet.....	366
Exporting from Views and Reports.....	367
Integrating with Google Applications.....	367
Incoming Gmail.....	367
Outgoing Gmail.....	368
Google Spreadsheets.....	369
Google Calendar.....	369
Google Maps.....	370
Using SOAP or REST to Integrate with Rollbase.....	371
Limits on API Calls.....	372
Monitoring SOAP Calls.....	373
Monitoring REST Calls.....	373
<b>Chapter 9: Security and Access Control.....</b>	<b>375</b>
Supported Methods of Authenticating Users.....	376
Built-in Security Levels.....	377
Setting the Authentication Method.....	378
LDAP Authentication details.....	380
LDAP Advanced Authentication details.....	381
HTTP POST Authentication details.....	383

---

---

HTTP GET Authentication details.....	384
OpenEdge Authentication details.....	384
Rollbase User Authentication.....	386
Setting and Changing Security Levels.....	386
Enabling Single Click Log In.....	387
Password Expiration Policies.....	388
Custom Validation Rules.....	388
Whitelist IP Addresses.....	388
Security Questions for Authentication.....	389
Forgotten Password.....	391
Portal Security.....	391
External Authentication.....	391
External System Single Sign On Example.....	392
Verifying External Authentication.....	393
User Roles and Permissions.....	393
User Roles.....	394
User Hierarchy of Permissions.....	395
Managing User Roles.....	396
Private Attribute.....	398
Component-level Permissions.....	398
Permissions Through Relationships.....	399
Field Level Permissions.....	399
Record Creator Pseudo Role.....	400
Page Versions.....	401
Location/Department/Function Permissions.....	402
LDF Field Default Values.....	403
LDF Groups.....	404
Setting Up LDF.....	405
Enabling an administrative user to log into a customer tenant.....	406

## **Chapter 10: Publishing and Distributing Applications.....407**

Design and Development Considerations for Distributing as XML.....	408
Components Included in an Application XML File.....	409
Use of Original IDs.....	410
Locking Applications.....	411
Attaching Seed Records.....	412
Providing a Test Drive.....	412
Testing and Verifying Application Correctness.....	413
Administrative Management of Published Applications.....	413
Generating Application XML.....	413
Publishing to the Rollbase Application Directory.....	415
Troubleshooting Published Applications.....	417

---

## Chapter 11: Advanced Setup and Administration.....421

Personal Setup.....	422
Administration Setup.....	422
Transfer Owners.....	423
Using Company-wide Settings.....	424
Calculating Sales Tax Example.....	424
Using Scripts to Change Setting Fields.....	425
Account Settings.....	425
Administrative Settings.....	426
Backup and Restore.....	426
Batch Jobs.....	428
Date Formats.....	430
Currency Formats.....	430
Billing and Support Settings.....	431
Language Support.....	432
Improving Translation Resources.....	436
Adding Support for Other Languages in Private Cloud.....	436
Translating Applications.....	437
Global Text Search.....	438
Support.....	438

## Chapter 12: Installing and Administering Private Cloud.....441

Introduction.....	441
Supported Platforms.....	443
Licensing.....	444
Packaged OpenEdge License Restrictions.....	445
Private Cloud Updates.....	445
Included Rollbase Applications.....	445
Third Party Software You Can Install.....	446
PDF Converter.....	446
JExcel API.....	447
Aspose.Words for Java.....	447
Aspose.Pdf for Java.....	447
StelsMDB Access JDBC Driver.....	448
FusionCharts.....	448
FusionWidgets.....	448
Installation.....	449
Pre-Requisites.....	449
Using the Rollbase Installer.....	450
Post-Requisites.....	452
Using Your Own Instance of Tomcat .....	452
Setting Up Rollbase Manually.....	453

---

Download and Unzip Rollbase Components.....	453
Set Environment Variables.....	454
Configuring a Supported Database.....	455
MySQL.....	456
OpenEdge.....	457
Oracle.....	458
SQL Server.....	458
Edit databases.xml.....	458
Starting Components and Logging In.....	460
Starting Components on Windows Systems.....	460
Starting Components on Linux Systems.....	461
Activating Your License.....	462
Upgrading from an Evaluation License.....	462
Upgrading a License Without Restarting.....	462
Troubleshooting.....	463
Installation Issues.....	463
License Error.....	464
Email Issues.....	465
Logging In.....	465
Administration.....	465
Overview.....	466
Monitoring System Components.....	468
Managing Customer Tenants.....	469
Creating a New Customer Record.....	470
Working with Customer Records.....	471
Moving and Restoring Customer Tenants.....	473
Enabling Logging for Charts and Views.....	474
Managing Databases.....	476
Adding a New Database for Use with Customer Tenants.....	476
Creating Custom Database Indexes.....	478
Adding Columns to a Private Cloud Database.....	478
Applications Directory and Support Portal.....	479
Test Drive.....	479
Set Up ISV Partners.....	480
Multi-server Environments.....	480
Planning Your Multi-server Architecture.....	481
Distributing Load with PAS.....	481
Working with Instances.....	482
Creating instances with TCMAN.....	483
Instance management with TCMAN.....	485
Installing and running an instance as a Windows service.....	485
Installing and running a PAS instance as a Linux daemon.....	486
Distributing Load with Apache and Tomcat.....	487
Configuring Apache and Tomcat for Private Cloud.....	488
Install the JK Module.....	488

---

Create a workers.properties File.....	488
Configure the httpd.conf File.....	489
Apache Troubleshooting.....	490
Configuring Multiple Instances of Rollbase Components .....	490
Adding Production Servers.....	492
Adding Auxiliary Servers.....	493
Assigning a Customer to a Dedicated Production Server.....	495
Load Balancing a Large Customer Tenant.....	496
Configuration File Reference.....	496
The components.xml File.....	497
Component Specific Properties.....	498
databases.xml.....	500
events.xml.....	502
fieldgroups.xml.....	502
legacyobjects.xml.....	502
listitems.xml.....	502
license.xml.....	503
securitylevel.xml.....	503
servicelevel.xml.....	504
shared.properties.....	505
PAS Command Line Reference.....	514
The tcman command.....	514
Manager actions.....	516
List deployed applications (list).....	516
Display OS and server information (info).....	517
Deploy a Web application (deploy).....	518
Undeploy a Web application (undeploy).....	519
Reload a Web application (reload).....	520
Display detailed server status (status).....	521
Display memory leaks (leaks).....	522
Start a Web application (enable).....	523
Stop a Web application (disable).....	524
Display global server resources (resources).....	525
Display Web application HTTP sessions (sessions).....	525
Server actions.....	526
Create an instance (create).....	526
Delete an instance (delete).....	528
Display and manage an instance's configuration (config).....	529
Display or modify the server features of an instance (feature).....	531
Clean up or archive server log files (clean).....	532
Display server instances (instances).....	533
Register an instance for tracking (register).....	535
Stop tracking an instance (unregister).....	536
Start an instance (start).....	536
Stop an instance (stop).....	538

---

---

Display server, OS, and runtime version information (version).....	539
Test a server configuration (test).....	540
General actions.....	541
Display help (help).....	541
Display runtime environment information (env).....	542

## **Chapter 13: Setup and Administration for ISVs.....545**

Getting Started.....	546
System Applications.....	548
Creating a Custom Log In Page.....	548
Creating a Page for Users to Retrieve Passwords.....	549
Customizing Page Title Tags.....	550
Using a Third-Party Cloud Service for Storage.....	551
Using Amazon S3.....	551
Using Microsoft Azure.....	553
Using the ISV Partner Application.....	554
Creating and Managing Customer Tenants.....	555
Pushing Application Updates to Other Tenants.....	555
Installing Application Updates.....	556
Version History and Rolling Back.....	557

## **Chapter 14: Reference.....559**

Server-side API.....	559
API Error Messages.....	561
Query API.....	561
rbv_api.selectQuery().....	565
rbv_api.selectQuery2().....	566
rbv_api.selectValue().....	567
rbv_api.selectNumber().....	568
rbv_api.selectCustomerQuery().....	568
rbv_api.selectCustomerValue().....	569
rbv_api.selectCustomerNumber().....	570
rbv_api.getCount().....	571
rbv_api.getRelatedIds().....	571
rbv_api.getRelatedIds2().....	572
rbv_api.getRelatedFields().....	573
rbv_api.getRelatedFields2().....	574
Object Script API.....	574
Object Script Example.....	575
rbv_api.getFieldValue().....	575
rbv_api.getNumFieldValue().....	576
rbv_api.getBinaryData().....	577
rbv_api.getTextData().....	577

---

rbv_api.isEmpty().....	578
rbv_api.setFieldValue().....	579
rbv_api.setBinaryFieldValue().....	580
rbv_api.setTextFieldValue().....	581
rbv_api.createRecord().....	582
rbv_api.updateRecord().....	583
rbv_api.attach().....	584
rbv_api.detach().....	584
rbv_api.runTrigger().....	585
rbv_api.deleteRecord().....	586
rbv_api.setCreator().....	587
User Selection API.....	587
rbv_api.isViewed().....	588
rbv_api.setViewed().....	588
rbv_api.isFlagged().....	589
rbv_api.setFlagged().....	589
rbv_api.getSelectedIds().....	590
rbv_api.isAppInstalled().....	591
Miscellaneous Methods.....	591
rbv_api.getIdByCode().....	591
rbv_api.getCodeById().....	592
rbv_api.getValueById().....	592
rbv_api.runReport().....	593
rbv_api.runTemplate().....	594
Date, Time, and Currency API.....	594
rbv_api.getCurrentDate().....	594
rbv_api.firstDayOfMonth().....	595
rbv_api.firstDayOfQuarter().....	595
rbv_api.firstDayOfYear().....	596
rbv_api.formatDate().....	597
rbv_api.formatNumber().....	597
rbv_api.formatCurrency().....	598
rbv_api.getExchangeRate().....	599
PDF Processing API.....	600
rbv_api.getPDFProperty().....	600
rbv_api.concatPDF().....	600
Hosted File API.....	601
rbv_api.getHostedAsText().....	601
rbv_api.getHostedAsBinary().....	602
HTTP API.....	602
rbv_api.sendHttpGet().....	602
rbv_api.sendHttpPost().....	603
rbv_api.sendJSONRequest().....	604
rbv_api.getHTTPParameter().....	605
XML Processing API.....	605

---

---

rbv_api.parseXML()	605
rbv_api.evalXpath()	606
JSON Processing API.....	607
rbv_api.jsonToString()	607
rbv_api.stringToJson()	607
Trigger API.....	608
rbv_api.setSharedValue()	608
rbv_api.getSharedValue()	609
rbv_api.isUI()	610
rbv_api.isPortal()	610
rbv_api.isMobile()	611
rbv_api.isAPI()	612
rbv_api.isDelayed()	612
rbv_api.isImport()	613
rbv_api.isCreate()	613
rbv_api.isUpdate()	614
rbv_api.isDelete()	614
Debugging API.....	615
rbv_api.print()	615
rbv_api.println()	616
rbv_api.printARR()	617
rbv_api.setVerbose()	617
rbv_api.isVerbose()	618
rbv_api.inArgs()	618
Log API.....	619
rbv_api.createActivityLog()	619
rbv_api.log()	619
Email API.....	620
rbv_api.openIMAP()	620
rbv_api.openPOP3()	621
rbv_api.getMailMessageCount()	622
rbv_api.getNewMailMessageCount()	622
rbv_api.getMailMessage()	622
rbv_api.getMailMessages()	623
rbv_api.closeMailSession()	624
Client-side AJAX API.....	624
Queries.....	624
rbf_createRecord()	625
rbf_deleteRecord()	626
rbf_getCount()	626
rbf_getCount2()	627
rbf_getFields()	628
rbf_getPage()	629
rbf_getPage2()	630
rbf_getRelatedFields()	631

---

rbf_getRelatedFields2()	632
rbf_getRelatedIds()	633
rbf_getRelatedIds2()	633
rbf_runTrigger()	634
rbf_selectNumber()	635
rbf_selectQuery()	636
rbf_selectQuery2()	636
rbf_selectValue()	637
rbf_setField()	638
rbf_updateRecord()	638
Field Manipulation	639
rbf_getFieldContent()	639
rbf_getFieldValue()	640
rbf_getPicklistCode()	640
rbf_getPicklistCodes()	641
rbf_setFieldContent()	641
rbf_setFieldValue()	642
rbf_setFieldDisabled()	642
rbf_setPicklistCode()	643
Data Formatting	643
rbf_getDate()	643
rbf_getDigits()	644
rbf_getFloat()	644
rbf_formatCurrency()	645
rbf_formatDate()	645
rbf_formatUsingMask()	646
rbf_getInt()	646
Grid Control Examples and API	647
Simple Grid Calculation Example	648
Advanced Grid Example	650
rbf_addGridRow()	651
rbf_delGridRow()	652
rbf_getGridField()	652
rbf_getGridPicklistCode	653
rbf_getGridPicklistControl()	653
rbf_getGridValue2()	654
rbf_getMaxRowIndex2()	654
rbf_setGridContent2()	655
rbf_setGridPicklistCode()	656
rbf_setGridValue2()	656
rbf_showGridTotals()	657
rbf_sumGridColumn2()	657
Miscellaneous	658
rbf_getCodeById()	658
rbf_getExchangeRate()	659

---

---

rbf_getIdByCode().....	659
rbf_isChecked().....	660
rbf_isEmpty().....	660
rbf_isSelected().....	661
rbf_isZero().....	661
rbf_setChecked().....	662
rbf_setErrorsCallback().....	663
rbf_setLookupFilter().....	663
rbf_setSelected().....	664
rbf_startServerDebugging().....	665
rbf_stopServerDebugging().....	665
AJAX Metadata API .....	666
rbf_createApplicationDef().....	666
rbf_createFieldDef().....	667
rbf_createObjectDef().....	668
rbf_createRelationshipDef().....	668
rbf_deleteApplicationDef().....	669
rbf_deleteFieldDef().....	670
rbf_deleteObjectDef().....	670
rbf_deleteRelationshipDef().....	671
rbf_getApplicationDef().....	672
rbf_getFieldDef().....	672
rbf_getObjectDef().....	673
rbf_getRelationshipDef().....	674
rbf_updateApplicationDef().....	674
rbf_updateFieldDef().....	675
rbf_updateObjectDef().....	676
rbf_updateRelationshipDef().....	677
Display Functions.....	677
rbf_activatePageTab().....	677
rbf_getSectionIdByTitle().....	678
rbf_getViewSelector().....	678
rbf_growl().....	679
rbf_growlError().....	679
rbf_hideGrowl().....	680
rbf_hideInfoMessage().....	680
rbf_growlInfo().....	680
rbf_growlSuccess().....	681
rbf_growlWarning().....	682
rbf_showInfoMessage().....	682
rbf_showMessage().....	683
rbf_showOrHidePageTab().....	683
rbf_showOrHideSection().....	684
rbf_setSectionCollapse().....	684
rbf_setViewSelector().....	685

---

Code Generator.....	685
Using the Code Generator.....	686
Metadata API and XML Reference.....	687
Metadata XML Reference.....	687
Application XML Elements.....	688
Object XML Definition.....	690
Field XML Definition.....	695
Relationship XML Definition.....	703
SOAP Metadata Methods.....	706
createApplicationDef().....	706
createObjectDef().....	707
createFieldDef().....	707
createRelationshipDef().....	708
deleteApplicationDef.....	709
deleteFieldDef().....	710
deleteObjectDef().....	711
deleteRelationshipDef().....	711
getApplicationDef.....	712
getFieldDef().....	713
getObjectDef() .....	714
getObjectDefNames() .....	714
getRelationshipDef().....	715
metadataSearch().....	716
updateApplicationDef().....	716
updateFieldDef().....	717
updateObjectDef().....	718
updateRelationshipDef().....	719
REST Metadata Methods.....	720
createApplicationDef.....	720
createFieldDef.....	720
createObjectDef.....	721
createRelationshipDef.....	722
deleteApplicationDef.....	722
deleteFieldDef.....	723
deleteObjectDef.....	724
deleteRelationshipDef.....	724
getApplicationDef.....	725
getFieldDef.....	726
getObjectDef.....	726
getObjectDefNames.....	727
getRelationshipDef.....	728
metadataSearch.....	729
updateApplicationDef.....	730
updateFieldDef.....	730
updateObjectDef.....	731

---

---

updateRelationshipDef.....	732
Rollbase REST Methods.....	732
appXML .....	733
bulkCreate .....	733
bulkCreateOrUpdate .....	734
bulkDelete .....	735
bulkUpdate .....	736
clearDataObjectCache.....	737
create .....	738
createArr.....	739
createCustomer .....	741
create2 .....	742
createRecord.....	743
delete .....	744
deleteArr .....	745
deleteRecord.....	746
getApplicationIds.....	747
getBinaryData.....	748
getBuildStatus.....	749
getCodeById .....	749
getCount .....	750
getDataField .....	751
getDataObj .....	753
getIdByCode .....	755
getPage .....	756
getRecord.....	757
getRelationships .....	759
getUpdated .....	760
header .....	761
install .....	761
installByAppId .....	762
login.....	763
logout.....	764
runTrigger .....	765
search .....	766
selectNumber .....	767
selectQuery .....	768
selectValue .....	770
setBinaryData.....	771
setDataField .....	772
update .....	773
updateArr.....	775
updateCustomer .....	776
updateRecord.....	777
update2.....	778

---

view .....	779
Rollbase SOAP Methods.....	779
DataObj Container Class.....	780
bulkCreate() .....	781
bulkCreateUpdate() .....	781
bulkUpdate() .....	782
clearDataObjectCache().....	783
create() .....	784
createArr() .....	785
createArr2() .....	786
createArrNoAudit() .....	787
createCustomer() .....	788
createRecord().....	789
delete() .....	790
deleteArr() .....	790
deleteArrNoAudit().....	791
deleteRecord().....	792
deleteRecords().....	792
detailedSearch().....	793
getBinaryData() .....	794
getCodebyId() .....	795
getCount() .....	796
getIdByCode() .....	796
getDataField() .....	797
getDataField2() .....	798
getDataObj() .....	799
getExchangeRate() .....	800
getPage() .....	801
getRelatedIDs().....	802
getRelationships() .....	803
getRuntimeStatus() .....	804
getUpdated() .....	804
getRecord().....	805
login().....	806
login2().....	807
logout() .....	807
metadataSearch() .....	808
SearchFilter() .....	809
selectNumber() .....	810
selectQuery() .....	811
selectValue() .....	812
setBinaryData() .....	813
setDataField() .....	814
setDataField2() .....	815
setExchangeRate() .....	816

---

---

setRelationship()	817
setRelatedIDs()	818
textSearch()	819
update()	819
updateArr()	820
updateArrNoAudit()	821
updateCustomer()	822
updateRecord	823
Rollbase CSS Styles	824
Table Styles	824
Table Cell Styles	826
Table Row Styles	829
Text Styles	830
Page Editor Styles	831
Link Styles	831
Sidebar Styles	832
Field Types	832
Text Field	832
Text Area Field	833
Checkbox Field	833
Decimal Field	833
Currency Field	834
Base Currency Field	834
Date Field	834
Date/Time Field	835
Time Field	835
Email Field	835
Phone Number Field	835
Password Field	835
Integer Field	836
Percent Field	836
Picklist Field	836
Picklist Multiselect	837
Radio Button Field	837
Group Checkbox Field	837
URL Field	838
Auto-Number Field	838
File Upload Field	838
Image Upload Field	838
Shared Image Field	839
Formula Field	839
Expression Field	840
Template Field	841
Document Template Field	841
Email Template Field	841

---

Related Field.....	842
Integration Link Field.....	842
Dependent Picklist Field.....	842
Version Number Field.....	843
Reference Field.....	843
Advanced Field Properties.....	843
System Field Types.....	844
Comments System Field.....	844
iCal System Field.....	844
vcard System Field.....	845
Organization Data System Fields.....	845
Time Zone System Field.....	845
User Role Field.....	845
LDF Filter Field.....	845
Parent Object Field.....	845
Tag Field.....	845
Portal Field Types.....	846
Captcha Image Field.....	846
Hidden Input Field.....	846
IP Address Field.....	846
<b>Chapter 15: Getting Help .....</b>	<b>847</b>

# Copyright

---

© 2014 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, EasyL, Fathom, Making Software Work Together, OpenEdge, Powered by Progress, Progress, Progress Control Tower, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, Sequelink, SpeedScript, Stylus Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, AppsAlive, AppServer, BusinessEdge, DataDirect Spy, DataDirect SupportLink, , Future Proof, High Performance Integration, Modulus, OpenAccess, Pacific, ProDataSet, Progress Arcade, Progress Pacific, Progress Profiles, Progress Results, Progress RFID, Progress Progress Software, ProVision, PSE Pro, SectorAlliance, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, Smart DataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.



---

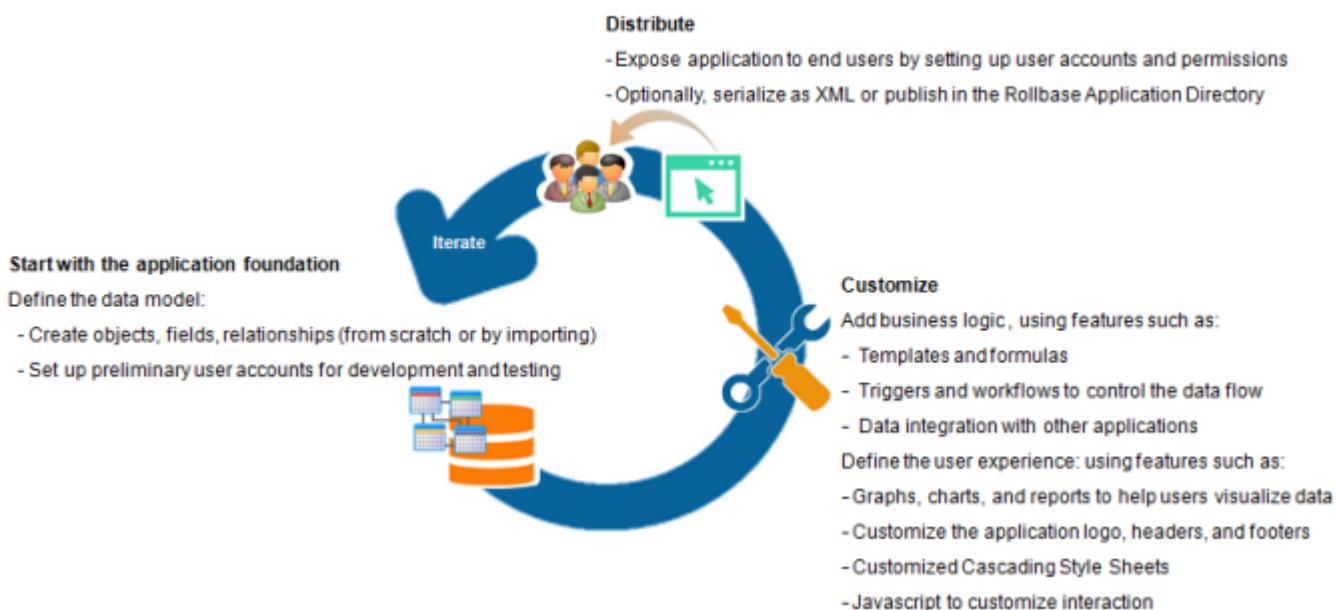
# Introduction to Progress Rollbase

---

Welcome to Progress® Rollbase,® a web-based platform for creating, customizing and distributing applications. Rollbase applications run in an integrated online environment and share a common security model, data model and user interface. Rollbase exemplifies Platform-as-a-Service (PaaS). There is no hardware or software to buy or install, both developers and end-users access Rollbase using a web browser. Rollbase allows you to focus on creating business value for users rather than on the expensive, complex and time-consuming tasks of managing software and infrastructure.

The Rollbase development environment supports citizen developers and business users who want to create custom applications. Its framework of wizards and dialogs provide drag-and-drop and point-and-click convenience. You can create sophisticated applications without writing a line of code, but you also have the option to use standards such as JavaScript and HTML to customize the logic and user interface. The following image illustrates the typical process for developing Rollbase applications:

## Rollbase Development Process Overview



The Rollbase **Application Directory** provides an online exchange where you can browse, try and install ready-made applications such as, a CRM system, a suite of integrated HR applications, a Bug Tracking System and others. Once a Rollbase application is installed, you can customize it to meet your specific business needs.

### Hosted and Private Cloud

Rollbase offers both hosted and private cloud environments. Private cloud users install Rollbase components on their own network. For information on Hosted and Private cloud, See [www.progress.com/products/rollbase](http://www.progress.com/products/rollbase).

### Affiliates, Resellers, and ISVs

Resellers and ISVs can use Progress Rollbase for developing and delivering custom SaaS Applications to their customers. Rollbase provides white label programs for both hosted and private cloud customers. For more information about working with Rollbase as a reseller or ISV, see [Setup and Administration for ISVs](#) on page 545

### Orientation Videos

The following videos provide a quick orientation to Rollbase:

- [Finding your way around in Rollbase](#)
- [The Quick Create Wizard](#)
- [An overview of application customization](#)
- [An overview of how to distribute Rollbase applications](#)

For details, see the following topics:

- [Supported Browsers and Secure Access](#)
- [Obtaining an Account and Logging In](#)

- [Basic Rollbase Concepts](#)
- [Navigating the Rollbase Environment](#)
- [Sample Applications](#)

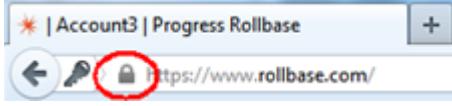
## Supported Browsers and Secure Access

Access Rollbase hosted cloud at <http://www.rollbase.com> using a computer running one of the following operating systems: Windows®, Linux®, or MacOS®. All you need is an up-to-date web browser with JavaScript and cookies enabled. Progress recommends use of the following browsers:

- Internet Explorer® 9.0 +
- Firefox® 21 +
- Chrome™ 27.0 +
- Safari® 6.0 +

Rollbase protects web browsing sessions with HTTPS and SSL. With HTTPS, Secure Socket Layer (SSL) technology protects your information using both server authentication and data encryption, ensuring that your data is safe, secure and available only to registered users of your account.

Depending on the browser, a lock icon displays in the address or the status bar to verify that you are accessing the Rollbase service via HTTPS. The screen shots below show where the lock icons appear in the Firefox and Internet Explorer browsers.

Firefox	
Internet Explorer	

## Obtaining an Account and Logging In

The first person to sign up for Rollbase from a particular organization becomes the administrator of the Rollbase tenant. (A tenant is a virtual space for creating and managing apps.) Administrators can create new user accounts, install applications from the **Application Directory**, customize applications for specific business needs and create new applications.

Administrators create user accounts by adding a **User** record. Each **User** record has a required **User Role** field. Any Rollbase user with the **Administrator** role has complete administrative privileges.

Private Cloud and ISV Partners have the option to create multiple tenants, which they do by adding **Customer** records. Therefore, the documentation often refers to these as Customer tenants. Typically, those with the ability to create tenants will want to use separate tenants for development and production.

## Logging In

After an administrator creates a **User** record, the new user will receive a confirming email with a temporary password and a login URL. Progress recommends changing passwords periodically. Administrators can require that passwords for their tenant meet certain criteria, such as length and inclusion of numbers.

# Basic Rollbase Concepts

Rollbase enables development and deployment of the following types of apps:

- Rollbase Web apps consist of a set of configurable components. Core application components include objects, tabs and portals. Each of these core components contains sub-components. The process of creating a web application in Rollbase involves defining core components and then building them out by adding and configuring sub-components. The Rollbase environment, whether hosted or Private Cloud, transparently handles the data storage for Rollbase Web applications.
- Rollbase Mobile apps can be associated with or be completely independent from a Rollbase Web application. When associated with a Web application, a Mobile app can access and update Web application data. The Rollbase Mobile App Builder enables rapid development and deployment of mobile apps. See [Rollbase Mobile](#) on page 272 for more information on mobile apps associated with Rollbase Web applications and <http://docs.mobile.rollbase.com/> for information on using the Rollbase Mobile App Builder.

## Rollbase Web App Capabilities

The Rollbase web app development environment allows you to:

- Create an application data model by defining objects, fields, and relationships.
- Customize the user interface by modifying components such as pages, views, and templates and by adding custom icons and logos.

---

**Note:** If you or an administrator customizes pages in your Rollbase tenant, the screen shots in the documentation might not match what you see on your screen.

---

- Define and customize workflow processes, statuses, and actions.
- Develop automated programmatic business logic such as with triggers.
- Generate and process template-based documents in MS Word, MS Excel, HTML, and plain text email formats.
- Enrich the user interface and workflow using client-side and server-side APIs.
- Use sophisticated full text search and filtering.
- Process data records using conversion mapping, comparison, parallel and sequential approvals, record queuing, multi-currency support, and other built-in functionality.
- Use built-in survey capabilities to create quizzes or questionnaires.

- Schedule triggers to automate data processing.
- Use the built-in calendar for task and event management.
- Define portals to create external facing applications that integrate with your Rollbase applications and seamlessly fit within existing websites and intranets.
- Secure your data using a number of authentication options and sophisticated access control.
- Enable integration with Google Apps: Gmail (outgoing and incoming), Spreadsheet, Calendar and Maps.
- Easily import your data and metadata from sources such as spreadsheets, Progress OpenEdge services, and data exposed by DataDirect Cloud or Microsoft Access.
- Easily migrate applications from Salesforce.com and the Force.com platform.
- Use SOAP or REST to access or modify application data.
- Publish, distribute, and install any application built using Rollbase.
- Expose Rollbase application views as mobile apps and/or use the Rollbase Mobile App Builder to create mobile apps with a customized user experience.
- ISV Partners can create and manage customer tenants to run their own SaaS business.

## Navigating the Rollbase Environment

The Rollbase interface appears differently to different types of users. Only users with administrative privileges see controls for development tasks such as creating and editing component definitions and customizing the user interface. Application end-users can only navigate through and view pages and components that have been exposed to them.

In the Rollbase environment, pages belong to one of two categories: application or setup. Each Rollbase component has a definition, which is saved in a setup page, and a realization — the application interface that is exposed to end-users.

For example, the following screen shows an application definition, including its properties and child components:

The screenshot shows the 'Application Setup > Applications > Room Reservation' screen. At the top, there is a tip message: 'Tip: This application can be published to the Application Directory for other customers to install. [Learn more](#)'. Below this is the 'Setup Application: Room Reservation' title and a toolbar with 'Edit', 'Publish', 'Delete', and 'More actions...'. The main content area is divided into sections: 'Application Details' and 'System Information'. In 'Application Details', fields include 'Application Name: Room Reservation', 'Deployed: ✓', 'Hidden', 'Field-Level Help: Custom Sidebar', 'Mobile-Web Enabled: Version 2', 'Description', 'Installed from: Import from Progress Rollbase XML', 'Installed At: 06/23/2014 05:50 PM', 'Tabs: Rooms, Furnishings, Devices, Reservations, Chart', 'Core Objects: Device, Furniture, Reservation, Room', and 'Portals'. In 'System Information', fields include 'Last Updated By: Adam Ministrator', 'Created By: Adam Ministrator', 'Last Updated At: 06/23/2014 05:50 PM', 'Created At: 06/23/2014 05:50 PM', 'ID: 107170179', and 'Original ID: 7549'. At the bottom is another toolbar with 'Edit', 'Publish', 'Delete', and 'More actions...'.

The following screen shows the resulting application interface. You can use the interface to test as you go. When the application is complete, you expose the application to end-users. See [Distribution Options](#) on page 80 for more information.

The screenshot shows the 'Rooms' tab of the application. The top navigation bar includes 'Rooms', 'Furnishings', 'Devices', 'Reservations', 'Chart', and a 'New' button. Below the navigation is a toolbar with 'Select', 'New Room', 'All Rooms', 'Filter', and 'More actions...'. The main content is a table with the following data:

Action	Room	Street Address 1	City	State/Province	Phone Number
<a href="#">Edit   Del</a>	Classroom 4	44 Corporate Way	Austin	Texas	(512)343-5688
<a href="#">Edit   Del</a>	Overflow Dining	45 Corporate Way	Austin	Texas	(512)323-7899
<a href="#">Edit   Del</a>	Atrium Social Corner	45 Corporate Way	Austin	Texas	(512)343-9877
<a href="#">Edit   Del</a>	Executive Conference	45 Corporate Way	Austin	Texas	(512)343-5600
<a href="#">Edit   Del</a>	Cafeteria	44 Corporate Way	Austin	Texas	(512)343-5879
<a href="#">Edit   Del</a>	Auditorium	45 Corporate Way	Austin	Texas	(512)343-4599
<a href="#">Edit   Del</a>	Fish Bowl Conference	45 Corporate Way	Austin	Texas	(512)343-6688

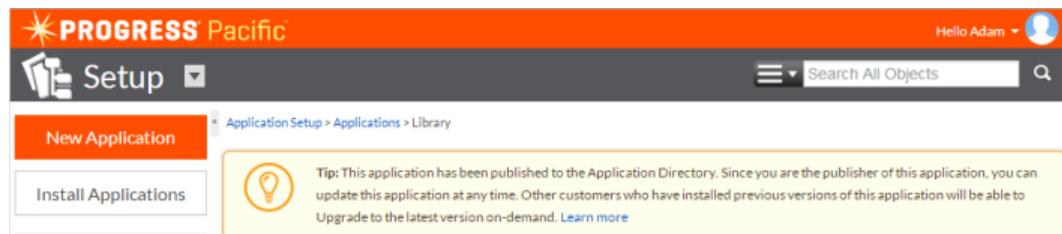
At the bottom, there are links to 'Rooms', 'Furnishings', 'Devices', 'Reservations', and 'Chart'.

When you create applications or add object definitions to existing applications, Rollbase creates a set of pages for each of those objects. By default, Rollbase creates a tabbed page containing a list view of all records for that object. The tabbed page contains menus and controls for finding, viewing, editing, and deleting object records. See [Application Tabs and Menus](#) on page 92 for more information about tabs and [Application Page Types](#) on page 87 for a list of object pages.

The heading banner colors provide a visual contrast between application pages and setup pages. The **Rollbase** application (not shown here) has the same color as **Setup** pages.

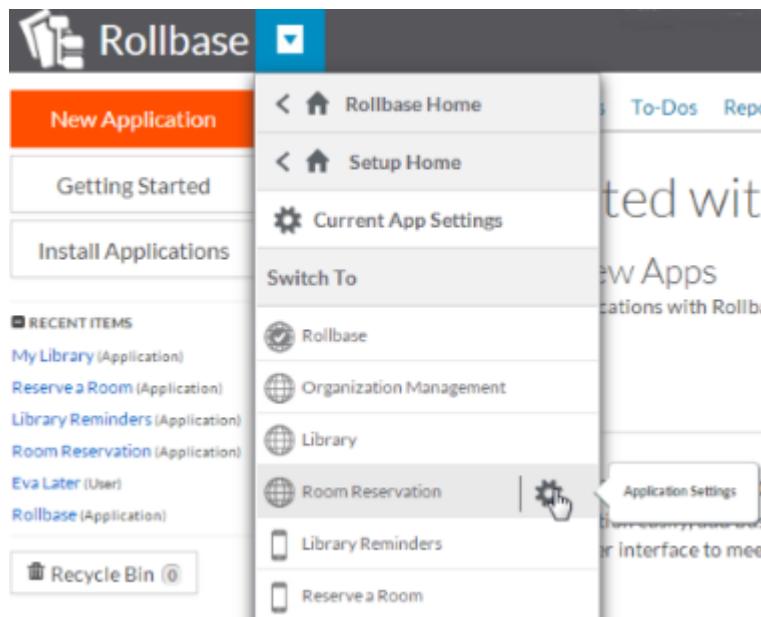


and application setup pages:



The banner contains a menu with shortcuts to applications and setup:

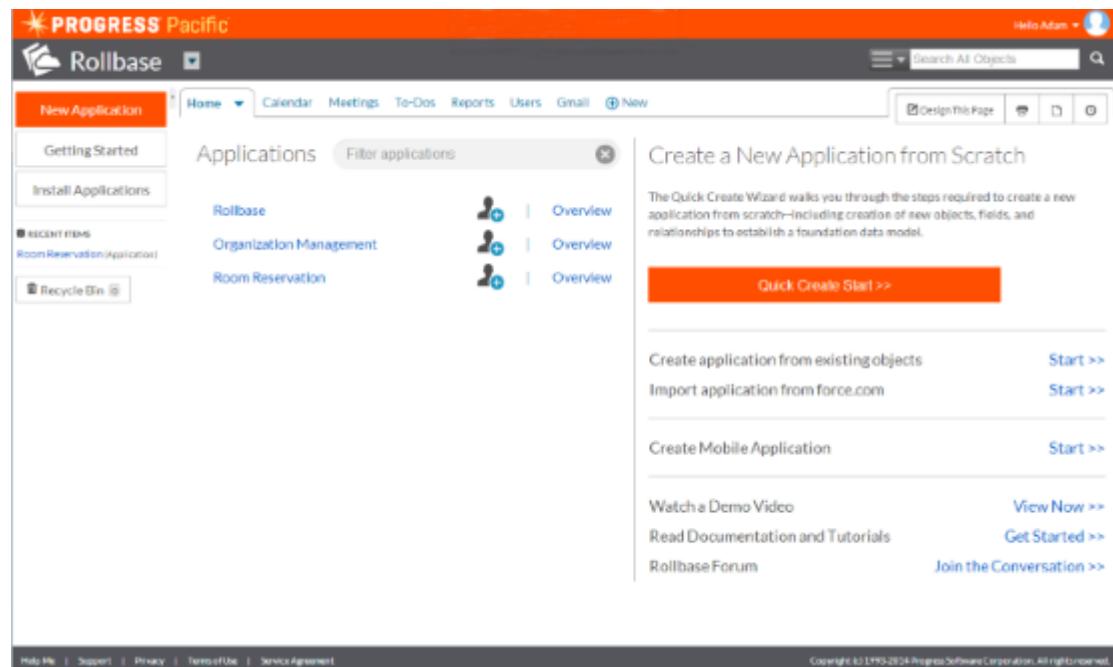
- Select **Rollbase Home**, **Setup Home**, or your application.
- Click **Current App Settings** to see application setup for the app you are viewing.
- To navigate to settings for a different application, select the application and click its **Application Settings** link.



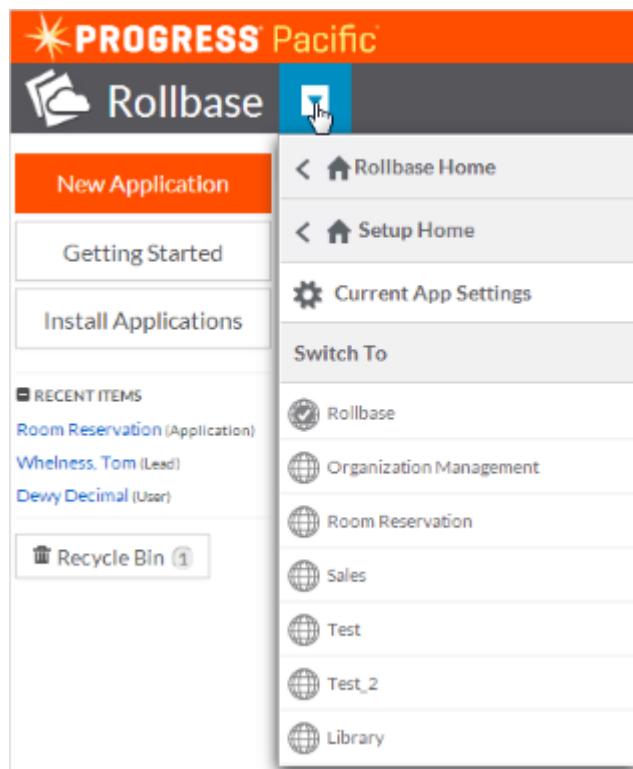
The topics in this section cover functionality that you will use frequently. Watch [Finding your way around in Rollbase](#) for a short tour of the Rollbase interface.

## The Main Screen

Users with a role of **Administrator**, such as application developers, see a screen similar to the one shown below when they log in. They can access both application and setup pages. In contrast, application end-users only see the applications and application components that are exposed to them.



The arrow next to the application name opens a menu for quick access to applications, and for administrators, to settings:

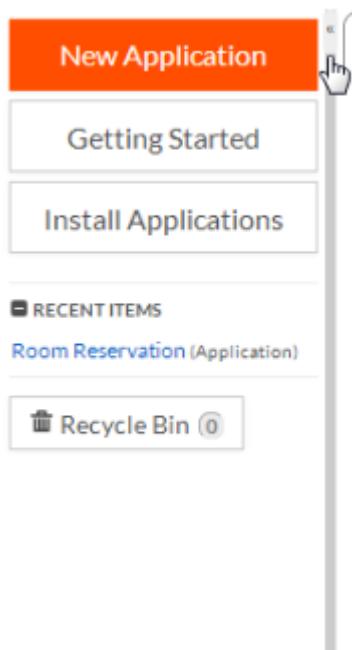


The following sections describe the main navigational controls:

- [Left Sidebar](#) on page 35
- [Top Right Navigation Controls](#) on page 36

## Left Sidebar

The << icon collapses or exposes the left sidebar:



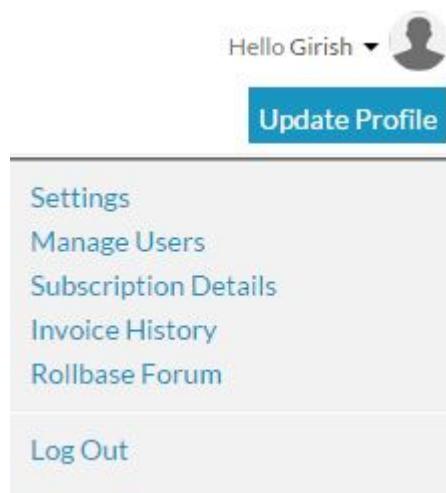
By default, the sidebar contains the following navigational controls:

- **New Application** — Launches the **Create Application** dialog.
- **Getting Started** — Only displays for new users. Launches a page with information and links to help you get started developing applications.
- **Install Applications** — Opens the **Application Directory** where you can find, test drive and install ready-made applications into your account.
- **Recent Items** — Displays a list of recently viewed data records, allowing quick access for editing. Record object types display in parentheses. Click on the link to open the record. administrative users also see recently viewed application components.
- **Recycle Bin** — Opens the **Recycle Bin**, from which you can review deletions, restore previously deleted data or purge it from the bin. Some data, such as uploaded files, cannot be restored. Rollbase purges deleted records permanently after 30 days.

## Top Right Navigation Controls

The top right screen area includes the following controls:

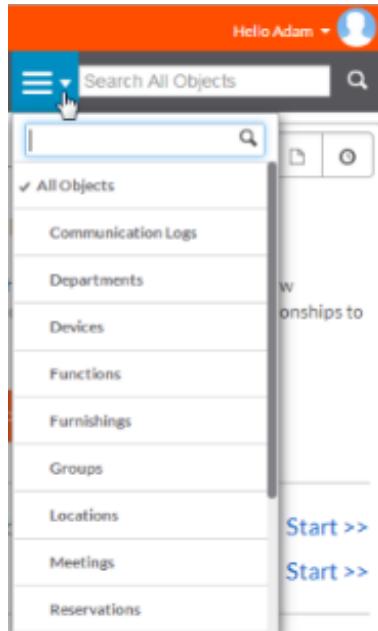
- A drop-down menu next to your profile avatar. The following image shows the menu that is visible to an administrative users:



The menu includes the following options:

- **Settings** — Opens the **Setup** page where you can manage the Personal Setup, Applications Setup, and Administration setup. Non-administrative users only have access to their personal settings.
- **Manage Users** — Opens the **Users** tab of your Rollbase application in which you can create and manage user records of your Rollbase account. This option is available only for administrative users.
- **Subscription Details** — Opens the **About Your Account** page which specifies all the subscription details. This option is available only for administrative users.
- **Invoice History** — Opens the **Invoice History** page that lists all the Progress Rollbase invoices generated for a tenant. You can filter the invoices using the **Data Range** field and hover over an invoice to download it as a PDF. This option is available only for the user identified as the billing administrator of Rollbase.

- **Forum** — Opens the Progress Communities site for Rollbase. In this page, you can access forums, technical, and product support related information.
- The **Search** and filter. The Rollbase search engine searches structured data, such as object records and unstructured data, such as file attachments. The filter menu allows you to narrow searches by object type or by metadata.



Topics in this section describe Rollbase interface components in more detail.

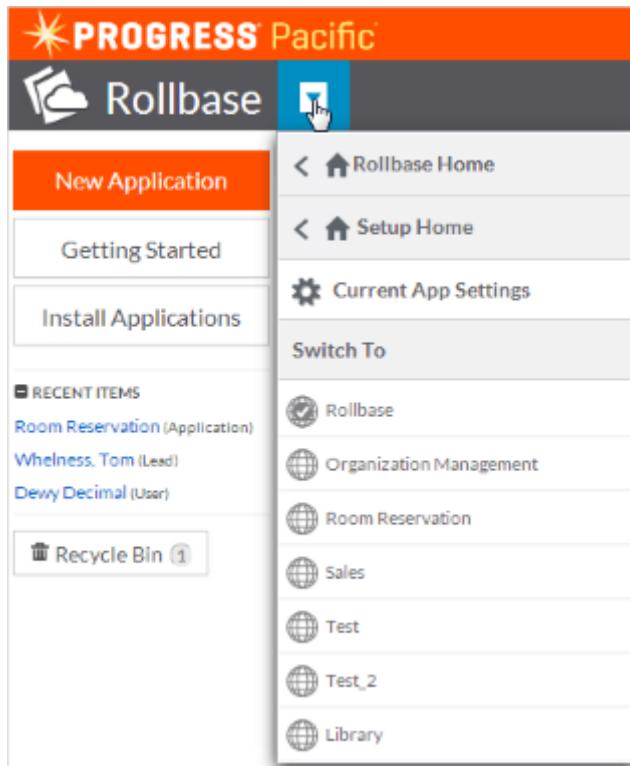
## Setup

Setup includes configuration for the following categories:

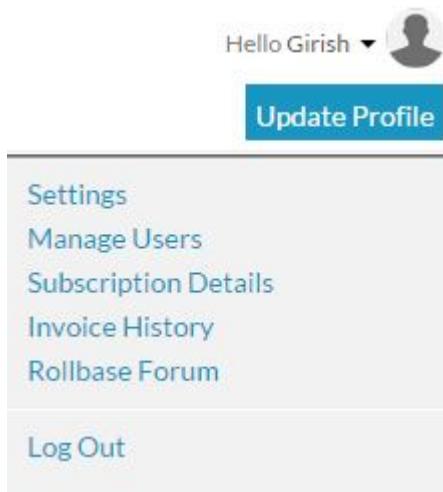
- **Personal Setup** — Preferences such as language, time zone, and Google app settings.
- **Applications Setup** — Definitions for applications and application components.
- **Administration Setup** — Global settings for the account and user role management.

Access **Setup** in either of the following ways:

- Select **Setup Home** from the list of applications:



- Select **Settings** from the drop-down menu next to your profile:



Users with an administrative role can configure personal, application, and account settings from the **Setup** page. Users without administrative privileges can only configure their personal settings.

**PERSONAL SETUP**  
Manage your account settings and password  
[My Settings](#) | [Change My Password](#)

**APPLICATIONS SETUP**  
Create, install, update, manage, customize and integrate applications  
[Applications](#) | [Objects](#) | [Tabs](#) | [Portals](#) | [Hosted Files](#) | [Application Directory](#)  
[SOAP API](#) | [REST API](#) | [Code Generator](#)

**ADMINISTRATION SETUP**  
Create and manage users, roles, authentication, backups, batch jobs and more  
[Users](#) | [Roles](#) | [Transfer Owners](#) | [User Access Log](#) | [Support Access](#) | [Settings](#)  
[Account Settings](#) | [Billing And Support Settings](#) | [Currency Codes](#) | [Exchange Rates](#)  
[Whitelist](#) | [Backup](#) | [Batch Jobs](#) | [Global Text Search](#) | [System Jobs](#) | [Portal Visitors](#)  
[Online](#) | [System Events Log](#) | [System Errors Log](#)

## The Rollbase Application

The Rollbase application gives administrators access to user accounts and the calendar. The following screen shot shows the Rollbase application **Home** page.

**PROGRESS Pacific** | [Hello Admin](#) | [Logout](#)

**Rollbase** | [New Application](#) | [Home](#) | [Calendar](#) | [Meetings](#) | [To-Dos](#) | [Reports](#) | [Users](#) | [Gmail](#) | [New](#)

**Applications** | [Filter applications](#)

<a href="#">Room Reservation Application</a>	<a href="#">New Application</a>
<a href="#">Getting Started</a>	<a href="#">Install Applications</a>
<a href="#">Recent Items</a>	<a href="#">Recycle Bin</a>

**Applications**

<a href="#">Room Reservation Application</a>	<a href="#">Overview</a>
<a href="#">Organization Management</a>	<a href="#">Overview</a>
<a href="#">Room Reservation</a>	<a href="#">Overview</a>

**Create a New Application from Scratch**

The Quick Create Wizard walks you through the steps required to create a new application from scratch—including creation of new objects, fields, and relationships to establish a foundation data model.

**Quick Create Start >>**

**Create application from existing objects** [Start >>](#)

**Import application from force.com** [Start >>](#)

**Create Mobile Application** [Start >>](#)

**Watch a Demo Video** [View Now >>](#)

**Read Documentation and Tutorials** [Get Started >>](#)

**Rollbase Forum** [Join the Conversation >>](#)

[Help Me](#) | [Support](#) | [Privacy](#) | [Terms of Use](#) | [Service Agreement](#)

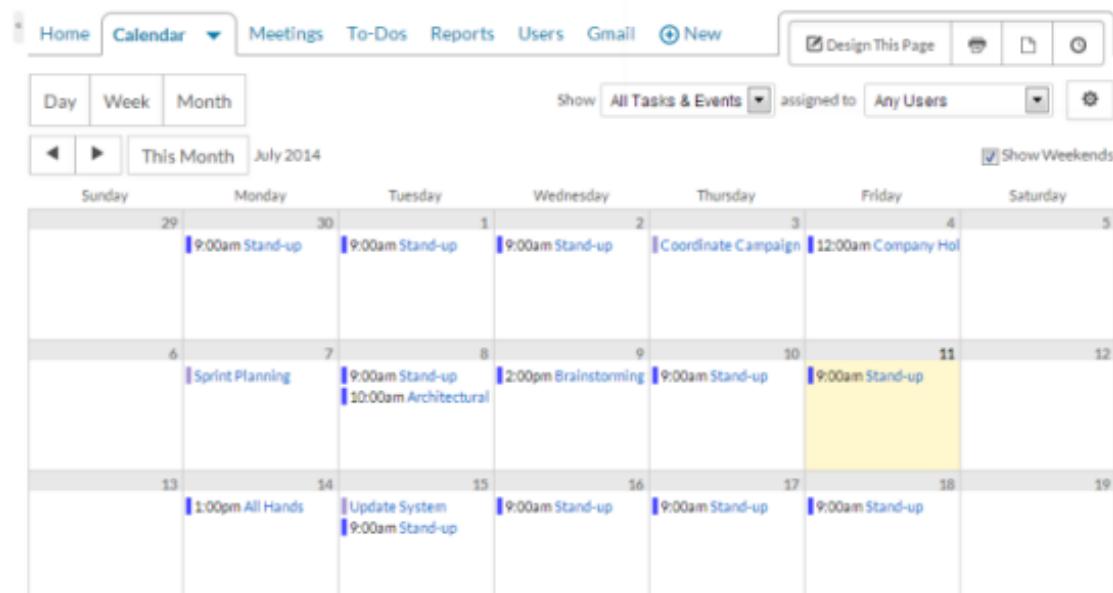
Copyright © 1990-2014 Progress Software Corporation. All rights reserved.

## Rollbase Calendar

The Rollbase application includes a calendar. Records whose object definition have an **Event** or a **Task** attribute display in the calendar. You can add the **Calendar** tab to any application to include its built-in functionality.

You can have the calendar display by day, week, or month. You can filter any calendar view by object type and by task records assigned to you, or to all users. Records marked as private that are not assigned to you will not appear in your calendar.

Hosted Rollbase users access the calendar through the Rollbase application or any application that includes the calendar object. Private cloud users of the classic Rollbase interface see calendar shortcut buttons in the sidebar. See [Private Cloud Administration Overview](#) for more information about switching between the Pacific and the classic interface.



You can use calendar functionality in any of the following ways:

- Create calendar entries in any of the following ways:
  - From the calendar itself: click a day, select the type of entry to add, and click **Create**.
  - From the Rollbase application, add **Meeting** or **To-Do** records.
  - From any application, add a record for any object that includes a task or event attribute.
- Add a **Calendar** component to any generic page using the **Page Editor**. For example, you might have object definitions for Training Classes, Conferences and Project Deadlines. Using a **Calendar** view, you can quickly see when you have any of these things planned and manage them accordingly.
- Synchronize Rollbase calendar events (but not tasks) with:
  - Google calendar, see [Google Calendar](#) on page 369
  - For other calendars, such as Microsoft Outlook, you can export records individually in the standard iCal format. Click the record you want to export and from the More Actions menu, select iCal. Rollbase exports the record locally and the Rollbase footer shows the exported

file's name. Click the downloaded file to open it in Outlook and Outlook will prompt you to take the appropriate action.

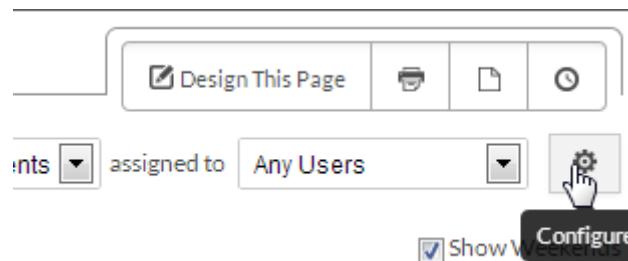
## Configuring the Calendar

Administrative users can set the following calendar properties:

- Default View (Day, Week, or Month)
- Show weekends on Week and Month view?
- Range of hours (in 0-24 range) to display on Day and Week view

To configure the calendar:

1. In the calendar toolbar, click **Configure**.



2. Configure the desired properties:

A screenshot of a configuration dialog box titled 'Calendar: Configure Calendar Component'. The dialog has tabs for 'Personal Setup', 'Applications Setup', and 'Administration Setup', with 'Administration Setup' selected. The main area is titled 'Calendar Properties'. It shows a 'Default View' dropdown set to 'Week', a checked checkbox for 'Show weekends on Weekly and Monthly views', and a section for selecting a 'range of hours to display in Daily and Weekly views of Calendar Component'. The 'From Hour' is set to 0 (0 - 10) and the 'To Hour' is set to 24 (17 - 24). At the bottom are 'Save' and 'Cancel' buttons.

3. Click **Save**.

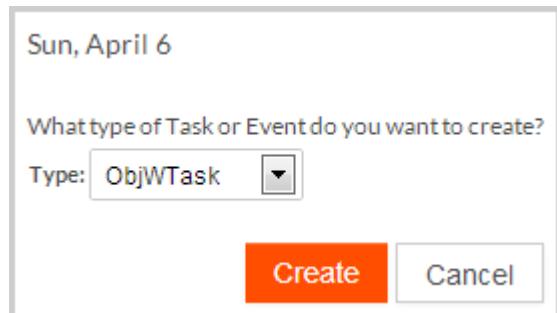
## Day, Week, and Month Views

The month view shows all days in the month and lists all tasks and events in chronological order. Clicking on a record takes you to the view page for that record.

Depending on the type of entry, clicking a link in day or week view displays more information or takes you to the record itself:



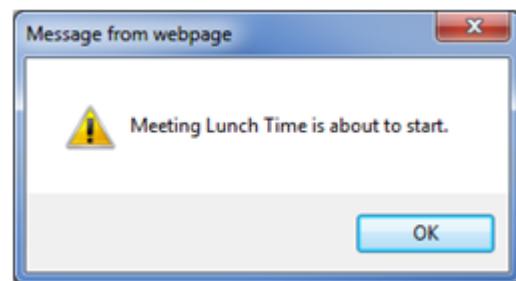
Clicking on a day or hour launches a dialog to create a new record:



## Calendar Notifications

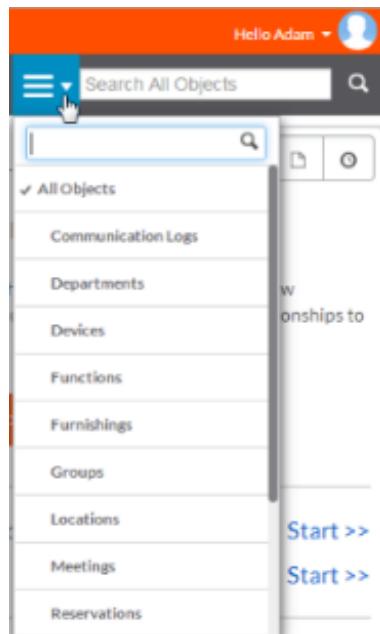
Event objects can include trigger-based notification, which sends an email reminder. For information on triggers, see [Trigger Overview](#) on page 179.

If logged in to Rollbase, users will receive pop-up notifications when an event assigned to them is about to start. Timing for that notification is determined by the **Pop-up Reminder** field on a record. Reminders are displayed only once.



# Search

The search field at the top right of the screen searches all pages in the tenant account for object records and unstructured data, such as file attachments. You can search all objects, by object type, or search all metadata.



Object fields have a property that specifies whether or not to **Index this Field as part of the text search engine**. If none of an object's fields has this property checked, a search will not include the object. See [Advanced Field Properties](#) on page 843 for more information.

## Search Syntax

The simplest way to search for data is to type a single term into the search box. Search queries in Rollbase are broken up into terms and operators. There are two types of terms: Single Terms and Phrases.

A Single Term is a single word, such as "Sales" or "Marketing." A Phrase is a group of words surrounded by double quotes, such as "Human Resources".

Rollbase supports advanced search syntax, such as wildcard character searches and the ability to combine multiple terms together with Boolean operators to form more complex queries.

## Wildcard Character Searches

Rollbase supports single and multiple character wildcard character searches:

- To perform a single wildcard character search, use the "?" symbol.
- To perform a multiple wildcard character search, use the "\*" symbol.

The single wildcard character search looks for terms that match those with the single character replaced. For example, to search for text or test you can use the search:

te?t

Multiple wildcard character search looks for 0 or more characters. For example, to search for test, tests or tester, you can use the search:

test\*

You can also use the wildcard character search in the middle of a term:

te\*t

---

**Note:** You cannot use a "\*" or "?" symbol as the first character of a search.

---

## Boolean Operators

Boolean operators allow you to combine search terms through logical operators. Rollbase supports AND, +, OR, NOT and -. The word operators must be entered in capital letters as shown.

Operator	Description	Example
OR	The OR operator is the default conjunction operator. If there is no Boolean operator between two terms, the search uses OR. The OR operator links two terms and finds a matching record if either of the terms exists in record fields or file attachments. This is equivalent to a union using sets. You can use the    symbol instead of the word OR.	For example, to search for records that contain either "Human Resources" or just "Human" use the query: "Human Resources" Human ...or: "Human Resources" OR Human"
AND	The AND operator matches records where both terms exist somewhere in one or more record fields or file attachments. This is equivalent to an intersection using sets. You can use the && symbol instead of the word AND.	To search for documents that contain "Human Resources" and "Sales" use the query: "Human Resources" AND "Sales"
+	The + required operator requires that the term after the + symbol exist somewhere in a single record field or in one of its file attachments.	For example, to search for records that must contain "Human" and may contain "Resources" use the query: + Human Resources

Operator	Description	Example
NOT	The NOT operator excludes records with fields or file attachments that contain the term after NOT. This is equivalent to a difference using sets. You can use the ! symbol instead of the word NOT. Note: The NOT operator cannot be used with just one term. For example, the following search will return no results: NOT "Sales"	For example, to search for records that contain "Human Resources" but not "Sales" use the query: "Human Resources" NOT "Sales"
-	The - prohibit operator excludes records with fields or file attachments that contain the term after the - symbol.	To search for records that contain "Human Resources" but not "Sales", use the query: "Human Resources" - "Sales"

## Escaping Special Characters

In a search query, you can escape the following characters:

+ - && || ! ( ) { } [ ] ^ " ~ \* ? : \

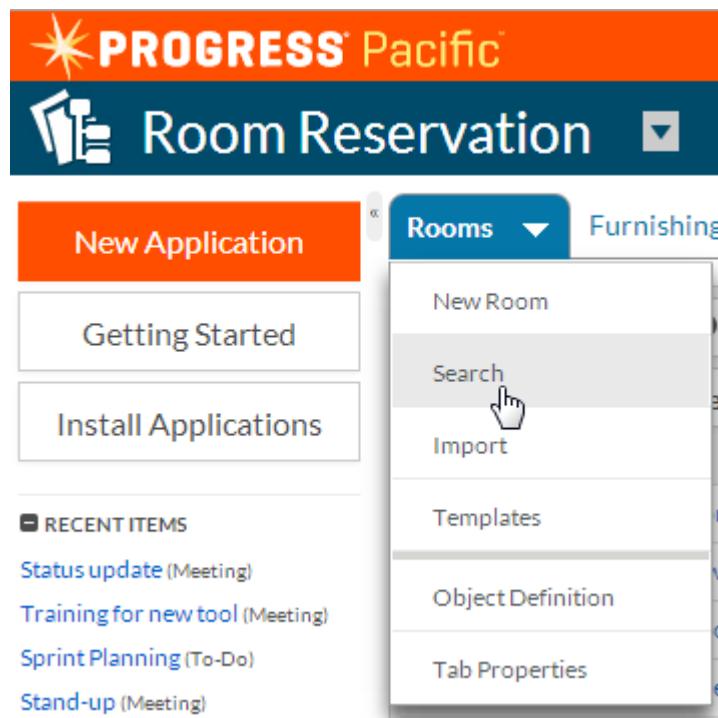
To escape these characters, use the backslash (\) before the character. For example, to search for (1+1):2, use the query:

\(1\+1\)\:2

## Object Search

To find records of a specific object type:

- Select the object type from the filter next to the [Search](#) box, or
- From the object tab:
  - Select search:



- Enter search criteria in the search fields:

Room: Search

Search for Keywords

Keywords

Search by Distance

Within Any miles from US ZIP code

Search Specific Fields | [Learn more](#)

Date Field Dates Interval

1 -None-- -None-- AND

2 -None-- -None-- AND

3 -None-- -None-- AND

4 -None-- -None-- AND

5 -None-- -None-- AND

Filter Conditions: All (AND)

Search

The **Search by Distance** appears only on search screens for objects that have a **Location** attribute. Search results display in the object's **Search Results** page. You can change the default view that is used to display search results by editing this page, selecting the **Search Results List** component, and selecting the desired view in the **Default List View** field in the **Properties** box.

## Metadata Search

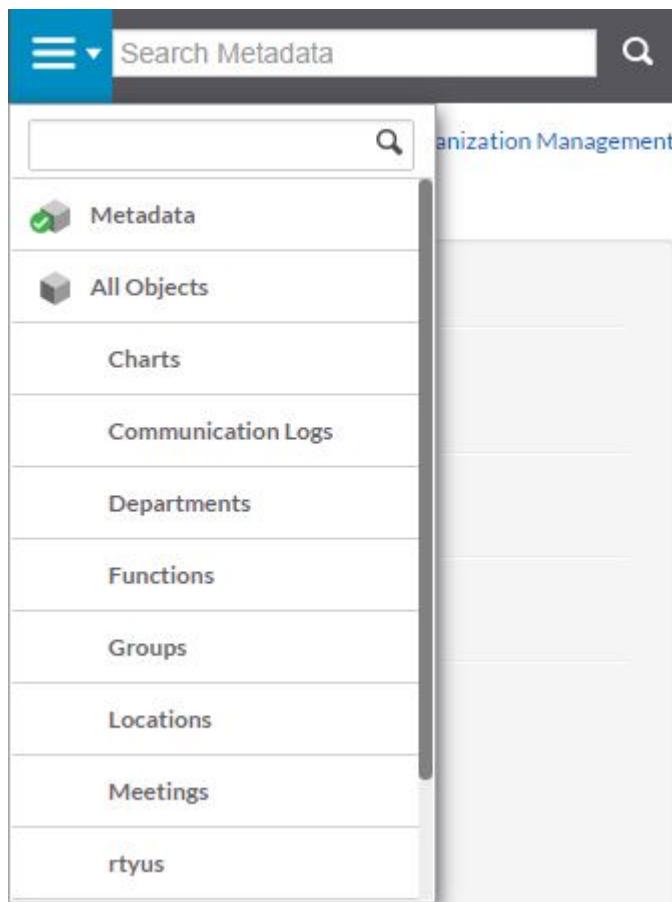
Metadata stores the definition of Rollbase applications and their components, including properties and attributes. [Metadata XML Reference](#) on page 687 lists all metadata elements. You can use Rollbase **Search** to find strings in the metadata and then sort the results by type. For example, since metadata includes the integration name, you could search for a field integration name and sort the results to locate the formulas that use that field.

The results of a metadata search include:

- The entity type, such as application, object, or trigger.
- A link to the page to view the entity
- The related object, if applicable
- Where the text was found
- When the entity was last updated
- Who last updated the entity

To perform a metadata search:

1. From the filter left of the **Search** box, select **Metadata**:



2. Enter a search string in the **Search** box.

---

**Note:** The metadata search is not case sensitive and you can use Wildcard characters in your search string as described in [Wildcard Character Searches](#) on page 43.

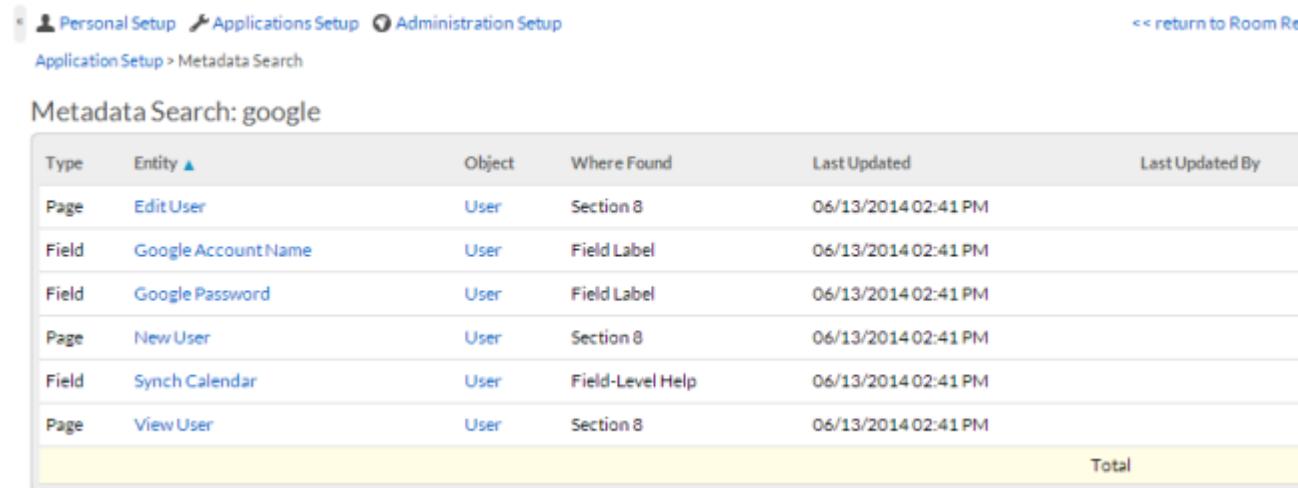
---

3. Press **Enter** or click the **Search** icon.

4. You can do the following:

- Click a heading in the result table to sort the rows.
- Click a link in the **Entity** column to view the entity.

The following screen shows sample results of a search for the string `google`:



The screenshot shows a search results page for 'Metadata Search: google'. The top navigation bar includes links for Personal Setup, Applications Setup, Administration Setup, and a 'return to Room' link. The search results table has columns for Type, Entity, Object, Where Found, Last Updated, and Last Updated By. The results are as follows:

Type	Entity	Object	Where Found	Last Updated	Last Updated By
Page	Edit User	User	Section 8	06/13/2014 02:41 PM	
Field	Google Account Name	User	Field Label	06/13/2014 02:41 PM	
Field	Google Password	User	Field Label	06/13/2014 02:41 PM	
Page	New User	User	Section 8	06/13/2014 02:41 PM	
Field	Synch Calendar	User	Field-Level Help	06/13/2014 02:41 PM	
Page	View User	User	Section 8	06/13/2014 02:41 PM	

## Language-Specific Search and Indexing

Text indexing uses the tenant-level base language. Foreign languages selected for individual users are not used in text indexing and full-text search in foreign languages is not guaranteed.

If you choose to change company-level base language from the **Account Settings** page, please ask support to re-create full-text index for your company.

The minimum length of a query for text search is one character for Asian (multibyte) languages and three characters otherwise.

# Printing and PDF Generation

## Printing Screens

All editable Rollbase pages have a **Print** button that renders the current page without the sidebar or header in a new window.



Links and buttons in the window are intentionally disabled. Click the printer icon to open the printer menu.

---

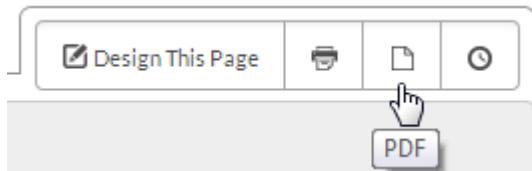
A screenshot of a 'Rooms' list view. The title bar says 'Rooms New Room +'. On the right, there is a navigation bar with the text 'All Rooms 1-7 of 7' and arrows for navigating through the list. Below the title bar is a table with columns: Room, Street Address 1, City, State/Province, and Phone Number. The table contains the following data:

Room	Street Address 1	City	State/Province	Phone Number
Classroom 4	44 Corporate Way	Austin	Texas	(512)343-5688
Overflow Dining	45 Corporate Way	Austin	Texas	(512)323-7899
Atrium Social Corner	45 Corporate Way	Austin	Texas	(512)343-9877
Executive Conference	45 Corporate Way	Austin	Texas	(512)343-5600
Cafeteria	44 Corporate Way	Austin	Texas	(512)343-5879
Auditorium	45 Corporate Way	Austin	Texas	(512)343-4599
Fish Bowl Conference	45 Corporate Way	Austin	Texas	(512)343-6688

Copyright(c) 1993-2014 Progress Software Corporation. All rights reserved.

## PDF Rendering of Record Lists

View and list pages provide a button for viewing the records they contain in PDF format.



The PDF rendering displays in a new window as shown, roll your cursor in the footer to access a toolbar for repositioning, saving, zooming, and printing.

Rooms

All Rooms 1-7 of 7

Room	Street Address 1	City	State/Province	Phone Number
Classroom 4	44 Corporate Way	Austin	Texas	(512)343-5688
Overflow Dining	45 Corporate Way	Austin	Texas	(512)323-7899
Atrium Social Corner	45 Corporate Way	Austin	Texas	(512)343-9877
Executive Conference	45 Corporate Way	Austin	Texas	(512)343-5600
Cafeteria	44 Corporate Way	Austin	Texas	(512)343-5879
Auditorium	45 Corporate Way	Austin	Texas	(512)343-4599
Fish Bowl Conference	45 Corporate Way	Austin	Texas	(512)343-6688



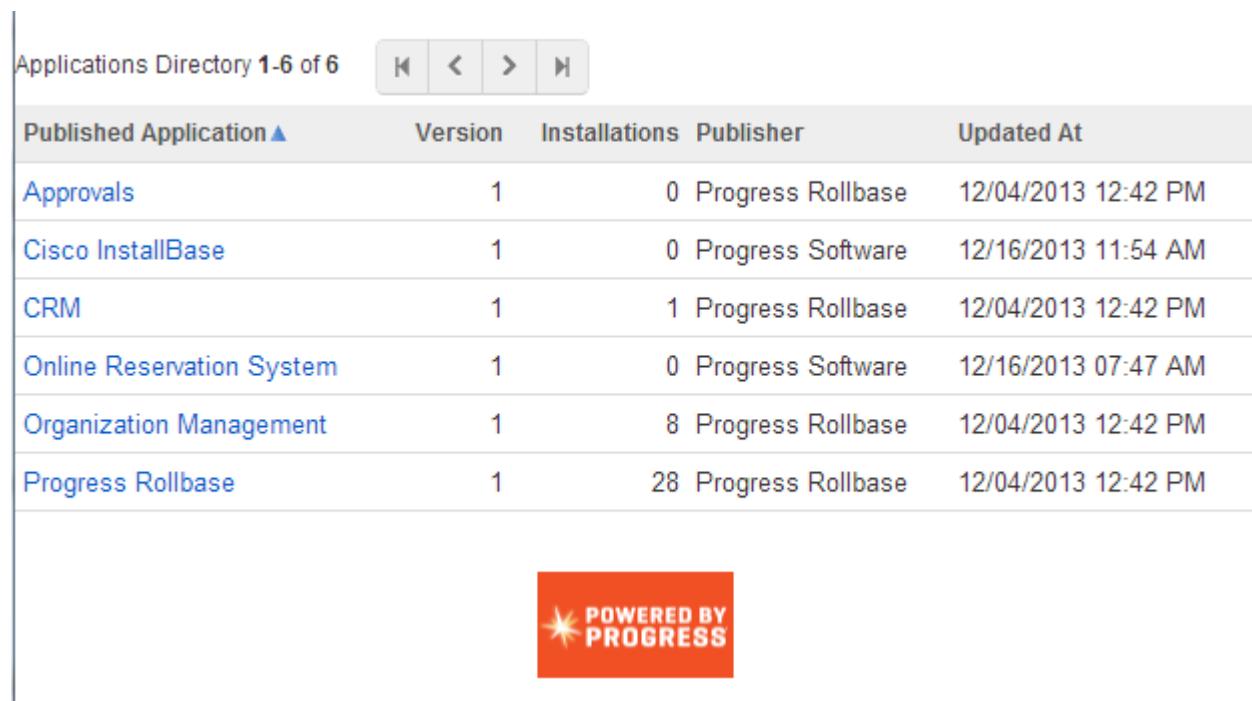
# Sample Applications

The best way to become familiar with Rollbase is to test drive or install one or more ready-made starter applications from the **Application Directory**. Many applications provide a **Test Drive** button, allowing you to try out that application in a read-only demo account. Unlocked applications can be customized to meet your needs once installed.

The hosted **Application Directory** contains only applications approved by Progress Rollbase. They were built by experienced Rollbase users. A Private Cloud **Application Directory** is similar but the administrator of the Master Server decides which applications will be available. Administrators can install any application from the **Application Directory** into their users' accounts with a single click. This enables them to immediately deploy critical business applications to the right people in the organization.

Users can browse the **Application Directory** without being signed in. If they do not yet have a Rollbase account, they can sign up for a free trial from an application page and that application will be installed into their tenant automatically once it is created.

To view the **Application Directory** when logged into Rollbase, click **Install Applications**. You can also search for applications using keywords. To install an application, click the application name link, and click **Install Now**. You can start using it by adding data records, customizing the application to fit your specific business needs, and assigning permissions for users.



The screenshot shows a table titled "Applications Directory 1-6 of 6" with a navigation bar at the top featuring icons for back, forward, and search. The table has columns for "Published Application", "Version", "Installations", "Publisher", and "Updated At". The data is as follows:

Published Application	Version	Installations	Publisher	Updated At
Approvals	1	0	Progress Rollbase	12/04/2013 12:42 PM
Cisco InstallBase	1	0	Progress Software	12/16/2013 11:54 AM
CRM	1	1	Progress Rollbase	12/04/2013 12:42 PM
Online Reservation System	1	0	Progress Software	12/16/2013 07:47 AM
Organization Management	1	8	Progress Rollbase	12/04/2013 12:42 PM
Progress Rollbase	1	28	Progress Rollbase	12/04/2013 12:42 PM



## Updating Applications

The authors of applications published in the **Applications Directory** can publish updates. If updates exist for an application you have installed, the application definition displays an **Install Updates** button that allows you to upgrade to the latest published version of that application.

Some applications are marked as **Managed**, meaning that the publisher retains full control of the application's structure; you cannot customize any of its components. Only the publisher can customize managed applications. See [Installing Application Updates](#) on page 556 for more information.

---

## What is New in Version 3.X

---

The topics in this section describe new features and changed behavior for Rollbase 3.X. For details, see the following topics:

- [3.1 Rollbase New Features and Changes](#)
- [3.1 Rollbase Mobile New Features and Changes](#)
- [3.1 Mobile App Builder New Features and Enhancements](#)
- [3.0 New Features and Enhancements](#)
- [3.0 Changed Behavior](#)
- [3.0 Rollbase Mobile Changes and Improvements](#)
- [3.0 Private Cloud Features and Changes](#)
- [Upgrading Rollbase Private Cloud](#)

### 3.1 Rollbase New Features and Changes

#### **Rollbase 3.1 New Features**

- **Support for object attributes in OpenEdge Service objects:** OpenEdge JSDO services, generated in Progress OpenEdge, can now enable Rollbase object attributes (such as, Workflow or Multi-Currency) in an OpenEdge Service object.

For information about enabling object attributes for an OpenEdge Service object, see [Enabling object attributes for an OpenEdge Service Object](#) on page 340.

- **Option to download Charts:** A **Download** option is introduced in Rollbase charts. You can use this option to save an XML-based (.SVG) image of a Rollbase chart in your computer.

For more information about the **Download** option, see [Using Charts](#) on page 212.

- **New system roles in Rollbase:** The following two new system roles have been added to Rollbase:

- **Portal Guest** role: Assigned to non-authenticated users of external-facing Portals. Unlike Portal Users, Portal Guests cannot log into a portal. Portal Guests can only access a public portal page.

- **AJAX API** role: Used to perform AJAX API calls when the current user is not authorized to perform an operation.

For more information about system roles, see [User Roles and Permissions](#) on page 393.

- **A page to assign billing and support contacts:** The administrator of the Rollbase tenant can make a user a billing administrator or a support contact.

For more information about the **Download** option, see [Billing and Support Settings](#) on page 431.

- **Newly added SOAP APIs:** The following are the newly added SOAP APIs:

- createArr2
- deleteRecords
- getDataField2
- metadataSearch
- setDataField2
- setRelatedIDs

For more information about SOAP APIs, see [Rollbase SOAP Methods](#) on page 779 and [SOAP Metadata Methods](#) on page 706.

- **Newly added REST APIs:** The following are the newly added REST APIs:

- getApplicationIds
- metadataSearch
- runTrigger

For more information about REST APIs, see [Rollbase REST Methods](#) on page 732 and [REST Metadata Methods](#) on page 720.

- **Newly added Client-side APIs:** The following are the newly added Client-side APIs:

- getRelatedIds2()
- getRelatedFields2()

For more information about Client-side APIs, see [Client-side AJAX API](#) on page 624.

- **Newly added Server-side APIs:** The following are the newly added Server-side APIs:
  - openIMAP()
  - openPOP3()
  - getMainMessageCount()
  - getMailMessage()
  - getMailMessages()
  - closeMailSession()
  - getRelatedIDs2()
  - getRelatedFields2()

For more information about the Server-side APIs, see [Server-side API](#) on page 559.

### Rollbase 3.1 product enhancements and changes

- **Redesigned JavaScript Editor for ease of use:** Earlier, the JavaScript Editor in Rollbase was a simple text editor. Rollbase 3.1 onward, the redesigned editor enhances the readability of your code by using different color schemes, providing resizing options and an auto-complete helper while typing built-in Rollbase templates in your Java Script code.
- **Redesigned HTML Editor for ease of use:** The redesigned HTML Editor provides you with three toolbars of editing option. This makes for better usability. Note that, using `shared.properties`, you can choose to disable the HTML editor menu options. The related properties are: `HideHtmlEditorMenu`, `HtmlEditorButtonRow1`, and `HtmlEditorButtonRow2`.
- **Customer tenants to permit master tenants to access their user accounts:** Earlier, master tenants could log into a customer tenant without any permissions from the customer tenant. Rollbase 3.1 onward, the administrator of a customer tenant must manually permit master tenant users to access its environment for a specified duration.

For more information about permitting administrative users to access a customer tenant accounts, see [Enabling an administrative user to log into a customer tenant](#) on page 406.

- **Quick navigation to the User management page:** The drop-down menu, next to the profile avatar of an Administrative user, provides an option to directly navigate to and manage users in the Rollbase instance.

For more information about the manage users option, see [The Main Screen](#) on page 34.

- **Upgrading FusionCharts:** Rollbase is upgraded to support FusionCharts 3.4.0 and above. For more information about installing FusionCharts, see [FusionCharts](#) on page 448.

- **The “Query API” system role is renamed to “Server API” and The “Portal Visitor” system role is renamed to “Portal User”.**

For more information about system roles, see [User Roles and Permissions](#) on page 393.

- **Document templates and Reports upgrade:** You can now create Rollbase templates and reports using a .DOCX document.

- **Shared.properties documentation update:** Documentation is now updated with all the properties available to you in `Shared.properties` file.

For more information about system roles, see [shared.properties](#) on page 505.

## 3.1 Rollbase Mobile New Features and Changes

The information described in this topic is specific to the Rollbase environment, see [3.1 Mobile App Builder New Features and Enhancements](#) on page 57 for descriptions of features specific to the Mobile App Builder environment.

### Rollbase Mobile 3.1 New Features

- **Mobile app authentication mechanism:** You can now specify which users can access your mobile app. By default, all users in the `User` object are authenticated. You can either select the `User` object or any other object that has a Portal user attribute. Additionally, you can allow any public user to access the mobile app anonymously.

For more information about setting Mobile App authentication, see [Editing Rollbase Mobile App projects](#) on page 306.

- **Generate metadata option for troubleshooting purposes:** As part of the **More actions** drop-down menu of your Mobile App Edit page, a **Generate Metadata** option is included to show a preview of metadata (JSON structure) in the application. You can view or download (a .zip file) the metadata information and use it to troubleshoot any discrepancies before exporting it to Mobile App Builder.

For more information about the **Generate Metadata** option, see [Editing Rollbase Mobile App projects](#) on page 306.

### Rollbase Mobile 3.1 product enhancements and changes

- **When creating Rollbase Mobile App based on exiting Web App, you export the core objects from the Web App to the Mobile App:** Earlier, when creating a Rollbase Mobile App, you specified the views that you want to export in the Mobile App. Rollbase 3.1 onward, you specify Core objects that you want to export in the Mobile app. On exporting a core object, all the views in those objects are automatically exported to your application.

**Note:** Only the selected core objects are exported to the Mobile App. All unselected objects, even if they are related to the selected core objects, are not exported to the Mobile App (unlike in 3.0.5 or earlier release of Rollbase Mobile).

For more information about the Generate metadata option, see [Creating a Rollbase Mobile App project](#) on page 302.

- **Dependent objects and synchronizing its metadata:** At the time of Mobile App creation, you only specify the core objects to be included in the Mobile App. If you must include any dependent objects to your Mobile App, you must select them in the Mobile App edit page and select the **Sync metadata for Dependent Objects** check box to include them in your Mobile App.

For more information about the dependent objects and synchronizing its metadata, see [Creating a Rollbase Mobile App project](#) on page 302.

- **The Download option is renamed to Export Web Resources:** Rollbase 3.1 onward, the **Download** option is renamed to **Export Web Resources** and is available under the **More Actions** menu of your Mobile App Edit page.

For more information about the **Export Web Resources** option, see [Editing Rollbase Mobile App projects](#) on page 306.

- **A Pacific account user can be a part of multiple Rollbase instances:** To enable Mobile development in Rollbase Private Cloud, you must link your account with a Rollbase Pacific account. Rollbase 3.1 onward, you can use the same Pacific account in multiple Rollbase instances.

For more information about enabling Mobile App Development in Private Cloud, see [Enabling Mobile App Development in Rollbase Private Cloud](#) on page 300.

## 3.1 Mobile App Builder New Features and Enhancements

In addition to the update of JQM and Cordova PhoneGap libraries and general performance improvements, the following features are new or enhanced in version 3.1. To upgrade projects created in earlier versions of the Mobile App Builder, see <http://documentation.progress.com/output/mobile-app-builder/index.html#context/mab/mab-upgrade>

- [Minified Export Release Binaries and Debug](#) on page 57
- [iOS Support Updated](#) on page 57
- [Secure REST functionality for OpenEdge mobile apps](#) on page 58
- [Table Component](#) on page 58
- [Team Development Enhancement for OpenEdge Developers](#) on page 58
- [Usability Enhancements](#) on page 58

### Minified Export Release Binaries and Debug

You can now export release and debug versions of browser-based and hybrid apps. Release binaries contain minified versions of JavaScript files. Minified JavaScript files do not contain unnecessary spaces and newlines, which makes them smaller and allows them transfer to and load from a web browser more quickly and for hybrid apps, run faster on devices. Debug binaries contain spaces and newlines, making them easier to debug.

### iOS Support Updated

This version of the Mobile App Builder supports export of hybrid phone apps for iOS 8.

### Enhanced Functionality for OpenEdge Express Projects

The Mobile App Builder now dynamically creates a UI for JSDO fields when you create an Express project.

## Secure REST functionality for OpenEdge mobile apps

OpenEdge mobile apps that use REST services instead of JSDO services can now keep secret keys, credentials, or other data safe from being opened by app users. To use this feature, store the secret data in a database under key names. In the Mobile Applications Monitoring Dashboard, select the **Secure REST** tab and create a proxy channel for replacing the key names with the secret data. In the mobile app, use key names and define the proxy channel. The mobile app calls key names, and in the proxy they are replaced with data, so the service receives real data.

The screenshot shows the 'Mobile Applications Monitoring Dashboard' with the 'Secure REST' tab selected. The 'Proxy name' is set to 'TestProxy'. The 'Proxy ID' is 'c630f882-6cc1-4292-bb8e-8fe53ca77e33'. The configuration section has two options: 'Use default proxy' (radio button) and 'Use proxy + store API keys in database' (radio button, selected). For the database proxy, the 'Database' is set to 'TestDB', and 'Collection', 'Key column', and 'Value column' are all set to 'Select value'. There is a 'Open database' button next to the database dropdown.

## Table Component

A new **Table** component in the Mobile App Builder makes it easy to display tabular data. You can define table properties to fit narrow device screens so that users will see the most important data. End users can also customize the table on their screens.

## Team Development Enhancement for OpenEdge Developers

If you have a paid account, you can share the Mobile App Project with others. In this release, one person can be a member of multiple teams. See [http://documentation.progress.com/output/mobile-app-builder/index.html#context/mab/mab\\_share](http://documentation.progress.com/output/mobile-app-builder/index.html#context/mab/mab_share) for more information. See <http://documentation.progress.com/output/pacific-console> for information on viewing and updating your account. Team functionality is not available for Rollbase developers in this release.

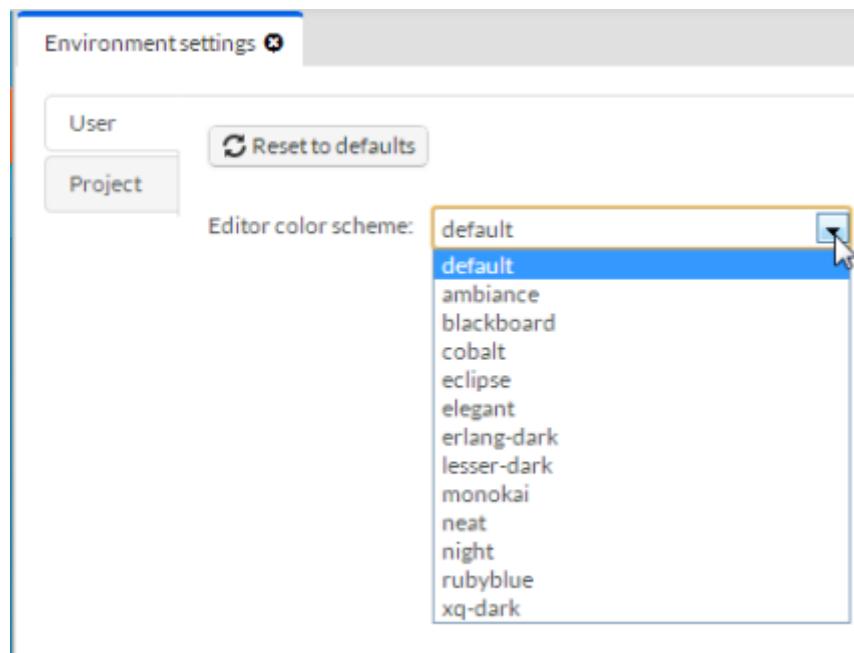
## Usability Enhancements

The Mobile App Builder interface has the following enhancements:

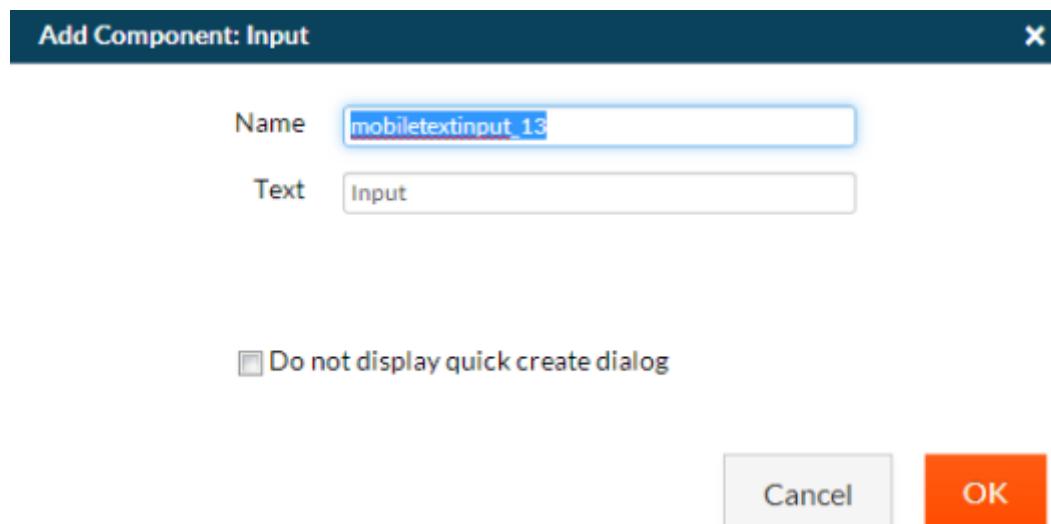
- To make it easier to recognize when you are in the Mobile App Builder environment, the banner includes the heading **Design Mobile Application** and the app name:

## Design Mobile Application: My Phone App

- The **Save** button is disabled until you make changes in the project.
- **Edit Page** tabs include the orientation and zoom controls:
- You now define the color scheme for all editors from the **Actions** menu. Select **Settings** to open the **Environment Settings** tab and select a scheme from the menu:

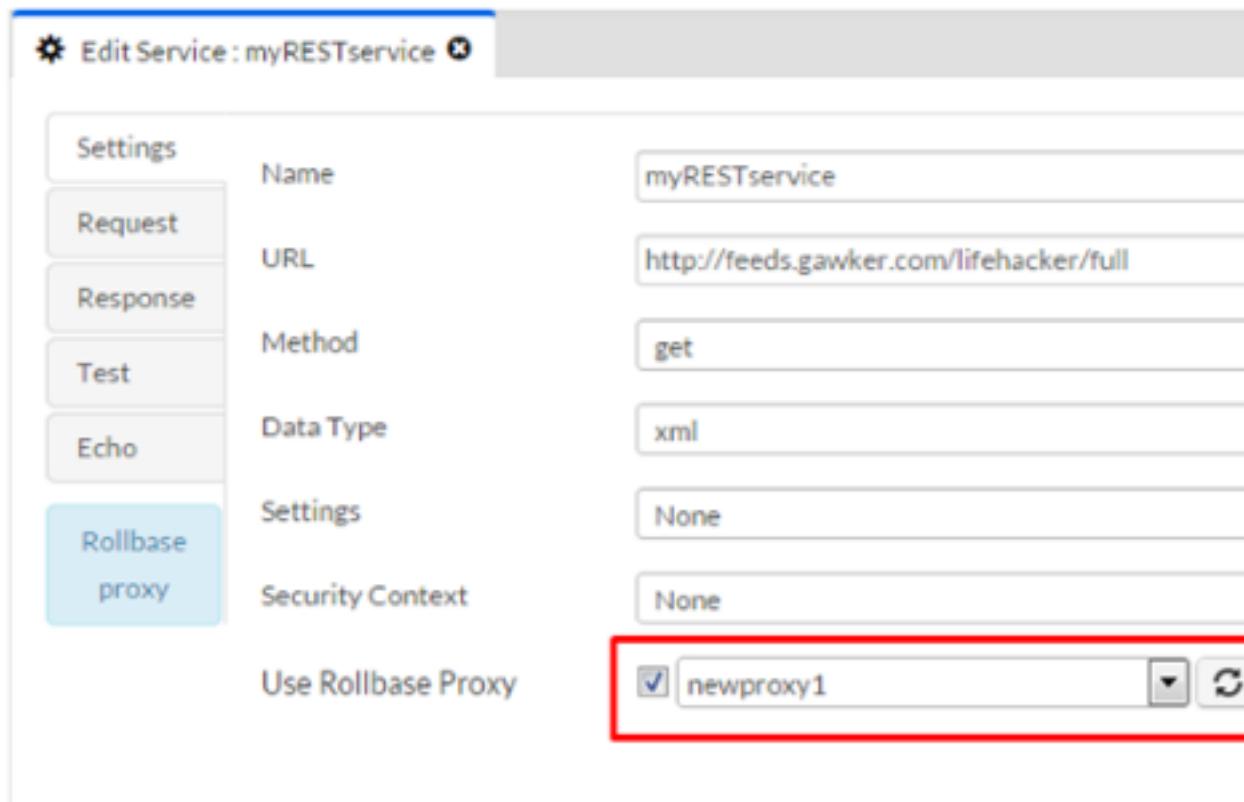


- When you add a component to a page, a quick create dialog allows you to set certain properties. For example, the Input component dialog allows you to set its **Name** and **Text** properties:



Check the box to disable this behavior if you commonly use the default values. You can enable it again from the **Actions** menu. Select **Settings** to open the **Environment Settings** tab. Select the **Project** side tab and check the **Add component quick create** box.

- When you create a new REST service, a dialog allows you to set the name. The **Settings** tab now allows you to use a Rollbase proxy, select an existing one, or create a new one. Using a proxy allows you to perform testing when cross-domain security would otherwise prevent it, as described in [Techniques for implementing and testing cross-domain services](#).



### Enhanced Documentation

Mobile App Builder documentation is now available in a responsive design HTML format that provides better navigation and other built-in functionality such as Google translate. Mobile app developers can access the new format from the **Help Me** link in the footer of the Mobile App Builder or from this link: <http://documentation.progress.com/output/mobile-app-builder>. Links within the development environment to documentation will also open the new format. Most, but not all content is available in the new format for this release. The remainder will be updated incrementally over time. Specifically, for tutorials that you do not find in the new format, see <http://docs.mobile.rollbase.com/tutorials/>, but note that some procedures and screen shots are out of date in the legacy content.

## 3.0 New Features and Enhancements

Rollbase 3.0 contains the following new features and enhancements:

- [Rollbase Mobile Apps and Web Apps](#) on page 61
- [Rollbase Usability Enhancements](#) on page 61
- [Metadata Search](#) on page 66
- [View all Invoices for a Tenant](#) on page 66
- [API Support for Accessing External Objects](#) on page 67
- [New Client-side Message Functions](#) on page 67

## **Rollbase Mobile Apps and Web Apps**

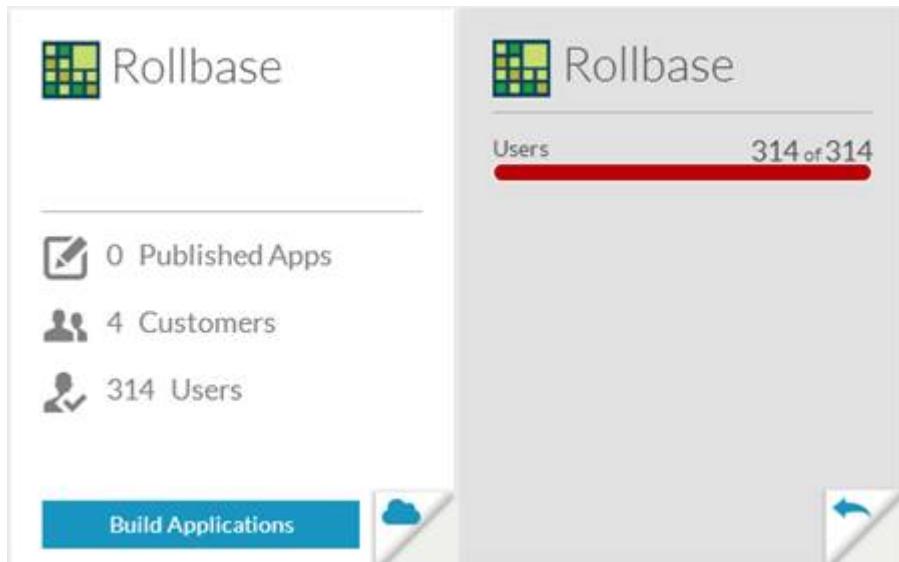
In this release, mobile apps are independent applications, similar to traditional Rollbase applications, which are now called Rollbase Web Apps. Therefore, when you create a new application in Rollbase, you can choose to create a Web App or a Mobile App. When you create a Mobile App, you specify whether you want to base it on an existing Rollbase Web App or not. If you base it on an existing Web App, the Mobile App has immediate access to the Rollbase objects in that Web App, and the services to access the views you choose are generated for you automatically. For Mobile Apps not based on Web Apps, you can use Application Setup to add existing Rollbase objects or create new Rollbase objects, then select the object views you want to access using generated services. Note that Mobile Apps initially based on Web Apps are not updated automatically if you make changes in the Web App. Mobile Apps are now independent of their base Web Apps, and you must update them separately using Application Setup just like Mobile Apps not based on Web Apps. Also, don't confuse this new terminology with Mobile Web Apps and Mobile Native Apps, which has to do with the way Mobile Apps are deployed, either as a Native App or as an App that users with mobile devices access using a browser.

See [3.0 Rollbase Mobile Changes and Improvements](#) on page 68.

## **Rollbase Usability Enhancements**

This version of Rollbase reflects the fresh look and feel of the Progress Pacific platform and a variety of usability enhancements, including the following:

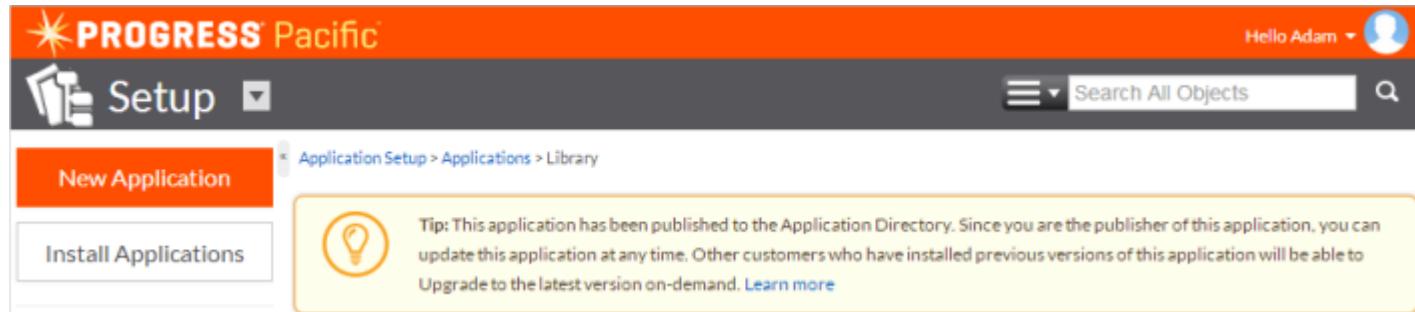
- For ISV users, when you log into the Pacific dashboard and flip your Rollbase Card to get a quick overview of the activity in your tenant, you will see an aggregation of your subtenant activity:



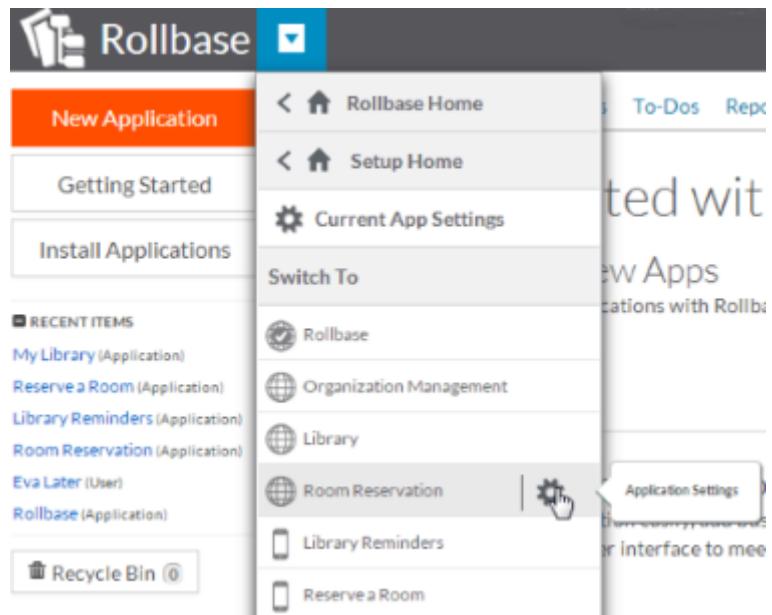
- The heading banner colors provide a visual contrast between application and setup pages. The banner remains visible as you scroll and allows you to access the menu of applications at all times.
- Application pages:



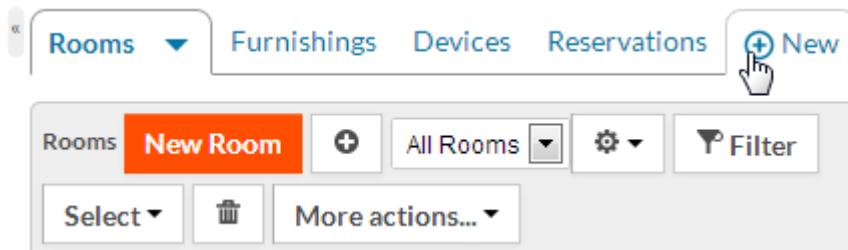
- Application setup pages (The **Rollbase** application and **Setup Home** have the same color banner.):



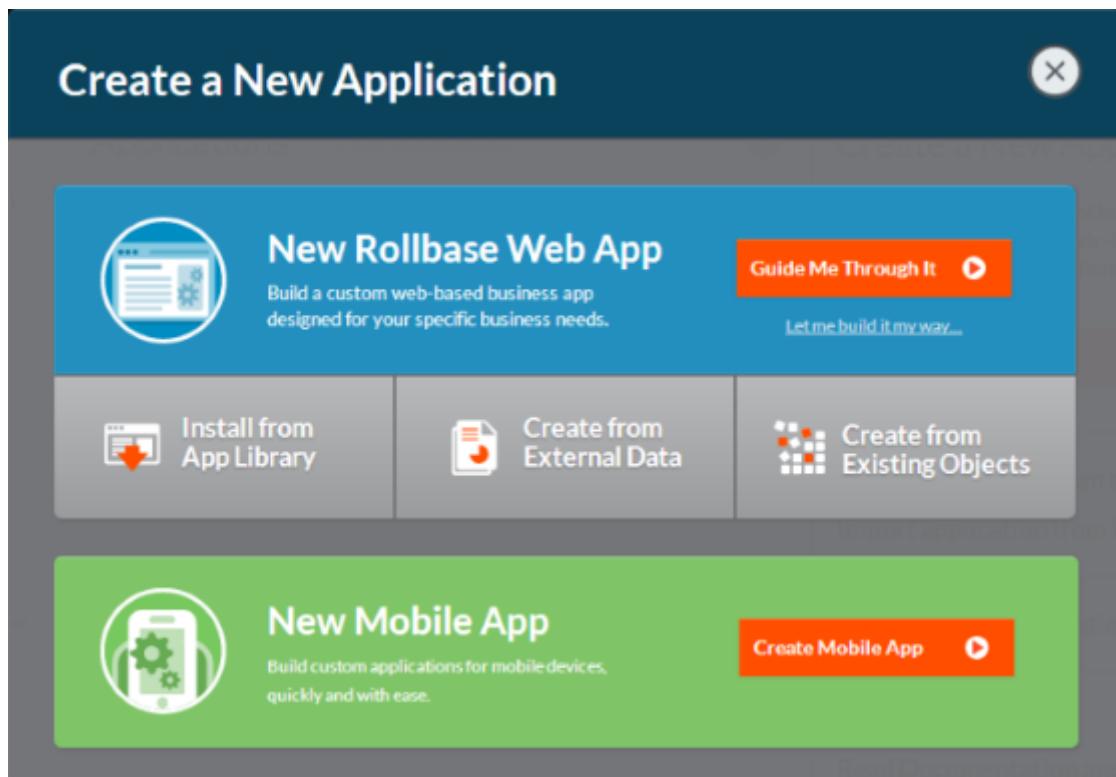
- Simpler navigation between applications and setup. The menu on the banner contains shortcuts to applications and setup:
  - Select **Rollbase Home**, **Setup Home**, or an application.
  - Click **Current App Settings** to see application setup for the application you are viewing.
  - To navigate to settings for a different application, select the application and click its **Application Settings** link.
  - Rollbase Web Apps and Mobile Apps are both available from this navigation menu. The globe icon identifies Web Apps and the mobile icon Mobile Apps.



- Application tabs have a new look, and the button to create a new object is more visible:

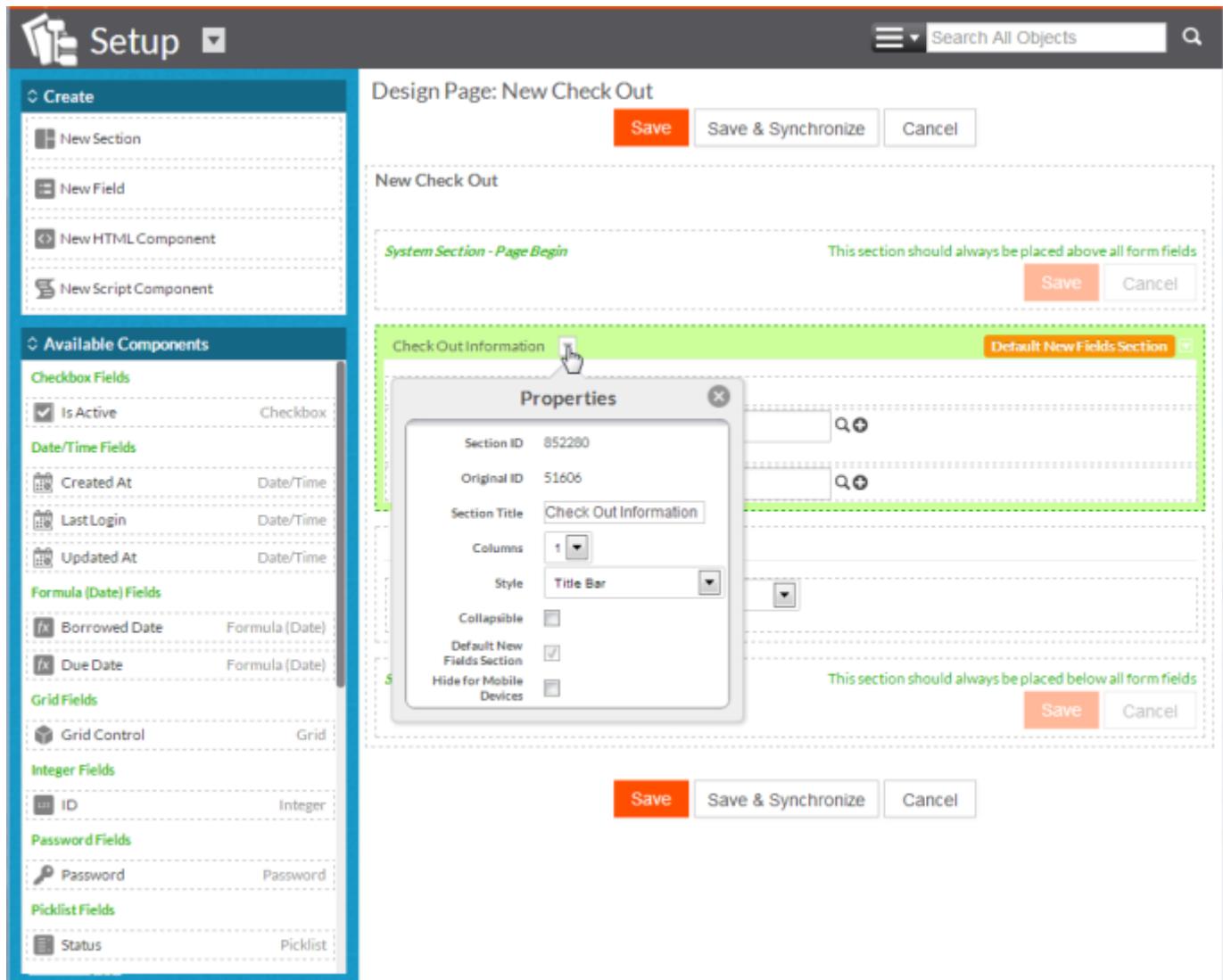


- The **New Application** button in the left sidebar launches the redesigned **Create a New Application** dialog, which includes new Mobile App creation screens:

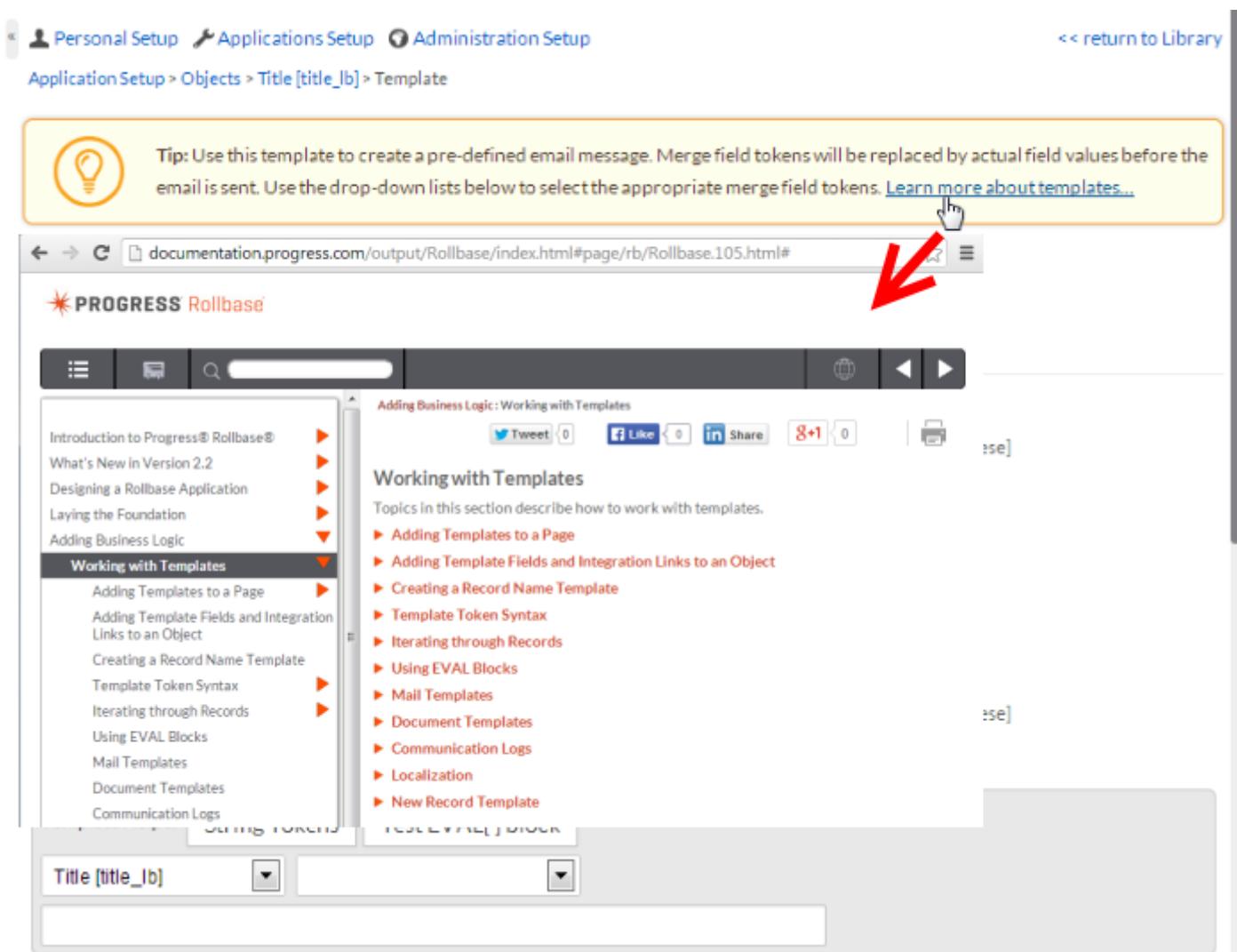


You will not see the option to create a new mobile app unless your plan includes mobile capability. See [Creating an Application](#) on page 104 for more information on the options available from this dialog.

- The **Page Editor** has a new look and feel. The **Properties** menu now opens from the title bar of each component:



- *Rollbase in Action* has been replaced with an updated *User's Guide* that is available in both HTML (online help) and PDF format. Most of the screen shots have been updated to reflect the latest interface. Instead of being organized according to feature, the main navigation takes into account the learning experience and development process. New content has been added to help citizen developers more quickly grasp Rollbase concepts.
- Links in the interface to help topics now open the online help instead of PDF files. In most cases, the links open to the point of interest instead of requiring you to search the PDF to find the relevant content:



Personal Setup Applications Setup Administration Setup << return to Library

Application Setup > Objects > Title [title\_lb] > Template

**Tip:** Use this template to create a pre-defined email message. Merge field tokens will be replaced by actual field values before the email is sent. Use the drop-down lists below to select the appropriate merge field tokens. [Learn more about templates...](#)

documentacion.progress.com/output/Rollbase/index.html#page/rb/Rollbase.105.html#

PROGRESS Rollbase

Adding Business Logic: Working with Templates

Working with Templates

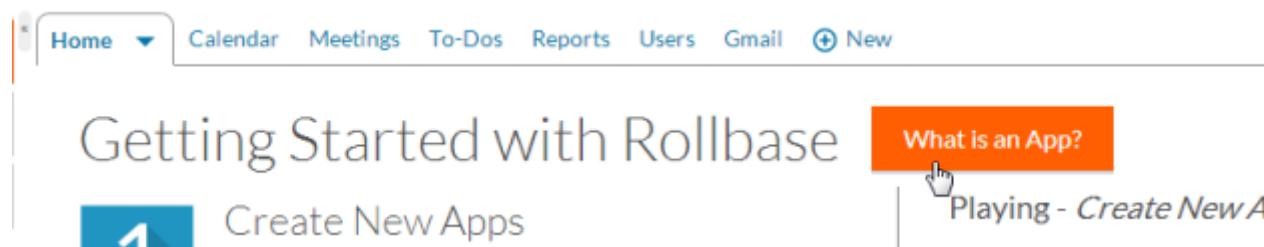
Topics in this section describe how to work with templates.

- ▶ Adding Templates to a Page
- ▶ Adding Template Fields and Integration Links to an Object
- ▶ Creating a Record Name Template
- ▶ Template Token Syntax
- ▶ Iterating through Records
- ▶ Using EVAL Blocks
- ▶ Mail Templates
- ▶ Document Templates
- ▶ Communication Logs
- ▶ Localization
- ▶ New Record Template

Adding Templates to a Page  
Adding Template Fields and Integration Links to an Object  
Creating a Record Name Template  
Template Token Syntax  
Iterating through Records  
Using EVAL Blocks  
Mail Templates  
Document Templates  
Communication Logs  
Localization  
New Record Template

Title [title\_lb]

- A new graphic illustrating the structure of a Rollbase Web App is available from the **Getting Started** page:



Home Calendar Meetings To-Dos Reports Users Gmail + New

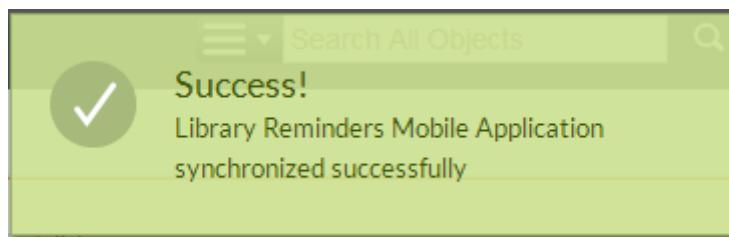
## Getting Started with Rollbase

Create New Apps

What is an App?

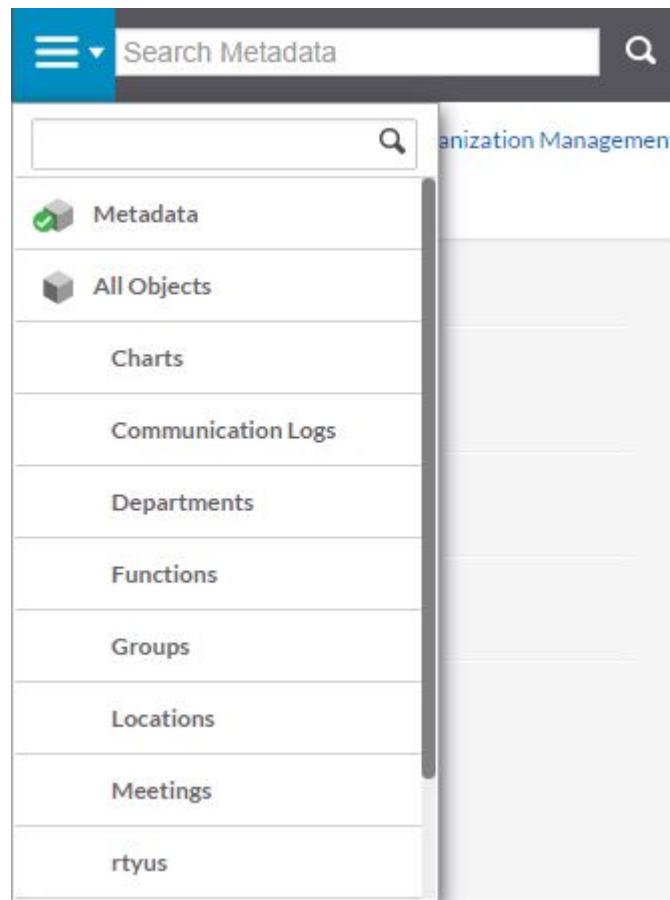
Playing - Create New A

- The appearance and behavior of messages has been improved, as illustrated by the following example:



## Metadata Search

Capability has been added to search application metadata such as field names. For example, you could find every formula that includes a reference to a particular field integration name. See [Metadata Search](#) on page 47 for more information.



## View all Invoices for a Tenant

If you are a Rollbase user that has been identified as the billing administrator, select the **Invoice History** option from the drop-down menu next to your profile avatar to view all the invoices generated for your tenant. You can filter the invoices using the **Data Range** field and hover over an invoice to download it as a PDF.

## API Support for Accessing External Objects

Methods in the SOAP and REST APIs now have a parameter specifying the integration name to support their use for external objects, including those mapped to OpenEdge Services. Use of server-side and AJAX APIs for external objects is available as beta functionality. Do not use the server-side or AJAX APIs for accessing OpenEdge Services in production systems. The [Reference](#) on page 559 page lists API categories by type. Within each type, the methods are listed alphabetically.

## New Client-side Message Functions

The following new methods provide finer-grained control over messages that display to end-users:

- [rbf\\_growl\(\)](#) on page 679
- [rbf\\_growlError\(\)](#) on page 679
- [rbf\\_hideGrowl\(\)](#) on page 680
- [rbf\\_growlInfo\(\)](#) on page 680
- [rbf\\_growlSuccess\(\)](#) on page 681
- [rbf\\_growlWarning\(\)](#) on page 682

# 3.0 Changed Behavior

The behavior of functionality described in this section has changed since the previous release:

- [Deprecated REST and SOAP Methods](#) on page 67
- [Backup storage](#) on page 67
- [OpenEdge Objects in Rollbase](#) on page 68
- [API Support for Accessing External Objects](#) on page 68

## Deprecated REST and SOAP Methods

The following methods have been deprecated. They will continue to work, but will not be supported in the future. Progress recommends that you use alternative methods if possible:

- The REST `create2` and `getDataObj` methods have been replaced with [createRecord](#) on page 743 and [getRecord](#), respectively.
- The REST `update2` method has been replaced with [updateRecord](#) on page 777.
- The following REST methods have been deprecated (some of these were for use in mobile apps, and have been replaced with Rollbase Mobile App Builder functionality): `appTree`, `recList`, `record`, `calendar`, `header`, `view`.
- The SOAP `getDataObj()` method has been replaced with [getRecord\(\)](#) on page 805.

## Backup storage

In past releases, Rollbase backup enforced a restriction of 10 MB per backup. In this release, you can create a backup of up to 1GB. See [Backup and Restore](#) on page 426 for more information.

## OpenEdge Objects in Rollbase

When working with OpenEdge service objects in Rollbase, you can now:

- Create relationships between Rollbase native objects and OpenEdge objects or between any two OpenEdge objects from different catalogs.
- Visualize data using charts.
- Use the Rollbase native filtering options—AND, OR, and Expression—that are enabled for OpenEdge objects. The AND native filter is the default filter criteria.

---

**Note:** In the earlier releases, for OpenEdge objects, **Search criteria** filter was the default filter as it was the only available filter option. So, if you were using the **Search criteria** filter prior to the Rollbase 3.0, on Rollbase 3.0 upgrade, you must manually edit those views/reports and specify the **Search criteria** filter as your choice of filter to retain the filter conditions (criteria and values) you had set.

---

For more information on relationships, see [Relationships Between Objects](#) on page 132. For information on charts, and gauges, see [Reports, Charts, and Gauges](#) on page 203. For information about enabling support for filtering and the available filtering criteria, see [Enabling support for filtering options and sorting](#) on page 328 and [Working with Views](#) on page 235.

## API Support for Accessing External Objects

Methods in the SOAP and REST APIs now have a parameter specifying the integration name to support their use for external objects, including those mapped to OpenEdge Services. Use of server-side and AJAX APIs for external objects is available as beta functionality. Do not use the server-side or AJAX APIs for accessing OpenEdge Services in production systems. The [Reference](#) on page 559 page lists API categories by type. Within each type, the methods are listed alphabetically.

# 3.0 Rollbase Mobile Changes and Improvements

This topic describes:

- [Rollbase Mobile Features](#) on page 68
- [Mobile App Builder Changes and Enhancements](#) on page 70
- [Rollbase Mobile App Builder Available for Private Cloud](#) on page 73
- [Migrating Mobile Apps Created with OpenEdge Mobile](#) on page 74

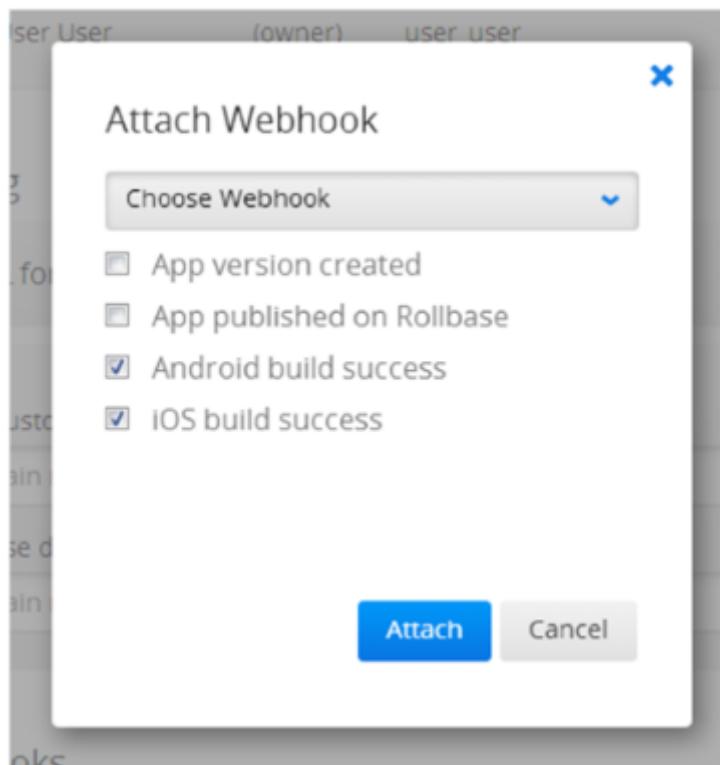
## Rollbase Mobile Features

This release includes the following changes and new functionality:

- **More options for using Rollbase objects as data sources** — When creating a mobile app, you can base that application on an existing Rollbase Web App and select the views to make available through the Mobile App. Rollbase Mobile automatically generates the services necessary to access the type of objects exposed in those views, including related objects. If you do not base a Mobile App on an existing Rollbase Web App, you can still take advantage of data stored by Rollbase and the automatic generation of JSDO services. You do this in the

Mobile App Setup by creating new objects or attaching existing objects and exposing views of those objects to the Mobile App. (cross-reference to details to be added).

- **Support for Server Code** — allows you to write custom JavaScript that Rollbase will execute. Any custom script can be invoked in a Mobile Apps using a REST service. You can create and edit scripts, test them, watch tracing and statistics information, create libraries, and define dependencies. To use Server Code script in your Mobile App you can easily create a REST service by selecting **Create New > Server Code Services**. See <http://docs.mobile.rollbase.com/documentation/backendservices/server-code/> for more information.
- **Import OpenEdge Mobile App into Rollbase** — You can now import a Mobile App project that you created using OpenEdge Mobile from the Mobile App Builder into Rollbase, where it will appear as a Rollbase Mobile App.
- **Offline/online Session Support** — The Instance Session Service provides two new events, **Offline** and **Online**, which fire when a Mobile App goes offline or back online (respectively) for one of the following reasons: 1) the device it is running in loses or regains access to the Internet, or 2) a JSDO that the Session Service supports has either lost or regained access to the server to which it is connected.
- **Selection of Cordova Plugins** — You can select and unselect the Cordova plugins to be used in your Mobile App (open App settings/Android Cordova plugins and App settings/iOS Cordova plugins). See <http://docs.mobile.rollbase.com/documentation/using-native-device-features-phonegap/> for more information on the plugins.
- **Support for Responsive Design** — A new Liquid component allows you to build a Mobile App with a responsive user interface. See <http://docs.mobile.rollbase.com/tutorials/building-mobile-app-with-responsive-ui/>.
- **User ID Template** — A new Push Notification UserID Settings template is available to add to your Mobile App. This template allows you to update the user ID in the database used for sending push notifications. See <http://docs.mobile.rollbase.com/documentation/plugins/update-userid-data-with-the-push-notification-userid-settings-template/>.
- **Custom Templates** — You can create custom templates to use when creating Mobile Apps. From the **Templates** tab you can find a list of your custom templates, usage statistics, and a way to open them for editing.
- **Webhook Integration** — A webhook is a method of augmenting the behavior of a service with custom callbacks. On the Integrations tab you can create Webhooks and specify a URL where POST requests should be sent. Then on the Apps tab you attach your Webhooks to a Mobile App. When any selected event occurs, an HTTP POST request invokes the webhook URL. See <http://docs.mobile.rollbase.com/documentation/webhooks/>



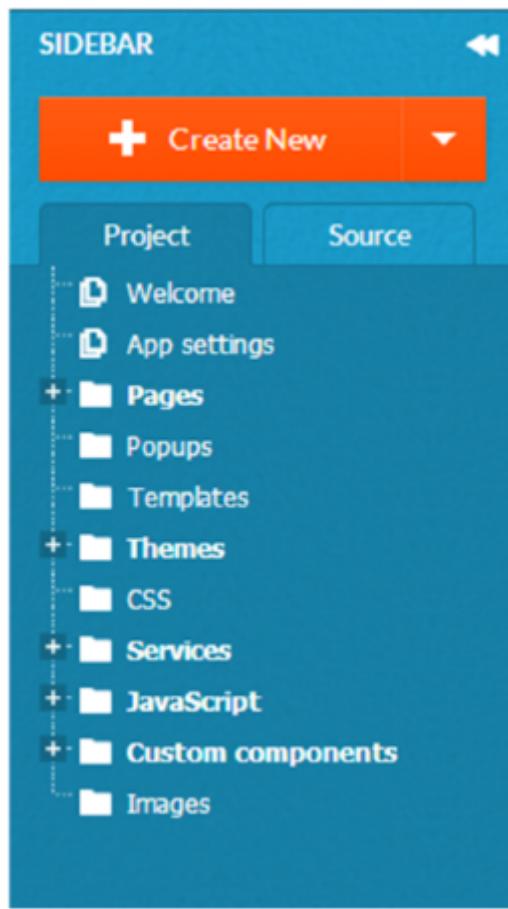
- **One user per app in Mobile App Builder** — You cannot share a Mobile App that you are designing in the Mobile App Builder with other users.
- **Page properties in One List** — On the **Edit Page** tab, all page and component properties display on the Details panel together. They are not divided in Custom and Common properties, but they all are listed in one block.
- **Adding Events** — To add events to a component, select the component and click **Add Events**. All events and actions appear on the panel. You can define the event and click **OK** to stop adding events, or click **Apply and Add Another** to define more events. To watch all events defined on the page, select **All Components** (on the **Design** tab) or **All Data Sources** (on the **Data Source** tab).

## Mobile App Builder Changes and Enhancements

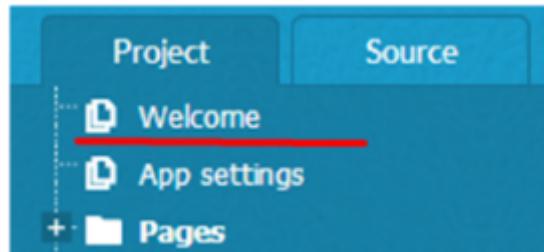
You will notice the following in the Mobile App Builder interface:

- The new banner and sidebar design.





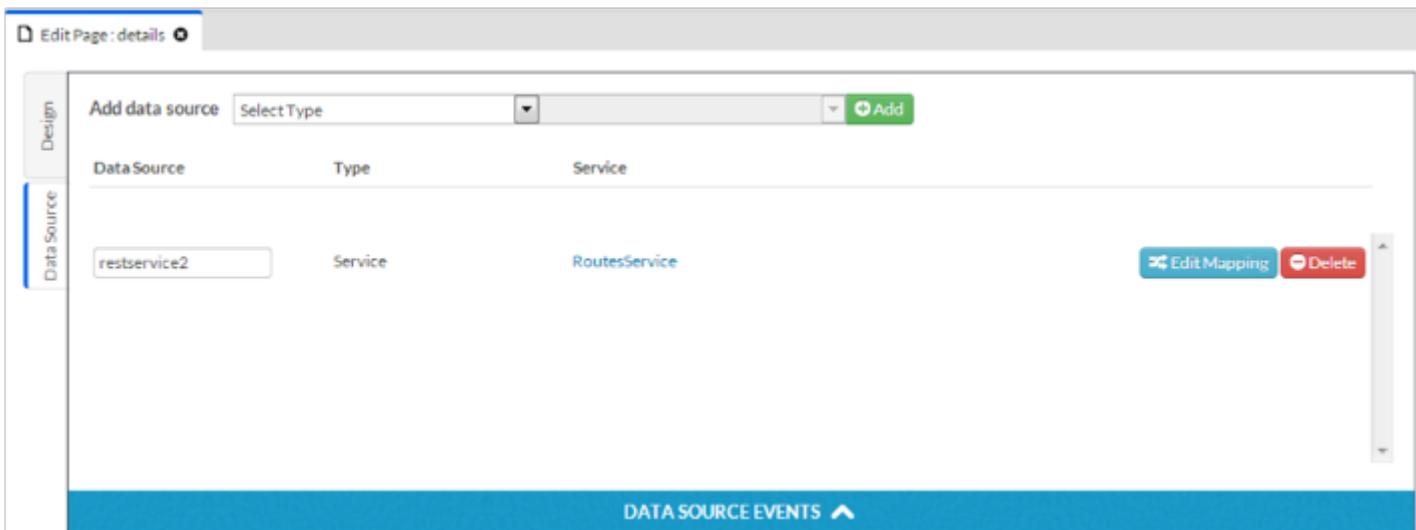
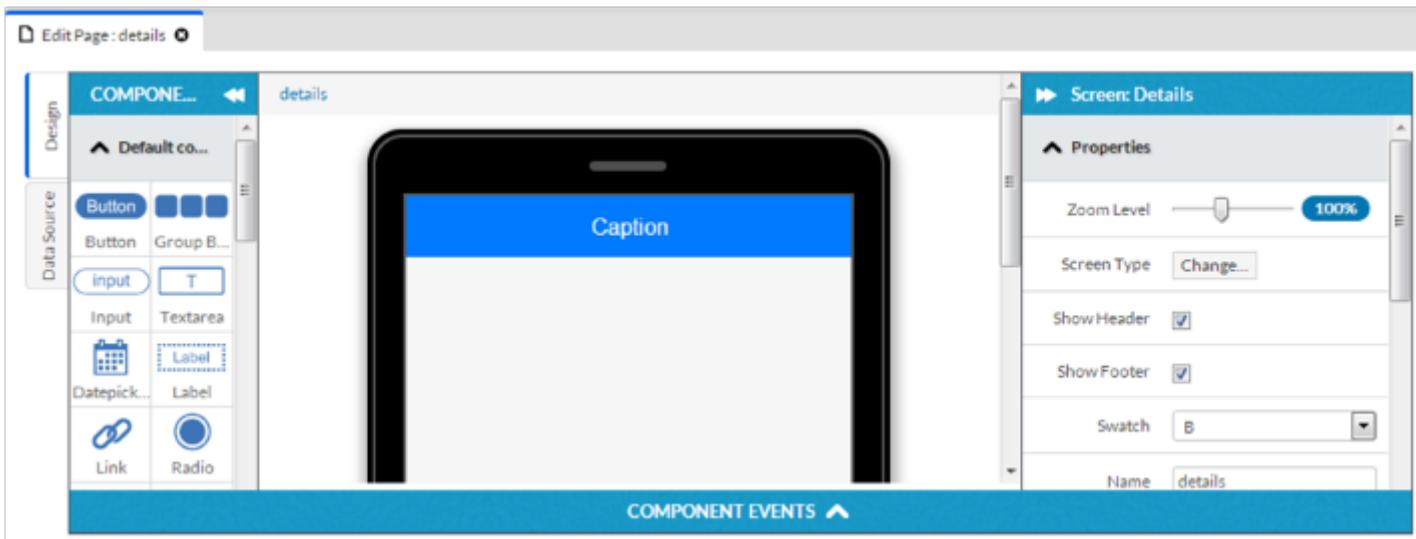
- The sidebar makes it easier to access the **Welcome** and **App Settings** screens:



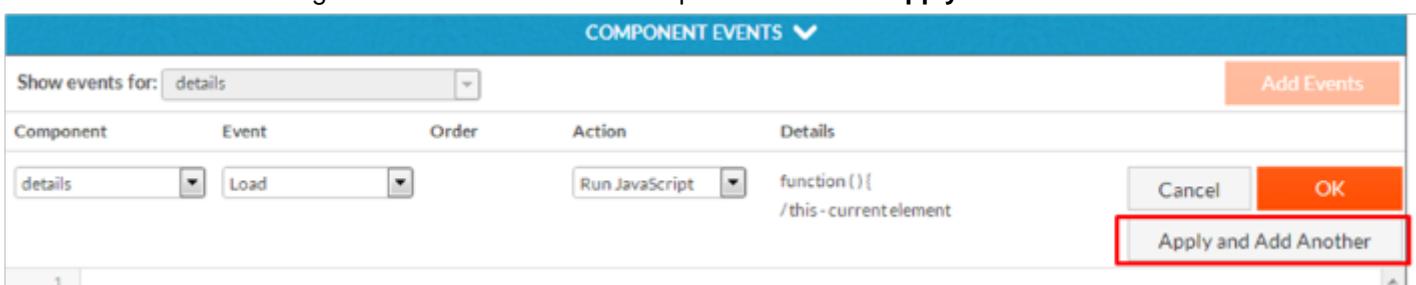
- Now you can easily log out from the Mobile Applications Monitoring Dashboard by clicking the **Logout** link in the user menu in the upper right corner:



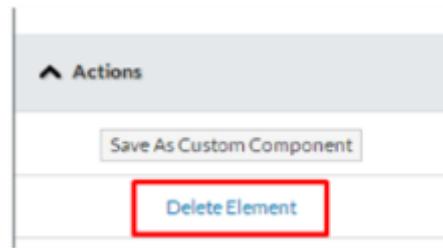
- On the the **Edit Page** tab, the **Component Events** and **Data Source Events** panels are more visible:



- Adding several events in a row is simpler with the new **Apply and Add Another** button:



- When designing the user interface, you can now delete an element using the **Delete Element** link on the **Details** panel:



- The **Test** menu has a new look. Instead of the **Public** check box, it has a toggle option to **Make test URL public/private**. Instead of the field with link, it allows you to **Copy test URL**:



## **Rollbase Mobile App Builder Available for Private Cloud**

Private Cloud users can now take advantage of new Rollbase Mobile functionality. When you create an application, you can now choose to create a Mobile app. After you enter the Mobile app definition and click **Save & Design**, you will be taken to the Mobile App Builder. You need to have a Progress ID to access the Mobile App Builder, as described in [Enabling Mobile App Development in Rollbase Private Cloud](#) on page 300.

The new Mobile app can be associated with an existing Rollbase Web app, in which case the services necessary to access the views you select will be generated for you automatically. Once the Mobile app is created, changes to the Rollbase Web app cannot be synchronized to the Mobile app and vice versa. However, from within the Rollbase tenant, you can add new objects to the Mobile app and synchronize them with the Mobile App Builder. See [Rollbase Mobile](#) on page 272 for more information.

---

**Note:** On Rollbase Private Cloud, to implement Push Notifications with a Mobile App, use <https://api.appdesigner.rollbase.com> as the base address instead of <http://api.mobile.rollbase.com> as is documented at:

[http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push\\_notifications\\_API](http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push_notifications_API).

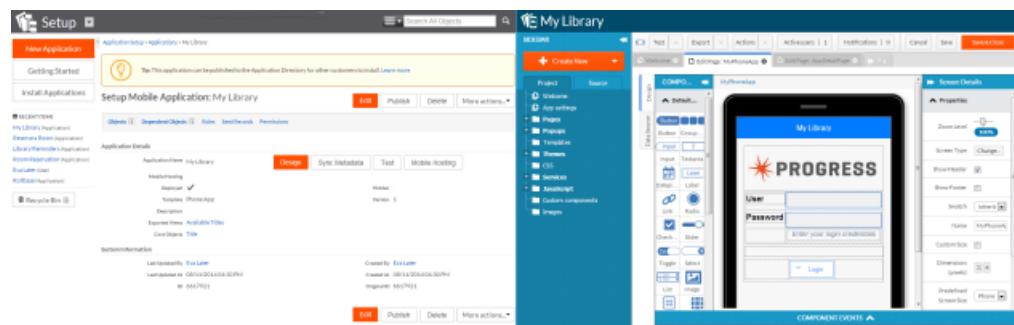
---

## Migrating Mobile Apps Created with OpenEdge Mobile

Progress Customers with projects previously built with OpenEdge Mobile will be able to take advantage of new capabilities in the Rollbase Mobile App builder such as Session Services and the login page. Please see [procedures for upgrading](#) and the white papers posted at <https://community.progress.com/technicalusers/w/mobile/2186.openedge-mobile-white-papers.aspx>.

To enable projects built with OpenEdge Mobile to take advantage of data stored (or to be stored) in Rollbase, they must be imported into Rollbase. This will result in a Rollbase Mobile App definition in the Rollbase environment that is associated with the project in the Mobile App Builder. The migration procedure includes creating a backup file of the original project and importing it into Rollbase to create the Rollbase Mobile App definition. See [Importing Mobile App projects created with OpenEdge Mobile](#) on page 303

The following illustration shows the Mobile App definition in Rollbase and the project in the Rollbase Mobile App Builder:



## 3.0 Private Cloud Features and Changes

The Rollbase Private Cloud 3.0 release includes the following new and changed behavior:

- [Progress® Pacific® Application Server](#) on page 75
- [Rollbase Mobile App Builder Available for Private Cloud](#) on page 75
- [A Production Server Can be Added to Private Cloud Without Restart](#) on page 75
- [Private Cloud Configuration Files Stored in Database](#) on page 75
- [Disabling or Enabling the New Pacific Look](#) on page 76
- [Option to Disable SSL Transport to Third-Party Storage for ISVs and Private Cloud](#) on page 76
- [OpenEdge Single Point of Authentication](#) on page 76

**Note:** A PDF version of the Rollbase User's Guide is included in the Private Cloud installation. However, Progress updates the online help version with important information, so you should always check there for the latest content. You can find the online help at: <http://documentation.progress.com/output/Rollbase/>

## Progress® Pacific® Application Server

The Private Cloud Installer now gives you an option to install the Pacific Application Server (PAS). PAS is based on Apache Tomcat®, but is tailored to be secure for use in production systems. PAS also simplifies the process required to create and run multiple Rollbase components on different hosts. See [Introduction](#) on page 441 for more information.

## Rollbase Mobile App Builder Available for Private Cloud

Private Cloud users can now take advantage of new Rollbase Mobile functionality. When you create an application, you can now choose to create a Mobile app. After you enter the Mobile app definition and click **Save & Design**, you will be taken to the Mobile App Builder. You need to have a Progress ID to access the Mobile App Builder, as described in [Enabling Mobile App Development in Rollbase Private Cloud](#) on page 300.

The new Mobile app can be associated with an existing Rollbase Web app, in which case the services necessary to access the views you select will be generated for you automatically. Once the Mobile app is created, changes to the Rollbase Web app cannot be synchronized to the Mobile app and vice versa. However, from within the Rollbase tenant, you can add new objects to the Mobile app and synchronize them with the Mobile App Builder. See [Rollbase Mobile](#) on page 272 for more information.

---

**Note:** On Rollbase Private Cloud, to implement Push Notifications with a Mobile App, use <https://api.appdesigner.rollbase.com> as the base address instead of <http://api.mobile.rollbase.com> as is documented at:

[http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push\\_notifications\\_API](http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push_notifications_API).

---

## A Production Server Can be Added to Private Cloud Without Restart

In past releases, Private Cloud users needed to stop and restart Rollbase to add new production servers. In this release, you can add the production server configuration through the UI and then start the new production server, as described in [Adding Production Servers](#) on page 492. Now Rollbase configuration files are stored in the database instead of requiring each Rollbase instance to have a copy of the configuration files. The files in the Master Server `config` folder are still the authority, you can manually change them or change them through the interface. If you change them manually and restart, they are stored in the database so that all instances will pick up the changes. See the related information in the next topic, [Private Cloud Configuration Files Stored in Database](#) on page 75.

## Private Cloud Configuration Files Stored in Database

In previous releases, each Private Cloud Rollbase instance needed a copy of the license and of configuration files such as `shared.properties` and `databases.xml`. In this release, the configuration files are stored in the database and you only need a copy of the license file in the `config` folder of the Master Server.

This behavior change has the following implications, which you should note:

- You still make configuration changes through the interface or manually to the files. You cannot edit the copies stored in the database.
- The Tomcat instance running the Master Server must be started first so that the files are stored properly in the database. PAS takes care of this for you, however if you are using your own Tomcat instance, and you have more than one Rollbase WAR that will run in the same instance, you need to do the following:
  1. Create a `master.xml` file that contains one element: `<Context></Context>`
  2. Save the file in the `$TOMCAT_HOME/conf/Catalina/localhost` folder.

---

**Note:** If the Master Rollbase WAR is installed on its own Tomcat server, you don't need to create the `master.xml`; just make sure that the server hosting the master web application is started and available for requests before starting the other servers.

---

## Disabling or Enabling the New Pacific Look

A new `PacificUIDisabled` property in the `shared.properties` file controls whether the UI has the Pacific look and feel or the classic look and feel. This value defaults to false for new installations. A value of true maintains the classic look and feel.

## Option to Disable SSL Transport to Third-Party Storage for ISVs and Private Cloud

ISVs and Private Cloud users can optionally store hosted files on other cloud services such as Amazon S3 and Microsoft Azure. Rollbase uses SSL by default to transfer data to third-party cloud storage accounts. While SSL should always be used in production systems, you might want to temporarily disable SSL for debugging purposes. To do that, you need to update the **Rollbase System Console** application to version 3, and the checkbox described in [Using a Third-Party Cloud Service for Storage](#) on page 551 will be available to disable SSL.

## OpenEdge Single Point of Authentication

Rollbase Private Cloud now enables OpenEdge users to log into Rollbase using their OpenEdge AppServer credentials. Therefore, the Rollbase administrative user can now set and configure OpenEdge Authentication as their choice of authentication method. For more information about setting the OpenEdge Authentication, see [Setting the Authentication Method](#) on page 378, [Linking a Rollbase Object to OpenEdge Data](#) on page 324, and [Creating an Application from OpenEdge Data](#) on page 326.

# Upgrading Rollbase Private Cloud

To migrate data from previous releases of Rollbase, you need to run scripts to update your database sequentially. That is, to update from version 2.1.x, you would need to first upgrade to 2.2, to 3.0, and then to 3.1. In addition, be sure to check [3.0 New Features and Enhancements](#) on page 60, [Rollbase Changed Behavior](#), and [3.1 Rollbase New Features and Changes](#) on page 53 to determine whether to change code in formulas or external systems that access Rollbase data through the SOAP or REST APIs.

## Upgrading to Version 3.1

You cannot install Rollbase Private Cloud 3.0.x until you have uninstalled the previous version. Due to a change in search functionality for version 3.0.5, Private Cloud users upgrading from a release prior to 3.0.5 must re-index all teants. Re-indexing must be performed on the master server by a user with the role of administrator. After you have upgraded your Private Cloud installation as described in (cross-reference to general steps for upgrading) follow these steps to re-index

The general steps to upgrade from from a prior release include the following:

1. Make a copy of your existing Rollbase config folder.
2. Stop Tomcat.

3. If you have any custom files, such as FusionCharts, in the `webapps` folder of your Tomcat instance, copy these to a folder for later use.
4. Stop the database.
5. If you installed using the Rollbase installer, run `uninstall`. You can launch it from the `Rollbase\uninstall` folder. If you were using the OpenEdge database, `uninstall` will not delete the OpenEdge working directory, which contains your data.
6. Remove Tomcat folders:
  - a. If you used the Tomcat that came with the installer, delete the `\apache-tomcat-<version>` folder.
  - b. If you used your own Tomcat instance, delete the following folders from the `webapps` folder: `master`, `router`, `prod1`, `search`, `storage`, `rest`, `webapi`, `rss`, `workflow`
7. Install Rollbase 3.1 using the installer or zip files:
  - a. If you use the installer, consult the configuration files you saved for host, database, and email configuration values. If you were using the OpenEdge Database, point the installer at the working directory that contains your data.
  - b. If you use the zip files to install, configure Rollbase as follows:
    - a. From the `config` folder, open the `shared.properties` and `databases.xml` files.
    - b. Use the files you copied as references and make any necessary updates. At a minimum, in the `shared.properties` file, enter the email information: host name and port number, and the email address and password for the main administrative user.
8. If you saved any custom files from the `webapps` folder as described in step 3, copy them into the new installation.
9. In the `Rollbase\sql` folder, find the appropriate script to update your database. Note: If you installed the Pacific Application Server for Rollbase, these files are in the `Pas_Instance\Rollbase\sql` folder.
  - If you are upgrading from 3.0.X to 3.1, use the `update_3_1_0.sql`.
  - If you are upgrading from an earlier version, you will need to use multiple scripts, starting with the next version higher than your existing installation and continuing until you reach the `3_1_0` script.
10. Open the sql script(s), and do the following:
  - a. Uncomment the section for your database type.
  - b. Verify that the sections for other database types are commented out.
  - c. Add the following commit statement to the end of the file: `COMMIT WORK`
11. If you are using a database other than the OpenEdge instance included with the installer, start your database. If you are using the OpenEdge database that comes with the installer, it should be running already.
12. Start the Pacific Application Server or Tomcat.

## Re-indexing Customer Tenants

Follow these steps to re-index customer tenants:

1. Log into the master server as an administrator.
2. Navigate to the **System Console**.
3. From the **System** tab menu, select **Reindex**.
4. On the **Reindex** page, select the appropriate scope:
  - Master zone
  - All customer tenants
  - Selected customer tenants

You can select the master zone and all or selected tenants. Progress strongly recommends reindexing the master zone and all customers, but reindexing customer tenants one or a few at a time can minimize performance impact.

Confirm that indexing has occurred by doing the following:

1. Navigate to **Setup Home**.
2. In the **Administration** section, click **Global Text Search**.
3. Click **View Search Logs**.

---

# 3

## Designing a Rollbase Application

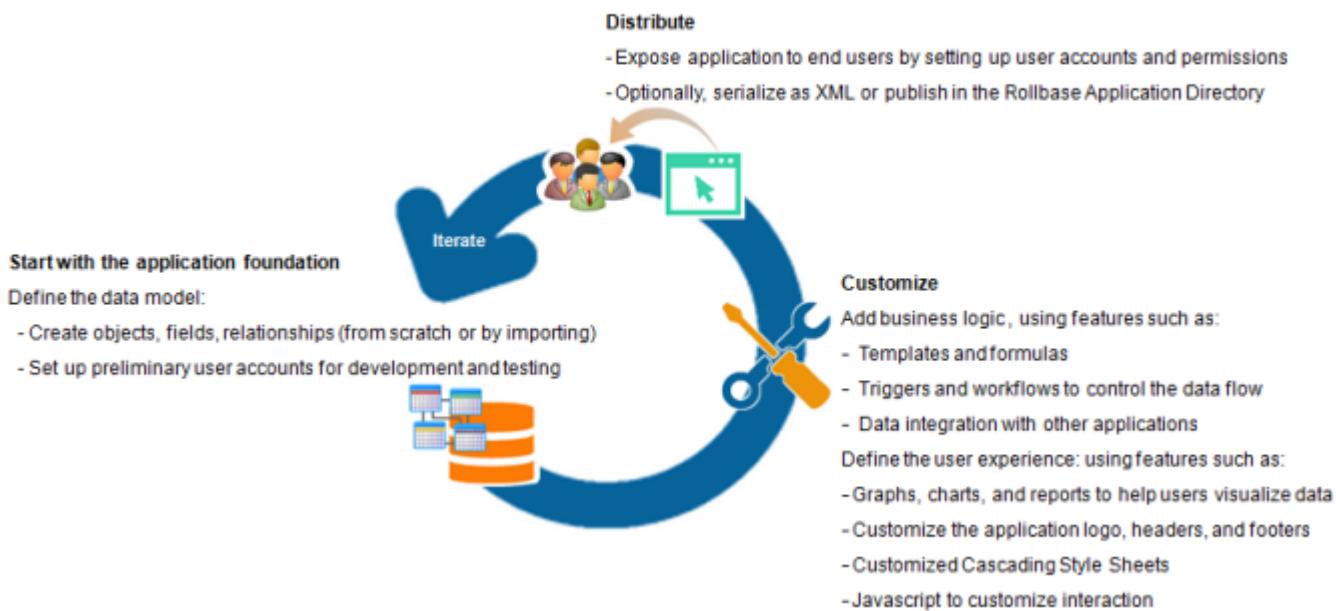
---

The core components of Progress Rollbase applications are objects, tabs and portals. Each of these components itself consists of sets of configurable sub-components such as fields, pages, and menus. You define these application components to form a fully functional SaaS solution.

As shown in the following graphic, the basic steps involved in creating a Rollbase application include:

1. Think about how your application will be exposed or distributed to users, through the public cloud, private cloud, as an application, and/or through a portal.
2. Create the application foundation, including the objects and relationships that define the data model.
3. Add application logic to derive business value from the data.
4. Define and customize the user interface.
5. Define permissions and security.
6. Test and deploy.

## Rollbase Development Process Overview



Of course, as with development of any application, these steps do not necessarily follow in order and are iterative. The following topics discuss each step in more detail:

For details, see the following topics:

- [Distribution Options](#)
- [Web Application Foundation](#)
- [Business Logic and Customizing the User Experience](#)

## Distribution Options

Rollbase offers hosted and private cloud environments:

- Progress Software manages and maintains the hosted Rollbase environment.
- You manage private cloud environments by installing Rollbase on your own internet resources or on any cloud platform.

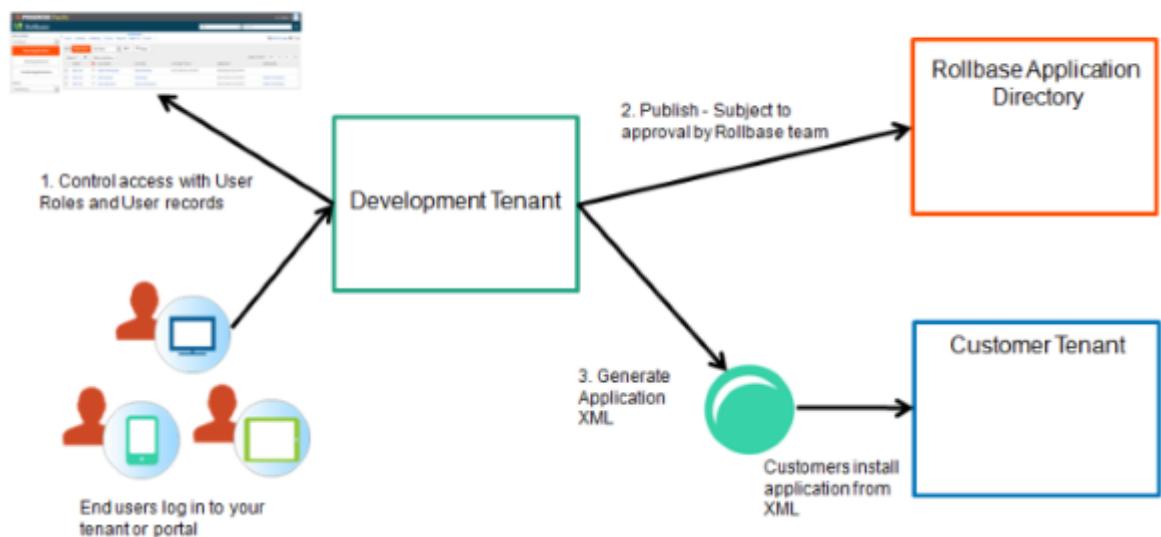
In both hosted and private cloud environments, Rollbase tenants are virtual spaces that define settings and applications for a group of users. The person who initially signs up for a Rollbase account becomes an administrator for a tenant. A tenant can act like a development sandbox and be used mainly by one or more developers. Or, a tenant can be used for development and when applications are stable, they can be exposed to end-users. You can also create Mobile Apps and distribute them to users as native mobile apps or as browser-based apps.

As the figure below illustrates, you can provide access for end-users to a Rollbase Web app in the following ways:

- By creating accounts for them to access the application in your tenant. This includes creating **User** records and setting application permissions.

- By creating a portal that exposes all or a subset of application functionality. See [Portal Overview](#).
- By generating the application as XML, which can be installed in any Rollbase tenant.
- By publishing to the Rollbase **Application Directory**, which gives any Rollbase user in any hosted tenant access to the application.

### Rollbase Application Publishing Options



If you plan to distribute your application to other tenants, Progress recommends thinking about installation and upgrades early in the design process. Try publishing your application to the **Application Directory** or generate it as XML and then test the installation before distributing it to others. See [Design and Development Considerations for Distributing as XML](#) on page 408 for more information.

### Portal Overview

Portals provide a flexible way to build and expose external-facing applications in public websites or intranets for external users, such as website visitors, partners, or employees on an intranet. With Progress Rollbase, you can create portals to allow just about any type of external user to access specific application functionality such as creating, editing, searching and viewing object records. Portals consist of an arbitrary number of pages that form a cohesive website. Portals can be designed to adopt the look and feel of any website.

Portals often expose a subset of Rollbase application functionality to a specific set of users. For example, a Rollbase application for managing sales leads might have a portal integrated with your website to collect information submitted by website visitors. An application for employee management might have a portal for self-service access to employee profile information, company directory, and benefits information.

For more information on designing and developing portals, see [Rollbase Portals](#) on page 257.

# Web Application Foundation

To create a Rollbase Web application foundation, you define application components. Component definitions consist of properties and attributes, and for some components, subcomponents (child components). Rollbase stores component definitions as metadata. With the appropriate permissions, external applications can access application metadata using SOAP or REST. See [Using SOAP or REST to Integrate with Rollbase](#) on page 371 for more information.

This section introduces and describes components that lay the foundation for a Rollbase application:

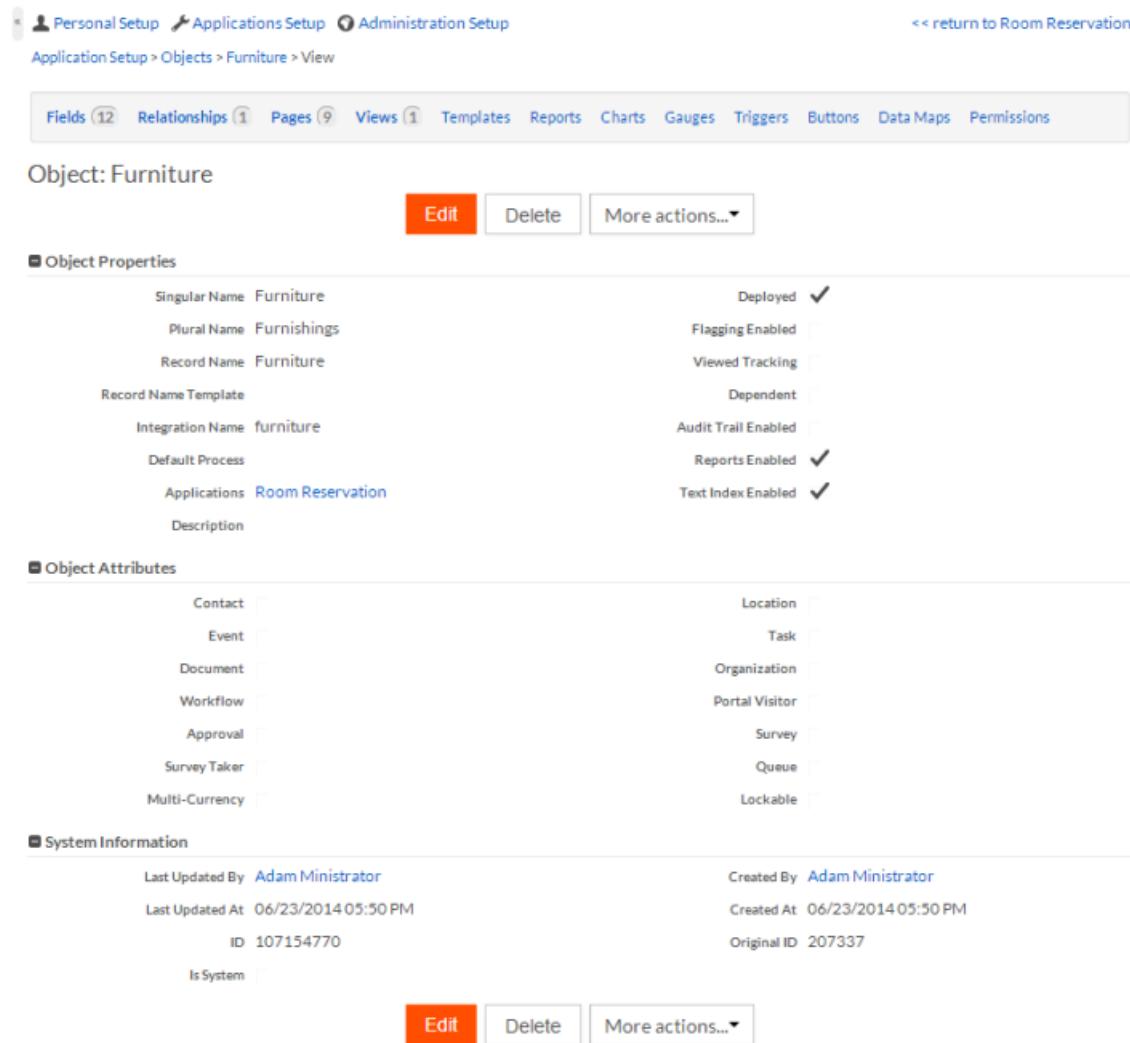
- Applications: A container for a logical grouping of objects, exposed to users through tabs, menus, and optionally, portals.
- Objects: The core component of Progress Rollbase applications, each object defines the structure for data of a particular type. Objects and relationships between objects create the application data model. Rollbase stores object data as records.
- Fields: The core component of objects, each field defines characteristics such as data type for a discrete piece of data. Users enter record data into fields, such as Name and Address.
- Relationships: Link objects together. For example, a shipment might be related to the individual ordered items that comprise that shipment.
- Pages, Tabs, and Menus: Pages make up the user interface of a Rollbase application. When you create an object, Rollbase creates a set of pages for creating, viewing, and editing records. Tabs and menus are the navigation controls associated with pages. Rollbase creates pages, tabs, and menus for you, but you have the option to customize them and to create others to enhance the experience of your end-users.
- Portals: Within an application, a way to expose all or subsets of Rollbase application functionality to end users and other external entities. Instead of logging into Rollbase to access application functionality, users log into the portal, which can be integrated into your website. You create the portal pages and control the user experience.
- Calendar: A built-in application that contains Task and Event objects that you can use to coordinate work for yourself and/or for a group of Rollbase users.

Watch a short video to see how quickly you can create an application with objects and relationships: [The Quick Create Wizard](#). The topics in this section provide more details on objects.

## Object Definition Overview

Object definitions are containers for fields. They have attributes and properties that specify how the individual records are structured, will behave, and can be accessed. Object definitions also contain the following child component definitions: fields, relationships, views, templates, pages, reports, charts, gauges, and triggers.

Object definitions specify read/write permissions per role and per user. They also have a complete administrative audit trail, allowing you to see the who, what, and when of any change that occurred to an object definition, its components, or the associated records. The following screen shows an example object definition for Furniture records. The ribbon of **Fields**, **Relationships**, and other components, contains links to jump to the section of the object definition where those child components are defined.



The screenshot shows the 'Furniture' object configuration in the Progress Rollbase application setup. The top navigation bar includes links for Personal Setup, Applications Setup, Administration Setup, and a 'return to Room Reservation' link. The main navigation bar shows 'Application Setup > Objects > Furniture > View'. Below this is a toolbar with links for Fields (12), Relationships (1), Pages (9), Views (1), Templates, Reports, Charts, Gauges, Triggers, Buttons, Data Maps, and Permissions. The main content area is titled 'Object: Furniture' and contains three sections: 'Object Properties', 'Object Attributes', and 'System Information'. The 'Object Properties' section includes fields for Singular Name (Furniture), Plural Name (Furnishings), Record Name (Furniture), Record Name Template, Integration Name (furniture), Default Process, Applications (Room Reservation), and Description. The 'Object Attributes' section lists Contact, Event, Document, Workflow, Approval, Survey Taker, Multi-Currency, Location, Task, Organization, Portal Visitor, Survey, Queue, and Lockable. The 'System Information' section shows Last Updated By (Adam Ministrator), Last Updated At (06/23/2014 05:50 PM), ID (107154770), Is System, Created By (Adam Ministrator), Created At (06/23/2014 05:50 PM), Original ID (207337), and a 'More actions...' button.

You can create object definitions using the **Quick Create** wizard, from scratch, by importing from spreadsheets or other applications, or by linking to data stored in other applications. An import usually brings in records and if those records do not map to existing object definitions, as a side effect, new object definitions can be created. Those procedures are described in:

- [Getting Started with the Quick Create Wizard](#)
- [Creating and Managing Objects, Fields, and Relationships](#)
- [Importing to Create a New Object](#) - describes how to create an object from a spreadsheet.
- [Linking a Rollbase Object to OpenEdge](#)
- [Using DataDirect Cloud to Access External Data](#)
- [Creating Rollbase Applications from Microsoft Access](#)
- [Creating Rollbase Applications from Salesforce Applications](#)
- [Using External Tables as Rollbase Objects](#)

Object definitions can be shared among applications. Records for a particular object in a tenant will be available in any application that includes that object. When you publish an application, you can choose to add sample records as **seed records** for the users who will install the application.

Once an object definition exists, in addition to editing and adding fields and properties, you can do the following:

- Clone an object: create a new object definition with a new name and copy all components, fields, triggers, etc. to the new object.
- Create and run triggers: select one or more triggers to run on all object records (for data maintenance etc.) See [Trigger Overview](#) on page 179 for more information.
- Define conditional formulas: formulas to enable/disable "Edit" and "Delete" functionality for object records.

## Object Attributes

Object attributes add behavior and fields to an object. For example:

- Objects with **Task** and/or **Event** attributes appear in the Rollbase Calendar.
- Objects with the **Document** attribute can store a file attachment. The file's contents are indexed in the full text search engine.
- Objects with the **Workflow** attribute have fields for managing workflow processes, statuses, and actions, allowing you to model many kinds of business processes.
- Objects with the **Location** attribute have fields to capture addresses.
- Objects with the **Contact** attribute have fields to capture phone numbers and e-mail addresses.

During object creation, you have the option to select which attributes to include. These are divided into two categories: **Attribute** and **Advanced Attribute**. If you use the **Quick Create** wizard, you can add attributes, but not advanced attributes. Once an object exists, you can add any type of attribute to it.

When you add an attribute during object creation or later, Rollbase adds a group of fields to that object. While you cannot delete these fields, except for the integration name property, you can modify all other properties. These new fields are added to the object view, create and edit pages in a new section. When you remove an attribute from an object, Rollbase will delete these fields as a group.

### Attributes

The following describe object attributes:

- **Contact**: adds fields to store contact information for people or organizations. You can send contact records by email and export them in the standard vCard format. The following fields are added when this attribute is enabled: **First Name**, **Middle Name**, **Last Name**, **Job Title**, **Phone**, **Mobile Phone**, **Fax**, **Email Address**, and **Contact Owner**.
- **Location**: adds fields to store information about people, places, or organizations with associated location and street address information. The following fields are added when this attribute is enabled: **Street Address 1**, **Street Address 2**, **City**, **State/Province**, **ZIP/Postal Code**, and **Country**. Using the **Location** attribute allows you to add a Google Map component to view pages, as described in [Google Maps](#) on page 370.
- **Event**: adds fields to schedule activities or meetings that have an associated start time, duration and group of participants. You can display, create, and manage **Event** records in the Rollbase Calendar and export them in the standard iCalendar format. The following fields are added when this attribute is enabled: **Event Subject**, **Start Date/Time**, **Duration**, **All Day**, **Private**, **Description**, **Location**, **Assigned To** and **Pop-up Reminder**.

- **Task:** adds fields to track deadlines, follow-ups, or to-dos that have a due date and one or more associated assigned users. You can display, create and manage **Task** records in the Rollbase Calendar view and export them in the standard iCalendar format. The following fields are added when this attribute is enabled: **Task Subject**, **Due Date**, **Priority**, **Private**, **Description**, and **Assigned To**.
- **Document:** adds a field for file attachments and a description. Rollbase indexes attached files in standard Word, Excel, PDF, or plain text formats for full text search. The following fields are added when this attribute is enabled: **Document File** and **Description**.
- **Organization:** adds lookup fields for associating records with organizational **Location**, **Department**, and **Function**.

## Advanced Attributes

The following describe advanced object attributes:

- **Workflow:** objects with this attribute can be routed through an automated or manual workflow process defined by a set of workflow statuses and actions. The following fields are added when this attribute is enabled: **Workflow Process**, **Workflow Status**, and **Workflow Actions** (list of available actions). See [Workflow Overview](#) on page 194 for more details.
- **Portal User:** objects with this attribute represent registered users of one or more portals. The **Portal User** fields can be used to require portal authentication and enable the creation of rich interactive portals that expose specific capabilities of your Rollbase applications to external users. The following fields are added when this attribute is enabled: **Login Name**, **Password**, and **Is Active**. See [Portal Security](#) on page 264 for more details.
- **Approval:** use this attribute for objects subject to an approval process. See [Approvals](#) on page 200 for more details.
- **Survey:** objects with this attribute enable a **Questions Library**, allowing you to create any number of questions that can then be attached to any record to form a questionnaire. You can rank each question's answer to assign an overall score to survey responses. See [Surveys and Quizzes](#) on page 217.
- **Survey Taker:** objects with this attribute contain fields to store responses to surveys.
- **Queue:** use this attribute to mark objects to form a queue. Users can pick up records from queue for further processing. See [Record Queues](#) on page 203.
- **Multi-Currency:** provides support for converting money amounts to and from different currencies. See [Multi-Currency Support](#) on page 214

# Business Logic and Customizing the User Experience

Raw data is often not very useful. Applications that meet real user needs will likely contain logic that retrieves, manipulates, and transforms data. Rollbase offers a variety of built-in mechanisms that simplify the addition of business logic. These include templates, formulas, workflows, and triggers. Since each application is unique, this documentation cannot provide details on how to use these features in every circumstance. Progress recommends that you acquaint yourself with Rollbase capabilities and the examples provided and then apply the principles to your use case.

See [Adding Business Logic](#) on page 151 for descriptions of business features. Watch a short video that demonstrates use of a few basic features: [An overview of application customization](#).

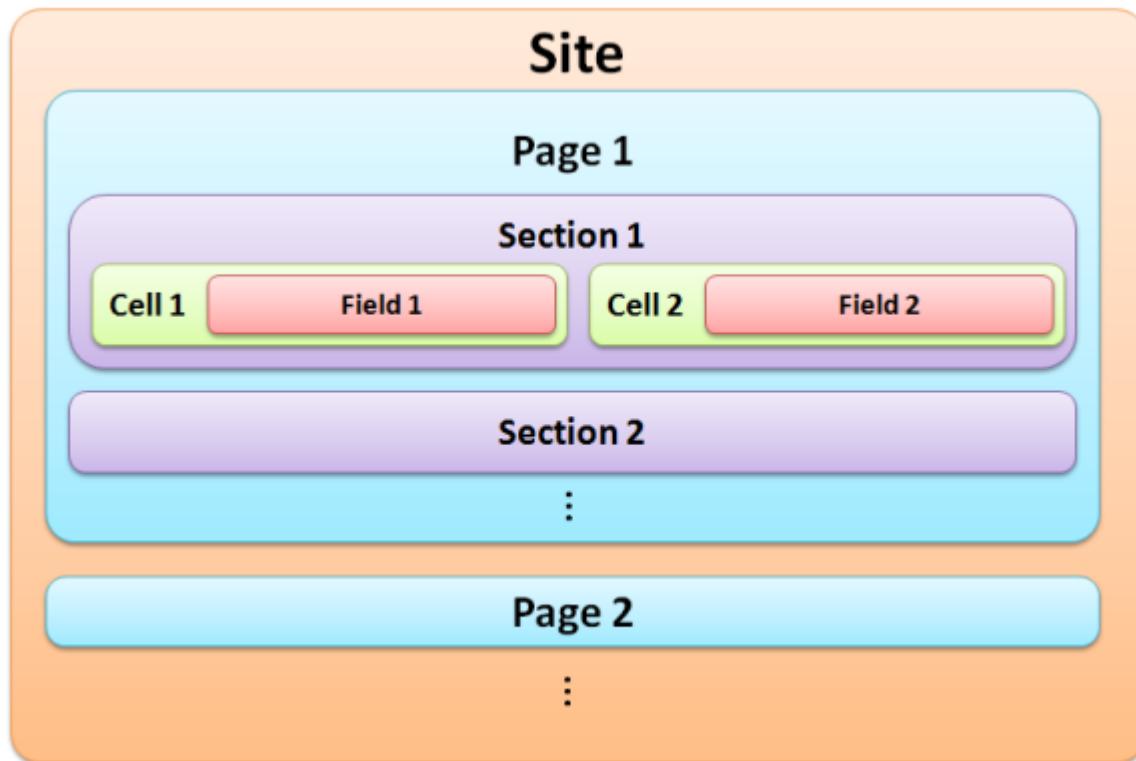
Your end-user experience is shaped both by business logic and by presentation. When you create application objects, Rollbase creates a basic tabbed interface for creating, updating, and deleting records. You can edit these pages to customize what they contain and their look and feel. The remaining topics in this section provide an overview of Rollbase User Interface components.

## Rollbase UI Components

The following table introduces hierarchy of Rollbase user interface components:

Component	Description
Site	A site is the topmost container for Rollbase pages. Rollbase applications and portals both represent different types of sites. (This term is not used in the actual product, but is introduced here to help explain the Rollbase UI framework.)
Page	A page represents a single UI page within a Rollbase application or portal. Pages can be formatted in one or two columns. Your Rollbase application can have several different versions of each page. Users and roles can be assigned to particular versions. You can also control the UI based on user and role. For more details, see <a href="#">Security and Access Control</a> on page 375.
Section	A section is a container within a page with an optional title and border. Sections can be organized into one, two or three-column layouts.
Page Tabs	Page tabs are an optional part of a view page that you can enable to organize multiple sections and reduce vertical scrolling.
Cell	A cell is a container within a section that holds a single component. The component represents data being displayed or a control used for editing or interacting with data, such as a list view, grid control, or object field.
Field	A field is the lowest level UI element on a page, wrapped in a cell to render its contents.

The following diagram illustrates the container hierarchy of Rollbase UI components:



## Application Page Types

The default application pages created by Rollbase provide an interface for users to view, create, update, and delete records. You can modify and clone these default application pages. You can create portal pages and generic pages. You control the visibility of pages by assigning permissions by role and/or by user. End users navigate through application pages using tabs and menus. See [Application Tabs and Menus](#) on page 92 for more information.

When you create an object definition, Rollbase automatically creates the following pages for you: records list, new, view, edit, status change, mass update, and selector list. If you later modify the underlying object definition, for example, by adding a field or attribute, you can choose which pages Rollbase will add the new component to. [Pages, the Page Editor, and Grid Controls](#) on page 223 describes how you can modify pages by adding and removing components.

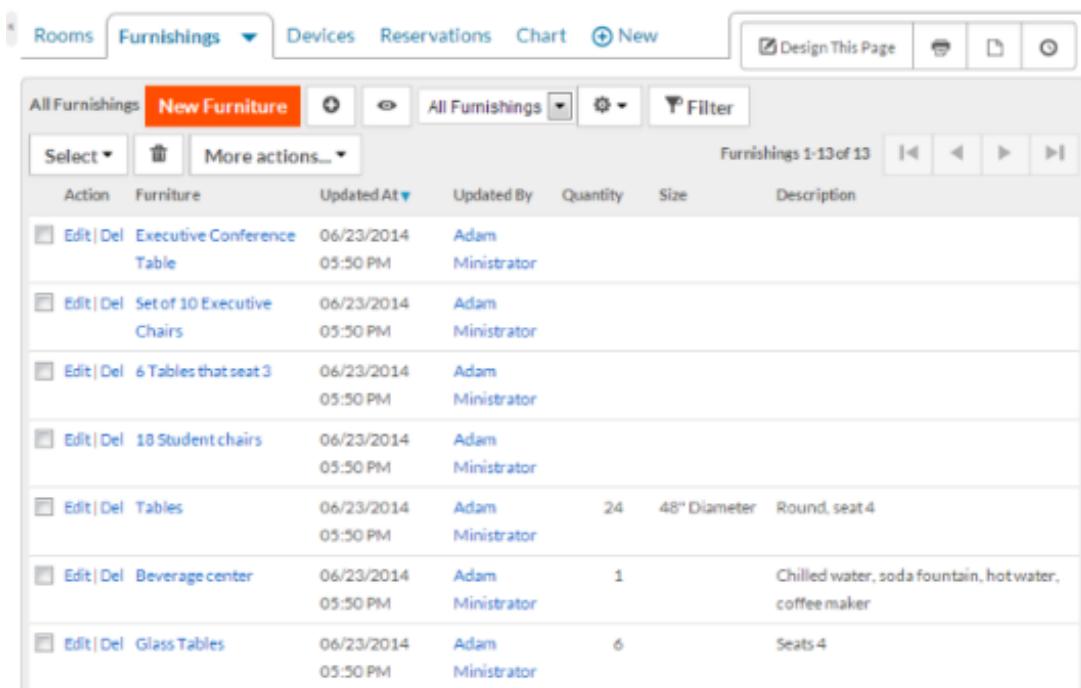
The following table describes each page type:

Type	Purpose
Generic	A generic page is not associated with a particular object. When you create a new generic tab, Rollbase creates a generic page for it. You can use a generic page to render dashboards and other information such as HTML and script components, and to calculate summary information using formula and template fields.

Type	Purpose
Records List	A records list page, associated with the object tab, contains a view component for all records of that object type.
New record	A new record page provides the controls for creating a new record. To open this page when viewing a list of object records, click <b>New &lt;object_name&gt;</b> . See an <a href="#">example of a new record page</a> .
View	A view record page displays an existing record. Open a view page when viewing a list of object records by clicking the record name.
Edit	An edit record page provides the controls for editing an existing record. Open an edit page from a list of records by clicking <b>Edit</b> . See an <a href="#">example of a selector list page</a> .
Status Change	A status change page is used to confirm changes to an existing record of a particular object type. You can optionally include other fields in this page to be updated. Different versions of this page can be associated with different statuses.
Mass Update	A mass update page provides the controls for updating a group of selected records. The default mass update page runs the <code>onUpdate</code> trigger. You can create different versions of this page to provide different mass update functionality. View a mass update page from an object tab by selecting one or more records and selecting <b>Mass Updates</b> from the <b>More Actions</b> menu.
Selector List	A selector list page displays in a pop-up window and allows users to select one or more records for a lookup field. You can open a selector list when creating, viewing, or editing a record, by clicking the lookup icon. See an <a href="#">example of a selector list page</a> .
Portal Page	A <b>Portal</b> page gives portal users access to object records or functionality. You can assign the same portal page to multiple portals. For more information about portals, see <a href="#">Rollbase Portals</a> on page 257

## Records List Page Example

This records list page example from a corporate room reservation system shows the controls available to add or edit furniture records.



The screenshot shows a web-based application interface for managing furniture records. At the top, there is a navigation bar with tabs: Rooms, Furnishings (which is currently selected and highlighted in blue), Devices, Reservations, Chart, and a New button. To the right of the tabs are several icons: a gear for design, a magnifying glass for search, a document, a trash can, and a refresh symbol. A checkbox labeled "Design This Page" is also present.

Below the navigation bar is a toolbar with buttons for "All Furnishings" (which is also blue), "New Furniture" (highlighted in red), and other actions like "Select", "More actions...", and "Filter".

The main content area is a table listing furniture records. The columns are: Action, Furniture, Updated At, Updated By, Quantity, Size, and Description. The table contains the following data:

Action	Furniture	Updated At	Updated By	Quantity	Size	Description
<a href="#">Edit</a>   <a href="#">Del</a>	Executive Conference Table	06/23/2014 05:50 PM	Adam Ministrator			
<a href="#">Edit</a>   <a href="#">Del</a>	Set of 10 Executive Chairs	06/23/2014 05:50 PM	Adam Ministrator			
<a href="#">Edit</a>   <a href="#">Del</a>	6 Tables that seat 3	06/23/2014 05:50 PM	Adam Ministrator			
<a href="#">Edit</a>   <a href="#">Del</a>	10 Student chairs	06/23/2014 05:50 PM	Adam Ministrator			
<a href="#">Edit</a>   <a href="#">Del</a>	Tables	06/23/2014 05:50 PM	Adam Ministrator	24	48" Diameter	Round, seat 4
<a href="#">Edit</a>   <a href="#">Del</a>	Beverage center	06/23/2014 05:50 PM	Adam Ministrator	1		Chilled water, soda fountain, hot water, coffee maker
<a href="#">Edit</a>   <a href="#">Del</a>	Glass Tables	06/23/2014 05:50 PM	Adam Ministrator	6		Seats 4

At the bottom right of the table, it says "Furnishings 1-13 of 13" and there are navigation arrows for the list.

## New Record Page Example

This example new record page from a corporate room reservation system has controls for adding related furnishings.

New Room

Room Information

Room:  Reservation:

Phone Number:

Furnishings: **Add Furniture**

Quantity	Furniture	Description	Size

Location and Address Information

Street Address 1:  State/Province:

Street Address 2:  ZIP/Postal Code:

City:  Country:

Save  Cancel

## View Page Example

This example view page from a corporate room reservation system shows record data. The **System Info** sub-tab contains information from system fields such as who created the record and when it was last updated.

Furniture: Bistro Chairs

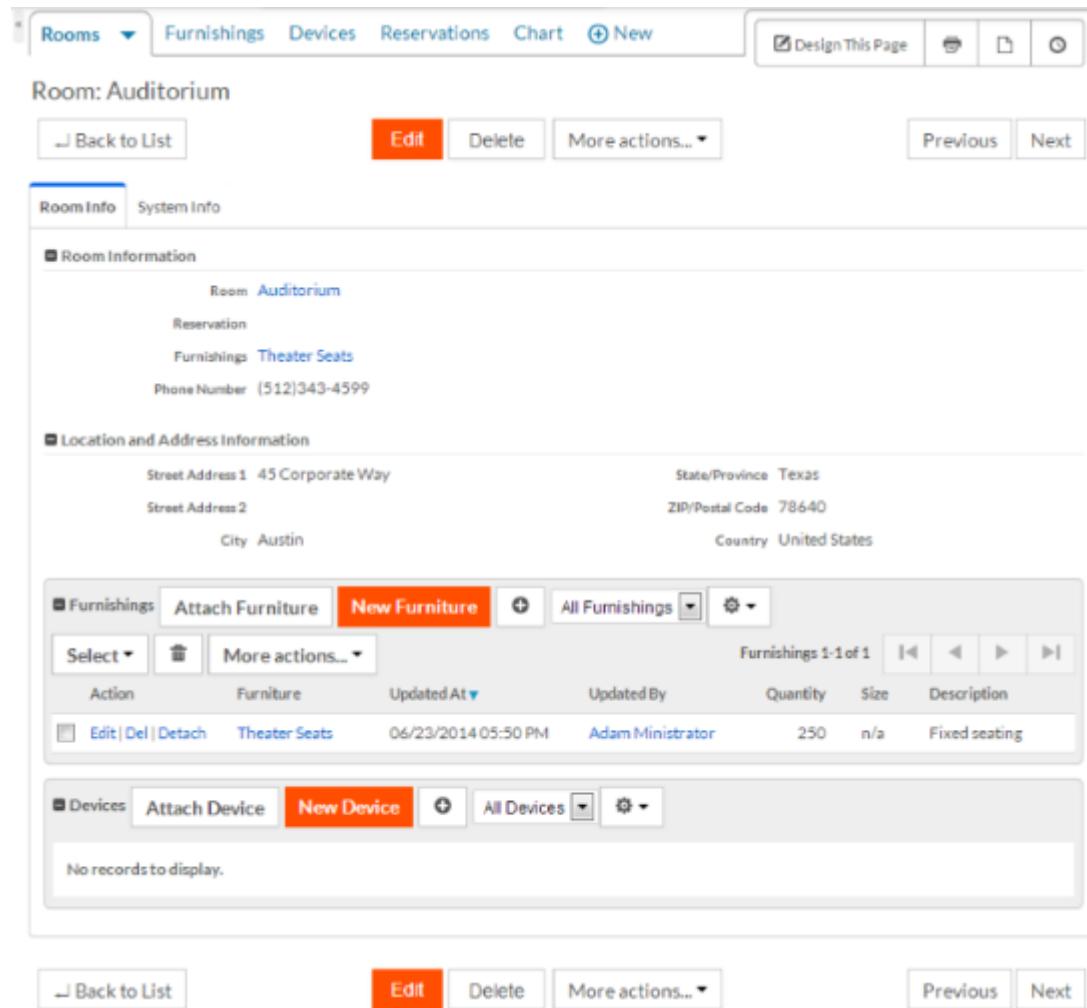
Back to List   More actions...

Furniture Info	System Info
<b>Furniture Information</b> Furniture: <a href="#">Bistro Chairs</a> Room: <a href="#">Atrium Social Corner</a> Quantity: 24 Size: Plastic molded Description: Plastic molded	

Back to List   More actions...

## Edit Page Example

This example edit page from a corporate room reservation system allows users to edit the record and add or edit the related furnishings and devices.



The screenshot shows an edit page for a room named 'Auditorium'. The top navigation bar includes 'Rooms' (selected), 'Furnishings', 'Devices', 'Reservations', 'Chart', and a '+ New' button. To the right are buttons for 'Design This Page' and other page controls. The main content area is titled 'Room: Auditorium' and includes a 'Back to List' link, an 'Edit' button (highlighted in red), a 'Delete' button, and a 'More actions...' dropdown. Below this are tabs for 'Room Info' (selected) and 'System Info'. The 'Room Information' section shows the room name 'Auditorium' and a 'Reservation' link. The 'Location and Address Information' section shows the address '45 Corporate Way, Austin, TX 78640' and the country 'United States'. Below these are two selector lists: 'Furnishings' and 'Devices'. The 'Furnishings' list shows one record: 'Theater Seats' (Action: Edit | Del | Detach, Updated At: 06/23/2014 05:50 PM, Updated By: Adam Ministrator, Quantity: 250, Size: n/a, Description: Fixed seating). The 'Devices' list shows a message: 'No records to display.' The bottom of the page includes a 'Back to List' link, an 'Edit' button (highlighted in red), a 'Delete' button, and a 'More actions...' dropdown, along with 'Previous' and 'Next' links.

## Selector List Page Example

This example shows a selector list page from a corporate room reservation system. Room records are related to Furniture records. To specify which room furniture belongs to, users can click the Room lookup field to see the list of rooms.

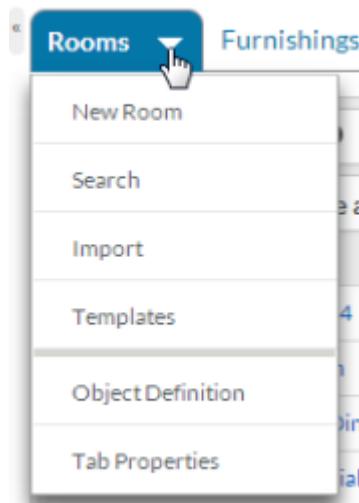
Room	Street Address 1	City	State/Province	Phone Number
Fish Bowl Conference	45 Corporate Way	Austin	Texas	(512)343-6688
Auditorium	45 Corporate Way	Austin	Texas	(512)343-4599
Cafeteria	44 Corporate Way	Austin	Texas	(512)343-5879
Executive Conference	45 Corporate Way	Austin	Texas	(512)343-5600
Atrium Social Corner	45 Corporate Way	Austin	Texas	(512)343-9877
Overflow Dining	45 Corporate Way	Austin	Texas	(512)323-7899
Classroom 4	44 Corporate Way	Austin	Texas	(512)343-5688

## Application Tabs and Menus

Rollbase application tabs include the following types:

- Generic: You can create generic tabs that contain pages with custom content such as arbitrary views, charts, HTML or script components, or 3rd party widgets.
- Object: By default, when you create an object, Rollbase creates an object tab for the records list page, which lists all records of that object type. Controls and tab menu items on this page allow the user to navigate to other pages for the same object, such as the new and view pages. See [Application Page Types](#) on page 87 for more information about the pages Rollbase creates for objects.
- Web: You can create web tabs that contain an embedded website, allowing you to embed other sites or web-based applications into your Rollbase applications.

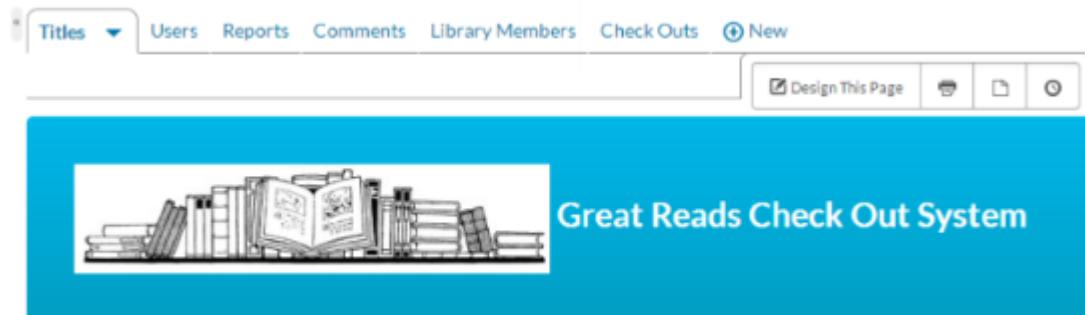
Tabs and their child menus provide navigation through Rollbase application pages. Tabs are at the top level of visibility for the application. Menu items are available from the tab drop-down arrow. For example, the following **Rooms** tab drop-down menu contains items for creating a new record, searching room records, importing room records, adding templates, viewing the object definition, and viewing tab properties.



The **Tab Properties** provide controls for adding and reordering menus and setting permissions that determine which menu items will be visible to users. You can organize navigation by creating new menus, removing menu items, or changing tabs to be child menus of other tabs (tabs that are assigned a parent become menus of the parent).

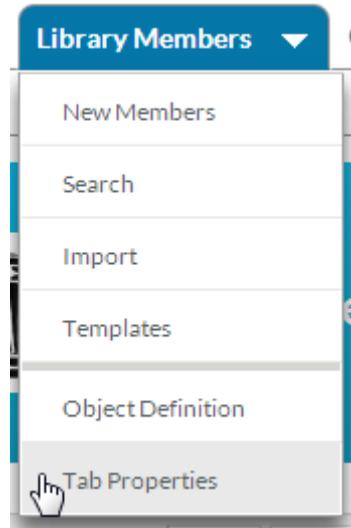
### Changing a Tab to be a Child Menu

The following screens show how to change a top-level tab to be a child menu. This example uses a Library application that includes six tabs, one for each application object:



To change a tab to a menu:

1. From the tab's menu, select **Tab Properties**:



2. Click **Edit**.
3. From the **Parent Tab** drop-down, select the tab on which you want this tab to appear as a menu:

Personal Setup Applications Setup Administration Setup

Application Setup > Tabs > Library Members > Edit

Tab: Library Members

Save Cancel

Define Tab Properties

Tab Name	Library Members
Tab Type	Object
Parent Tab	None (top level tab)
Object	Object Tabs allow record navigation between objects. A screenshot shows a navigation bar with tabs like Home, Calendar, Meetings, etc., and a main content area with a green overlay.
Description	
Description	Users

None (top level tab) ▾

- Calendar
- Chart
- Check Outs
- Comments
- Departments
- Devices
- Functions
- Furnishings
- Gmail
- Groups
- Home
- Locations
- Meetings
- Reports
- Reservations
- Rooms
- Titles
- To-Dos
- Users

a particular object definition.

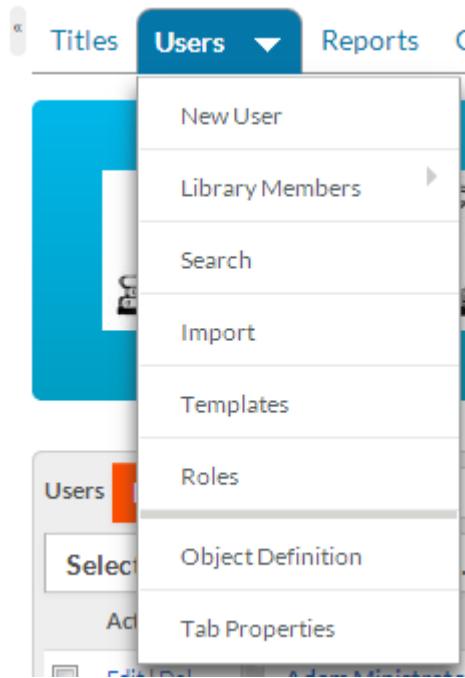
Name

Select View

row titles.

#### 4. Click OK.

With the **Library Members** tab set to have the **Users** tab as its parent, it no longer appears as a tab. Instead, Library member records and the pages to modify them are available from the menu on the **Users** tab:



**Note:** After a tab has been demoted to be a child menu item, if you want to edit it, its settings are only available from the parent tab definition. Go to the application setup page, select the parent tab, and select the child from the **Menus** section.

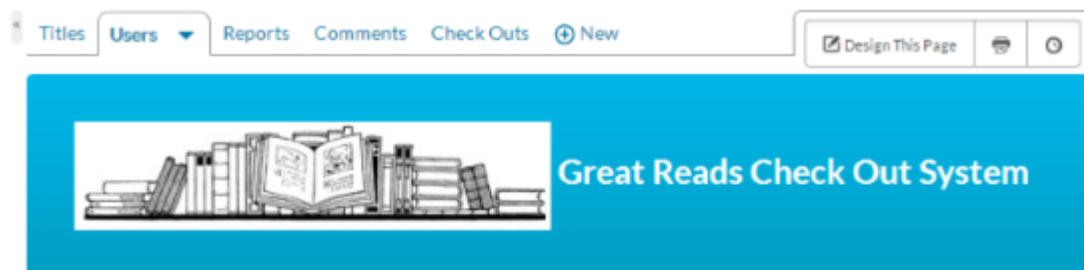
### Promoting a Child Menu Tab to be a Top-Level Tab

If you have demoted a top-level tab to be a child menu of another tab, follow these steps to make it a top-level tab again:

1. Navigate to the setup page for the parent tab.
2. In the **Menus** section, click **Edit** in the child tab's row.
3. From the **Parent Tab** menu, select **None**.
4. Navigate to application setup and add the newly promoted tab back to the appropriate application(s).

## Tabs on Pages

You can organize sections on record view, edit, and new record pages using tabs. This helps to visually separate components on a complex page. For example, the following screen shows the new page for **User** records. It contains multiple sections, each with multiple fields. A user would have to scroll to enter the data for all fields:



## New User

**Save** **Save & New** **Cancel**

## Contact Information

First Name	<input type="text"/>	Email Address	<input type="text"/>
Middle Name	<input type="text"/>	Phone	<input type="text"/>
Last Name	<input type="text"/>	Mobile Phone	<input type="text"/>
Job Title	<input type="text"/>	Fax	<input type="text"/>

## Login and Role

Login Name	<input type="text"/>	Approver <input type="checkbox"/>
User Role	<input type="text"/> Administrator	<input type="button" value="▼"/>

## Reporting Structure

Reports To	<input type="text"/> <input type="button" value="Search"/>	Direct Reports	<input type="text"/> <input type="button" value="Search"/>
------------	--	----------------	--

## User Preferences

Rows Per Page	<input type="text"/>	Do Not Animate Collapse <input type="checkbox"/>
---------------	----------------------	--

This example shows the same page, but with page tabs that organize the sections.

## New User

**Save** **Save & New** **Cancel**

Contact

User Preferences and Settings

## User Preferences

Rows Per Page	<input type="text"/>	Do Not Animate Collapse <input type="checkbox"/>
---------------	----------------------	--

## User Settings

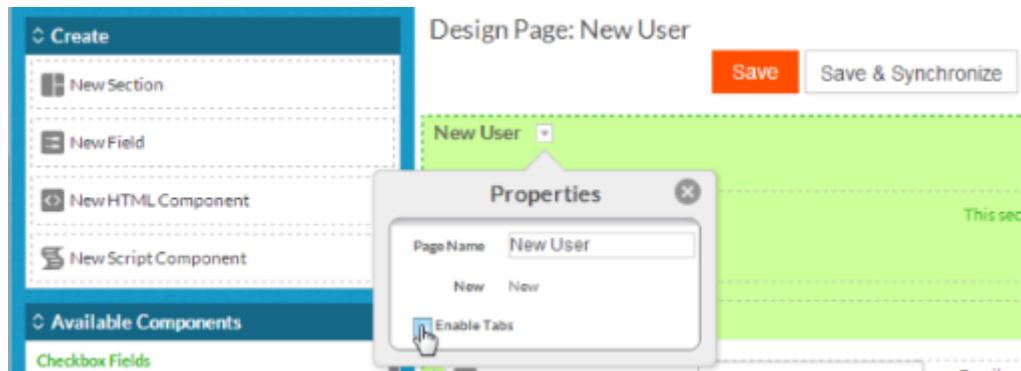
Language	<input type="text"/> English
Date Format	<input type="text"/> 06/19/2014 10:25 AM
Time Zone	<input type="text"/> -- Please select --
Email Footer	<input type="text"/>
Email Encoding	<input type="text"/> ISO-8859-1

**Save** **Save & New** **Cancel**

Tabs on view pages are displayed on demand when the user selects them. The content of these tabs is generated and sent via AJAX as needed. This may present a problem if tabs include JavaScript. In this case, you can check the **Do not use AJAX loading for Tabs** on the page's **Properties** page. On pages that allow users to browse records, the selected tab continues in focus as the user clicks **Next** or **Prev**. The tab selection is reset to the first tab if a user goes back to the list of records.

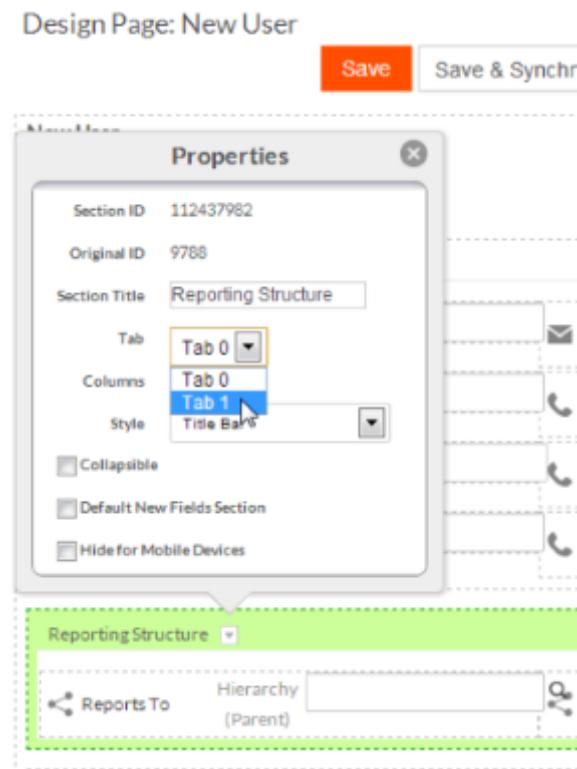
To enable tabs, follow these steps:

1. Navigate to the page on which you want to enable tabs, such as a new page.
2. Click **Design This Page**.
3. Click the menu next to the **New <Object\_Name>** section title and select **Enable Tabs**:



The first tab, named **Tab 0**, and a button to add tabs will display.

4. Click **Add Tab** to add as many tabs as you like.
5. Associate each section with a tab (the first section is automatically associated with tab 0):
  - a. Click the menu next to the section title.
  - b. In the **Properties** pop-up, select a tab from the **Tab** menu:



- When you are finished, click **Save** to apply the changes to this page or click **Save and Synchronize** to apply the changes to additional pages.

## Page Cells

The following table lists the main cell components. Rollbase adds some to pages when they are created; you can add others by dragging them onto a page from the list of available components in the **Page Editor**.

Many cells have page-level properties that can be changed in the **Page Editor**. For details about using the Page Editor, see [Editing Pages](#) on page 224.

Cell Component	Use	Where Available
Form	Contains buttons to edit, cancel or submit a form.	Added automatically to view, <b>New &lt;object&gt;</b> , edit, status change, and mass update pages. Cannot be deleted.
Chart	Renders the selected chart.	Generic and list - if at least one chart is created for this object.
Gauge	Renders the selected gauge.	Generic and list - if at least one gauge is created for this object.
Comments Table	Renders a list of comment records created on a particular record.	View page

Cell Component	Use	Where Available
Audit Trail	Renders a list of audit trail records created on particular record.	View page
View	List of records which can be paged, sorted, filtered, grouped, etc.	Generic, list, and selector - only one view component per object type can be placed on a page.
View of Related Records	List of records related to or having a relationship with the record currently being viewed.	View page
Recurrent Events	List of recurrent instances of event or task	View page
Detailed Search	A configurable search component providing a way to allow users to do field-specific searches (for instance, Amount>Min AND Amount<Max).	Generic and list
Field	Renders an object field in view or edit mode. View mode (in certain cases) allows inline editing using the icon. Many field types have properties that can be configured at the page level.	View, <b>New &lt;object&gt;</b> , edit, status change, and mass update pages. When new fields are created each field is added to the <b>Default New Fields Section</b> of each page.
Grid Control	A configurable component used in edit pages to create and edit a group of records related to the current (master or parent) record. For more details, see <a href="#">Using Grid Controls to Manage Multiple Records</a> on page 227.	<b>New &lt;object&gt;</b> , edit, view, and status change - No more than one grid control can be used on any given view page.
Organization Tree	Displays an interactive hierarchy of locations, departments, or functions.	List (for LDF objects only)
Report Link	Link to a particular report	Generic, list, and view
Template HTML	Template-based HTML component	All pages
Script Component	Template-based script component	All pages

---

## Laying the Foundation

---

Objects, their fields, and the relationships between them form the data model for a Rollbase application. From these, Rollbase generates a set of pages for viewing, creating, editing, and deleting records. The topics in this section describe how to work with applications, objects, and records.

The following videos provide a quick orientation to Rollbase:

- [Finding your way around in Rollbase](#)
- [The Quick Create Wizard](#)
- [An overview of application customization](#)
- [An overview of how to distribute your applications](#)

The topics in this section describe how to create and manage applications and objects.

For details, see the following topics:

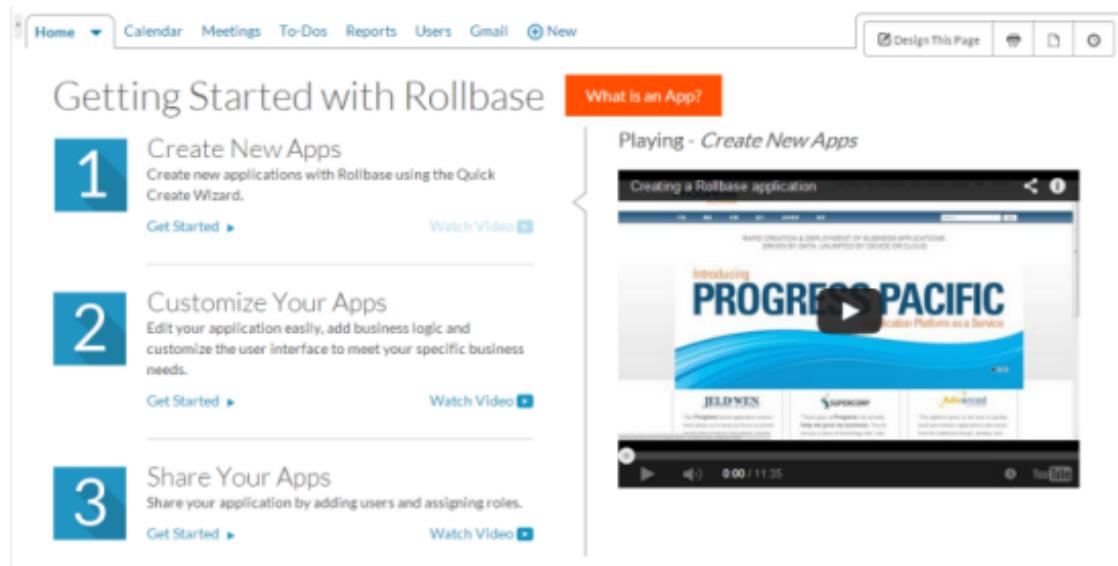
- [Getting Started with the Quick Create Wizard](#)
- [Creating and Managing Applications](#)
- [Creating and Managing Objects, Fields, and Relationships](#)
- [Views](#)
- [Relationships Between Objects](#)
- [Working with Records](#)

# Getting Started with the Quick Create Wizard

The **Quick Create** wizard walks you through the steps required to create a Web application and add objects and relationships to it. Follow these steps to use the **Quick Create Wizard**:

1. In the left navigation pane, click **Getting Started**.

The **Getting Started** page displays:



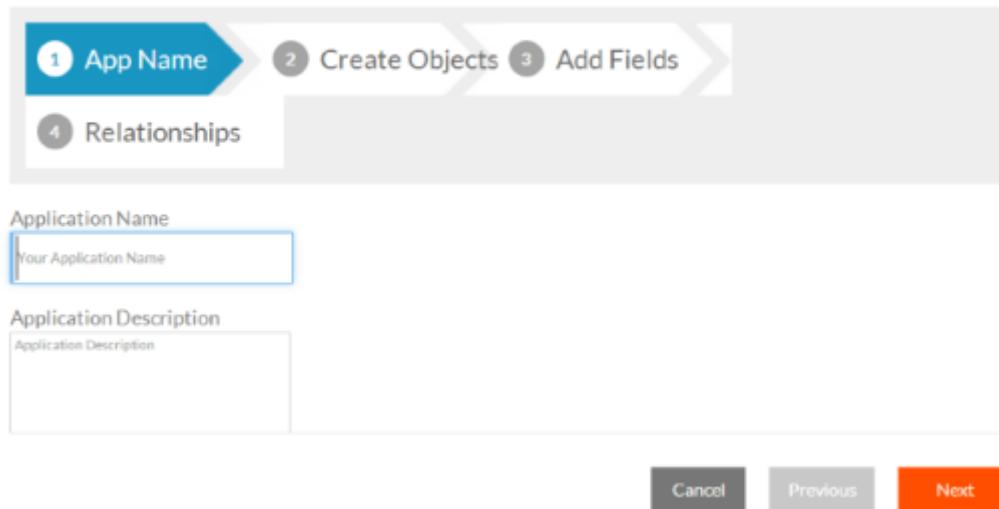
The screenshot shows the 'Getting Started with Rollbase' page. At the top, there is a navigation bar with links for Home, Calendar, Meetings, To-Dos, Reports, Users, Gmail, and a New button. To the right of the navigation bar are buttons for Design This Page, Print, Copy, and Refresh. The main content area is titled 'Getting Started with Rollbase'. It features three numbered steps: 1. Create New Apps, 2. Customize Your Apps, and 3. Share Your Apps. Each step has a large blue number, a title, a brief description, a 'Get Started' link, and a 'Watch Video' link. To the right of the steps, there is a video player window titled 'Playing - Create New Apps' showing a video titled 'Creating a Rollbase application'. The video player has a play button, a progress bar showing 0:00 / 11:35, and other video controls.

**Note:** If the **Getting Started** page does not display, you might need to update your Rollbase application.

2. Click the **Get Started** link under number one.

The **Quick Create** wizard displays:

## Quick Create Application



3. Enter a name and optionally, a description for your application..
4. Click **Next**.

The **Create Objects** page displays.

5. Follow these steps to add object definitions:

- In the **Object Name** field, enter the singular form of your object name. Rollbase will use the plural form for lists of objects.
- From the **Attributes** drop-down list, select desired attributes. See [Object Attributes](#) on page 84 for more information about attributes.
- To add more objects, click **Add Object**
- When you are finished, click **Next**.

The **Add Fields** page displays, with the first object highlighted.

6. To add fields, follow these steps:

- Enter a singular name in the **Field Name** box.
- From the **Field Type** drop-down list, select a data type. For a description of field data types, see [Field Types](#) on page 832.
- Optionally, enter a **Default Value** that will populate this field when users create new records.
- To add another field, click **Add Field**.
- To add fields to another object, select that object in the left pane.
- When you are finished, click **Next**.

The **Relationships** page displays.

7. In the **Related To** field of the object to which you want to add a relationship, click **Select**. Choose the appropriate relationships as follows:
  - a) Check the info graphic to visualize relationship cardinalities.

- b) If the relationship cardinality icon has arrows (  ) click them to display all of the available cardinalities.

- c) Click the plus icon to select a relationship.
- d) When you are finished with relationships, click **Submit**.

An overview of the application displays. You can see the pages Rollbase created for you and you can view and edit your application.

## Creating and Managing Applications

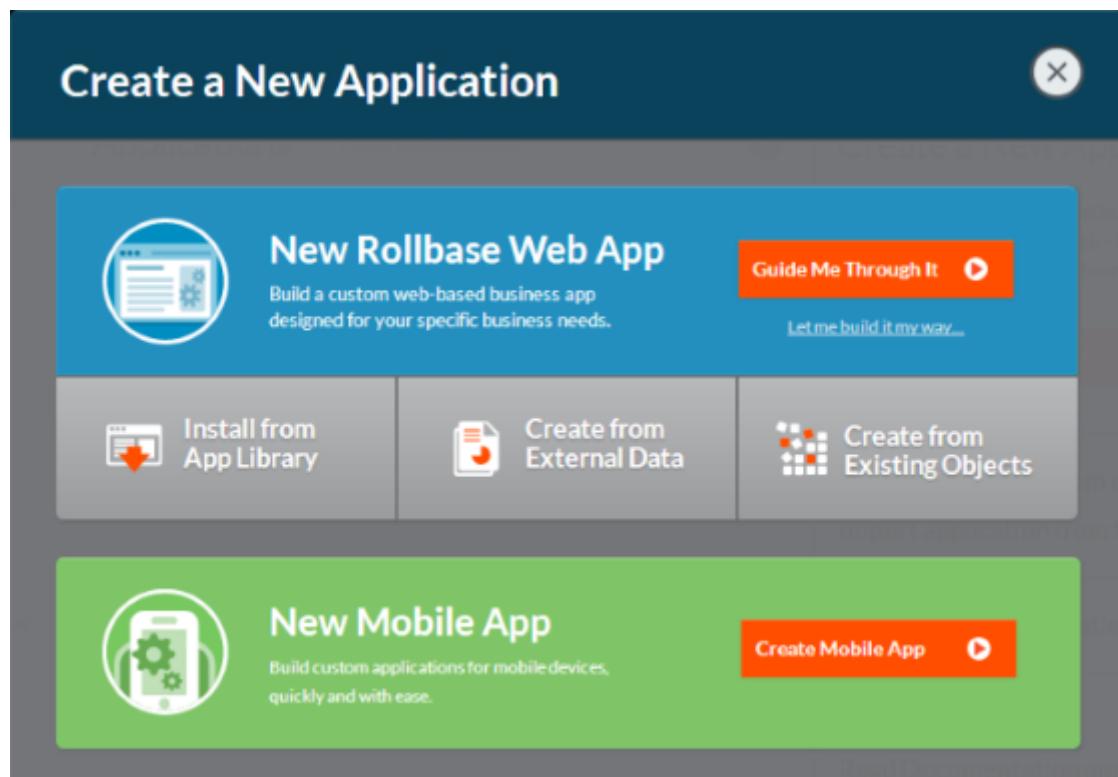
You can quickly create a new Rollbase application from scratch, install an application that has been distributed to you in XML format, create an application that includes existing objects, or create an application by importing from external data sources. The topics in this section describe the options available to create and manage applications and objects.

### Creating an Application

Follow these steps to create a new Rollbase application:

1. In the left navigation pane, click the **New Application** button.

The **Create Application** dialog displays.



2. Choose the appropriate option for creating an application:

- **New Rollbase Web App** — Two options: click **Guide Me Through It** to launch the **Quick Create Application** wizard, or click **Let me build it my way** to create a new empty application to which you will need to add components.

- **Install from App Library** — Opens the **Application Directory**, which contains a list of applications you can install.
- **Create from External Data** — Creates an application by importing from: Rollbase application XML, Force.com, MS Access, an OpenEdge JSDO catalog or a ZIP archive of a Rollbase Mobile application. See the following for more information:
  - [Generating Application XML](#) on page 413
  - [Creating Rollbase Applications from Salesforce Applications](#) on page 350
  - [Creating Rollbase Applications from Microsoft Access](#) on page 345
  - [Creating an Application from OpenEdge Data](#) on page 326
- **Create from Existing Objects** — Creates a new application and allows you to add existing objects to it.
- **New Mobile App** — Displays the **Create New Mobile App** screen:

After naming the app and choosing the type, choose one of:

1. **From Scratch.** This option creates a mobile app that is completely independent from any Rollbase Web App. If you want the mobile app to use Rollbase application data, do not use this option.
2. **Based on Existing App** allows you to select a Rollbase application. The underlying framework for accessing Rollbase application views will be automatically generated for you.

See [Simple Rollbase Mobile App Example](#) on page 281 for more information on creating a mobile app. See <http://docs.mobile.rollbase.com/> for help using the Rollbase Mobile App Builder.

The remaining steps vary depending on the option you choose.

## Editing Applications

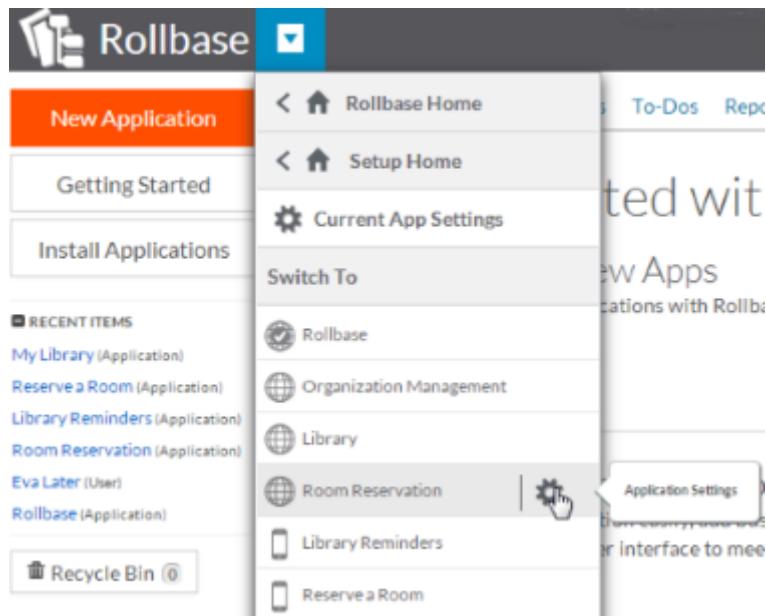
You can view and edit the following from **Setup**:

- The application definition, the attributes and properties set when the application was created.
- Application component definitions.

To view or edit application or component definitions, follow these steps:

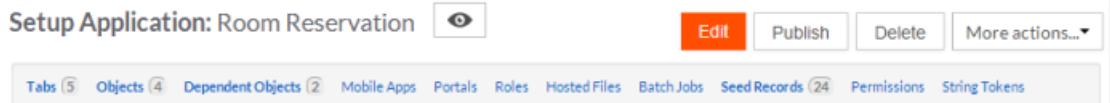
1. From the header menu:
  - To navigate to settings for the current application, click **Current App Settings**.

- To navigate to settings for a different application, select the application and click its **Application Settings** link.



2. Perform any of the following operations from the application details page:

- Edit the application components (such as Tabs, Objects) listed in the ribbon below the application name:



Click the component link to navigate to the component area

- Delete** an application.
- Access the **More Actions** menu to perform advanced operations. See [Application Actions](#) for the different options available in this menu.

3. Click the **Edit** button next to the application name to edit any application properties. These attributes and properties include:

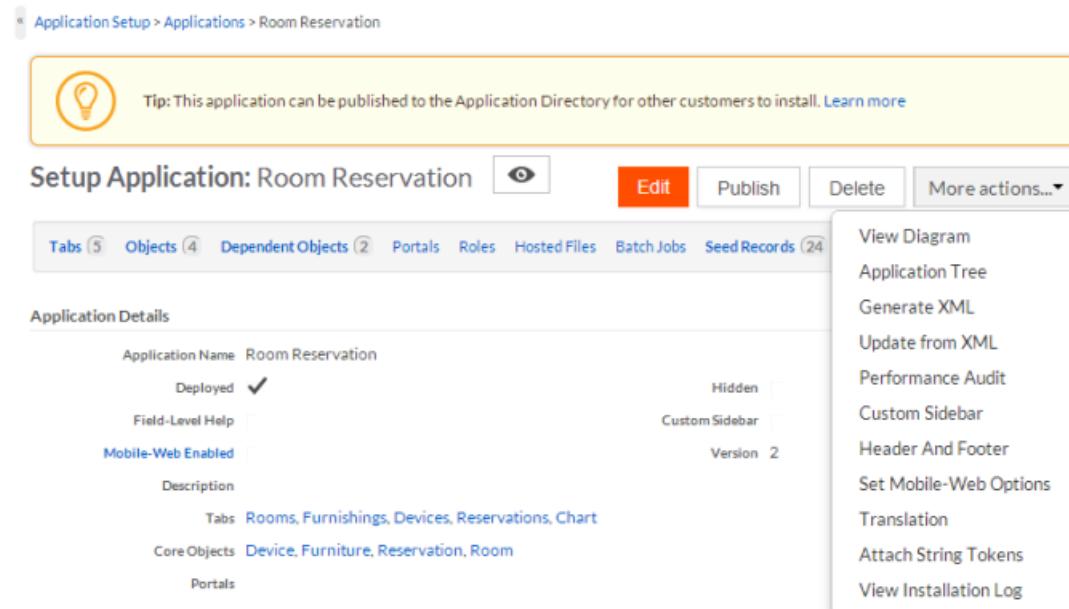
- Deployment status:** Specifies:
  - Whether an application is deployed. If deployed, all users with permissions to view the application will see it when they log in. If not deployed, only users with permissions who have an administrator role will see it
  - Whether an application is hidden. If hidden, it will not be listed in the application drop-down menu
  - Whether any field-level help you have supplied in component definitions will be available to end-users who click ?
- Application name and Description**

- An optional **Custom Logo** to appear in the upper-left corner of each page of the application. The maximum size is 256 KB. Additionally, you can choose to hide an application logo by selecting the **Hide Logo** check box.
- **Tabs, Core objects, Dependent Objects, and User Roles** that must be included in the application when it is published. The lists specify the order in which they must appear.

## Application Actions

The **More Actions** menu on the application view page offers additional options to manage your application. To access **More Actions**:

1. Navigate to the setup application list: **Setup Home > Applications**
2. Click the application name.
3. Select the **More Actions** drop-down menu.



The screenshot shows the 'Setup Application: Room Reservation' page. At the top, there is a tip message: 'Tip: This application can be published to the Application Directory for other customers to install. [Learn more](#)'. Below the tip is a 'More actions...' dropdown menu with the following options: View Diagram, Application Tree, Generate XML, Update from XML, Performance Audit, Custom Sidebar, Header And Footer, Set Mobile-Web Options, Translation, Attach String Tokens, and View Installation Log. The main application details are listed, including Application Name (Room Reservation), Deployment status (Deployed), and other settings like Field-Level Help, Mobile-Web Enabled, and Version 2.

Application actions include the following:

### View Diagram

The **More Actions** menu on the application view page includes a **View Diagram** option. This option generates a Entity-Relationship (ER) diagram that helps you visualize application objects and relationships. The diagram is useful during development and to get an overview of applications you did not develop but installed from XML. The EER diagram is generated by a free service called [yuml.me](http://www.yuml.me) (<http://www.yuml.me>) and for large applications, can take a few minutes to load.

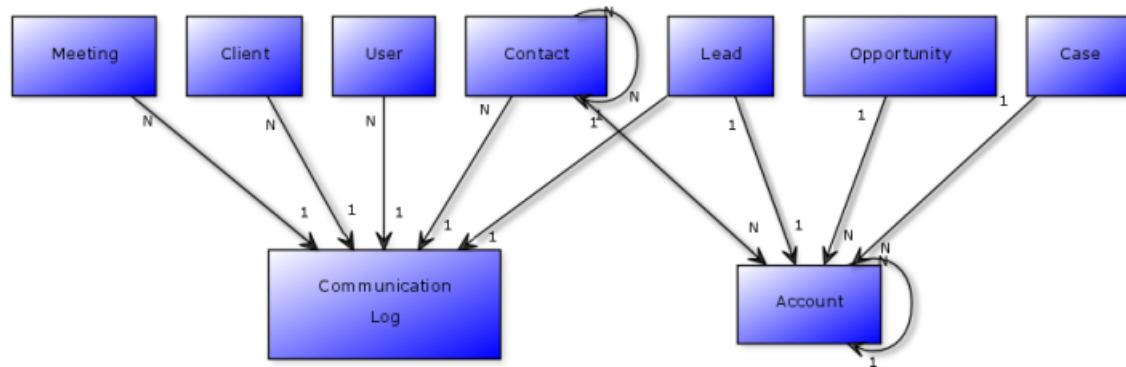
The following ER diagram shows the objects in the CRM application, which is available from the **Applications Directory**:

## Progress | Rollbase



## CRM: Entity-Relationship Diagram

**Tip:** This diagram visually represents objects included (explicitly or implicitly) in the "CRM" application, and relationships between them. Please wait while the UML generator loads this image. [Read more](#)



Copyright (c) 1993-2013 Progress Software Corporation. All rights reserved.

## Application Tree

The **More Actions** menu on the application view page includes an **Application Tree** option. This option allows you to view all application components in a tree. Components with errors will be highlighted in red. Expand parent nodes to see child components.



## Library: Application Tree

Tip: Application tree visually represents all elements included in your application. It also displays all errors and discrepancies in your application. If you have any errors fix them before publishing this application. [Learn more](#)

- Library
- Dependent Objects
- Communication Log
- Objects
- Pages
- Menus
- Portals
- Hosted Files
- Seed Records

Copyright (c) 1993-2014 Progress Software Corporation. All rights reserved.

## Generate XML

The **More Actions** menu on the application view page includes a **Generate XML** option. The **Generate XML** option allows you to generate and save an application in XML format. The resulting XML can be installed in other Rollbase tenants. The XML files for complex applications can be over 5MB. After the file is generated choose **Save** or **Save As** to save a copy locally.

Applications with errors will not generate. If this is the case, try creating an [application tree](#), which will show you any components that have errors in them.

Options available when generating XML include the following:

- Version of application to be exported as XML. This value is auto-incremented each time you generate new XML.
- Use the "Lock Status" radio buttons to choose whether Users in destination tenants can:
  - Modify all parts of the installed application (choose Unlocked)
  - Modify only unlocked Objects, Menus and Portals (Partially Locked)
  - Modify nothing; the entire cannot be customized by the tenant (Locked)

See [Publishing and Distributing Applications](#) on page 407 for more details.

Note:

Application: Sample App

Generate XML

Sample App: Generate XML

Red = Required Information

Application Name: Sample App

Version: 5

Current version: 4

Lock Status:  Unlocked (users can modify all application's components)  
 Partially Locked (users can modify selected application's components)  
 Fully Locked (users cannot modify any application's components)

Sample App | Select All | Select None

Dependent Objects

USER

Objects

Child

Fields

Relationships

## Update from XML

The **More Actions** menu on the application view page includes an **Update from XML** option. Use this option to upload Rollbase Application XML file and install updates. For information on installing application updates, see [Installing Application Updates](#) on page 556.

## Performance Audit

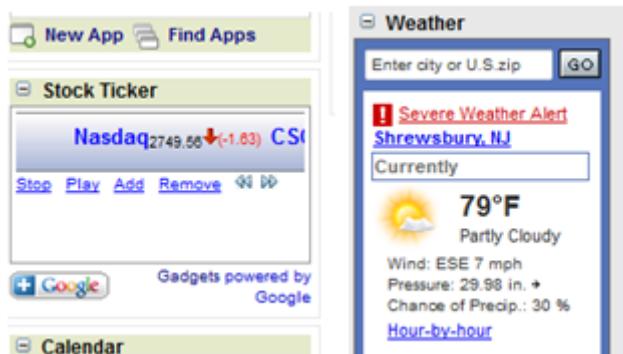
The **More Actions** menu on the application view page includes an **Performance Audit** option. Use this option to:

- Validate all formulas used by your objects and display any errors.
- Check whether formulas are using loops through related records. Loops are inefficient and should be replaced with Query API methods whenever possible.
- Check whether views are using template and formula fields that can decrease performance.

## Custom Sidebar

The **More Actions** menu on the application setup page includes an option to add a **Custom Sidebar**. Custom sidebars appear in the left sidebar. (see [Navigating the Rollbase Environment](#) on page 31) and will be included in the published application. You can add your own component using HTML or Javascript. You can also use web links to widgets or gadgets from providers such as Google and Yahoo.

The screen shots below show a Google Stock Ticker gadget and weather gadget added to the sidebar.



## Header and Footer

The **More Actions** menu on the application view page includes an option to add a custom header and footer. The HTML, Javascript, or template tokens you add here will be rendered on all application pages. You can also include helper template tokens that are not related to any particular object. If the code references CSS or graphic files, you must upload them as hosted files, see [Hosted Files](#) on page 267 for more information.

For example, using an uploaded image, the following HTML classes and declarations create the header and footer shown in the screen shot below:

```
<!-- In the HTML Header box -->
<style>
.my-header {
  background: #eee;
  margin-bottom: 10px;
  padding: 10px;
  margin-bottom: 25px;
  font-size: 25px;
  font-weight: bold;
  padding: 35px;
  border-radius: 5px;
  background: #00b7ea;
  background: -moz-linear-gradient(top, #00b7ea 0%, #009ec3 100%);
  background: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%,#00b7ea), color-stop(100%,#009ec3));
  background: -webkit-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
  background: -o-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
  background: -ms-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
  background: linear-gradient(to bottom, #00b7ea 0%,#009ec3 100%);
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#00b7ea',
  endColorstr='#009ec3',GradientType=0 );
  color: #fff;
}
.my-footer {
  background: #eee;
  margin-bottom: 10px;
  padding: 10px;
  margin-bottom: 25px;
  font-size: 12px;
  font-weight: bold;
  text-align: center;
  padding: 15px;
  border-radius: 5px;
  color: #fff;
  background: #45484d; /* Old browsers */
  background: -moz-linear-gradient(top, #45484d 0%, #000000 100%); /* FF3.6+ */
  background: -webkit-gradient(linear, left top, left bottom,
  color-stop(0%,#45484d), color-stop(100%,#000000)); /* Chrome,Safari4+ */
  background: -webkit-linear-gradient(top, #45484d 0%,#000000 100%); /* Chrome10+,Safari5.1+ */
}
```

```

background: -o-linear-gradient(top, #45484d 0%,#000000 100%); /* Opera
11.10+ */
background: -ms-linear-gradient(top, #45484d 0%,#000000 100%); /* IE10+ */
background: linear-gradient(to bottom, #45484d 0%,#000000 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#45484d',
endColorstr='#000000',GradientType=0 ); /* IE6-9 */
}
</style>
<div class="my-header">
<img src='{!!#HOSTED_FILE.53382#url}' border='0' align='absleft' />
Great Reads Check Out System
</div>

<!-- In the HTML Footer -->
<div class="my-footer">
May your days always have happy endings!
</div>

```

Action	Title	Author(s)	Publisher
Edit   Del	Mall Street Journal	Store Ink	
Edit   Del	The Night the Lights Went Out for Georgie	Aamaras, D	
Edit   Del	Bitnapped	Carroll, J. T.	
Edit   Del	Architecture in the Fifth Century	Digger, R.	Textbooks for Now
Edit   Del	Wonders of the Ancient World	Histor, A	TriplePlay Publishing House

## Set Mobile-Web Options

The **More Actions** menu on the application view page includes an option to **Set Mobile-Web Options**. Use this option if you want Rollbase to generate a Mobile-Web (formerly known as Mobile Edition) version of your application. See [Supporting Mobile Users](#) on page 271 for more information.

## Translation

In tenants configured for additional languages, the **More Actions** menu on the application view page includes an option for **Translation**. You can create an Excel spreadsheet that translates (or localizes) your application into a foreign language. For more information on localization, see [Localization](#) on page 171.

## Attach String Tokens

The **More Actions** menu on the application view page includes an option to **Attach String Tokens**. You can attach selected strings for localizations to your Rollbase application. Attached string tokens will be published and installed as part of the application.

For more information on string tokens, see [Language Support](#) on page 432.

## Application Permissions

The **Permissions** section on the application view page allows you to manage permissions by **User Role** and by individual **User**. Permissions set for a user override those set for roles, allowing fine-grained access control. In addition to permissions set at the application level, you can set relationship-based permissions and role-based field-level permissions.

For information about managing application permissions and access control, see [Security and Access Control](#) on page 375.

## Deleting an Application

Deletion of an application has the following consequences:

- All objects (and corresponding data records) that are assigned to the deleted application and are not assigned to any other application will be permanently deleted.
- All menus, portals, and hosted files that are assigned to this application and are not assigned to any other application will be permanently deleted.
- The application record will be permanently deleted.

Deletion of an application does not cause deletion of System objects.

Rollbase requires the following pre-requisites to avoid accidental deletion of important information:

- Before deleting an application, un-check the **Is Deployed** box.
- Wait for an hour, then delete your application.

## Installing and Updating from the Application Directory

To install an application published to the Rollbase hosted or Private Cloud **Application Directory**: click **Install Applications**, browse the list of available applications, take a test drive if enabled, read reviews and click **Install Now**. After finishing the installation, you can start using the newly installed application.

Check for updates to installed applications by navigating to **Setup > Applications Setup > Applications** and clicking **Check Updates**. If updates have been published and the version of the installed Application is lower than the currently published version, click **Install Updates** to proceed.

## Overriding Changes

When installing updates from the **Application Directory**, note the **Override Changes** checkbox below the **Install Updates** button. For unlocked applications, this checkbox controls important behavior:

- If checked, Rollbase overwrites existing application components with those in the new application version. This happens for locked and partially locked applications regardless of whether this checkbox is checked.
- If unchecked, Rollbase only creates new application components and will not overwrite existing components to preserve any customizations made since the previous install or update. Note that locked and partially locked applications receive all updates for locked components, but unlocked components will not be modified.



The following provide examples of how overriding works:

- Suppose you created a portal and the updated application does not have a portal with the same name. The portal you created will not be overwritten, regardless of whether **Override Changes** is checked or not.
- Suppose you added three fields to an object. If **Override Changes** is checked and/or that object is locked by the application publisher, updating will override the object and the fields will not be available. Any pages to which the fields were added will no longer show those fields.

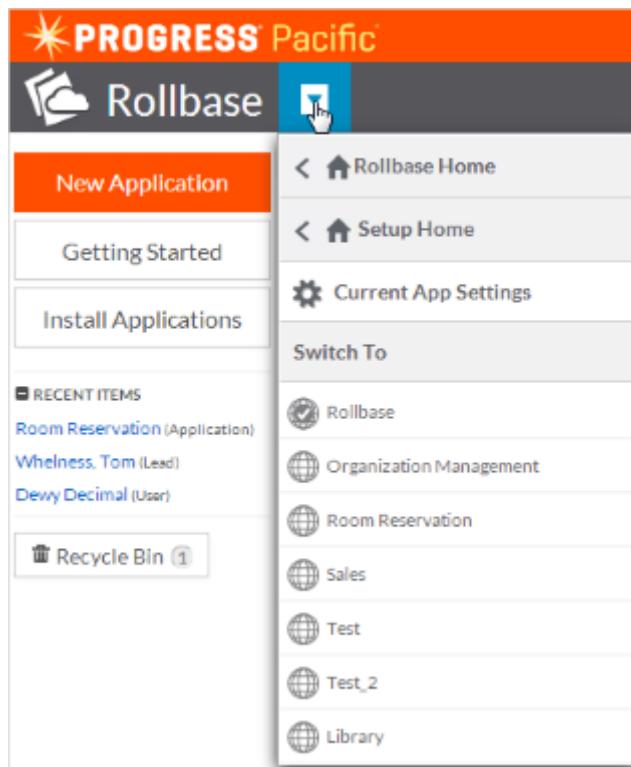
## Installing and Updating Applications from XML

Optional: add any pre-requisites for the task here.

If you received a Rollbase application as an XML file, you can install it in your own tenant.

To install an application from XML, follow these steps:

1. From the header menu, select **Setup Home**.



2. In the **APPLICATIONS SETUP** section, click **Applications**:

**PERSONAL SETUP**  
Manage your account settings and password  
[My Settings](#) | [Change My Password](#)

**APPLICATIONS SETUP**  
Create, install, update, manage, customize and integrate applications  
[Applications](#) | [Objects](#) | [Tabs](#) | [Portals](#) | [Hosted Files](#) | [Application Directory](#)  
[SOAP API](#) | [REST API](#) | [Code Generator](#)

**ADMINISTRATION SETUP**  
Create and manage users, roles, authentication, backups, batch jobs and more  
[Users](#) | [Roles](#) | [Transfer Owners](#) | [User Access Log](#) | [Support Access](#) | [Settings](#)  
[Account Settings](#) | [Billing And Support Settings](#) | [Currency Codes](#) | [Exchange Rates](#)  
[Whitelist](#) | [Backup](#) | [Batch Jobs](#) | [Global Text Search](#) | [System Jobs](#) | [Portal Visitors](#)  
[Online](#) | [System Events Log](#) | [System Errors Log](#)

A list of existing applications displays:

Action	Order No	Application	Version	Deployed	Published	Updates Available	Installation Date	Publisher
Edit	1	Rollbase	5	✓			06/13/2014	Progress Rollbase
Edit	2	Organization Management	6	✓			06/13/2014	Progress Rollbase
Edit   Publish	3	Room Reservation	3	✓			06/23/2014	
Edit   Publish	4	Library	13	✓			07/11/2014	

3. Click **Import From**.

The **Import From** screen displays.

Type	Description
<input checked="" type="radio"/> Progress Rollbase XML File	This option will install application developed in another Customer and serialized as XML file.
<input type="radio"/> Salesforce.com (Force.com)	This option will import your currently active Salesforce.com application including all of its objects, fields, page layouts and data.
<input type="radio"/> Convert MS Access Database	This option allows you to upload a Microsoft Access Database (mdb or accdb) and automatically convert it into a working web application (recommended when you want to start from an existing MS Access application).
<input type="radio"/> OpenEdge Service	Import Objects and Relationships from uploaded OpenEdge JSDO catalog file
<input type="radio"/> Mobile Project Backup	This option will create mobile application from a project backup archive (zip artifact)

4. Select **Progress Rollbase XML file** and click **Next**.

The **Install or Update from XML** screen displays.

Install or Update Application from XML

**Next >**

**Cancel**

Application XML

Application XML  No file chosen

**Next >**

**Cancel**

5. Click **Choose File** and browse to select the application XML file.

6. Click **Next**.

A tree of components included in the application displays:

- For new installations, you have the option to uncheck boxes for locked components to avoid having them installed as part of the application.

- For upgrades, if the application is fully locked, all components will be updated/created/deleted automatically during the installation process. If the application is partially locked, all locked components will be automatically updated/created/deleted during installation. You have control over what happens to unlocked components uncheck boxes for components you do not want to upgrade.
  - If a component was deleted from the latest version of the application but is present in the tenant where you are installing the update, this component will be listed at the bottom of the tree. Check the box next to each deleted component if you wish to delete that component as well. This is a safety precaution designed that allows you to bypass such deletion.
  - Components deleted in the latest version of the application will be deleted in the destination tenant if the **Override Changes** box is checked. If an application or portal page is updated, the page's entire structure (sections, cells) will be replaced by the structure defined in the application XML.
7. Review the application tree and uncheck boxes next to components you do not want to install/update.
  8. Click **Install** to continue with the installation.

## Creating and Managing Objects, Fields, and Relationships

If you use the [Quick Create wizard](#) to create an application, you can create objects, fields, and relationships between objects when you create the application. You can also create a new object definition in the context of an application, or from the list of available objects in setup.

There are two types of objects in an application:

- Core objects will be published and installed with the application, should you decide to distribute it in XML format. Core objects are explicitly assigned in the application edit page. When you create a new object with a tab, it becomes a core object by default.
- Dependent objects are pre-requisites for applications: they must be installed prior to installing the application. Dependent objects can be explicitly assigned in the application edit page as well. However, the system will automatically add objects as dependent objects to maintain the application's integrity in the following cases:
  - The **User** object is always included as a dependent object for all applications.
  - If any core objects have the **Approval** attribute, the **Approval** object will be added as a dependent object.
  - Any object associated with a tab which is not included as core.

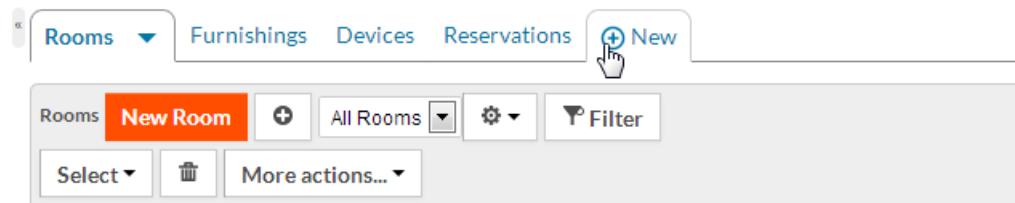
Topics in this section describe how to work with Rollbase objects, fields, and relationships.

## Creating a New Object Definition

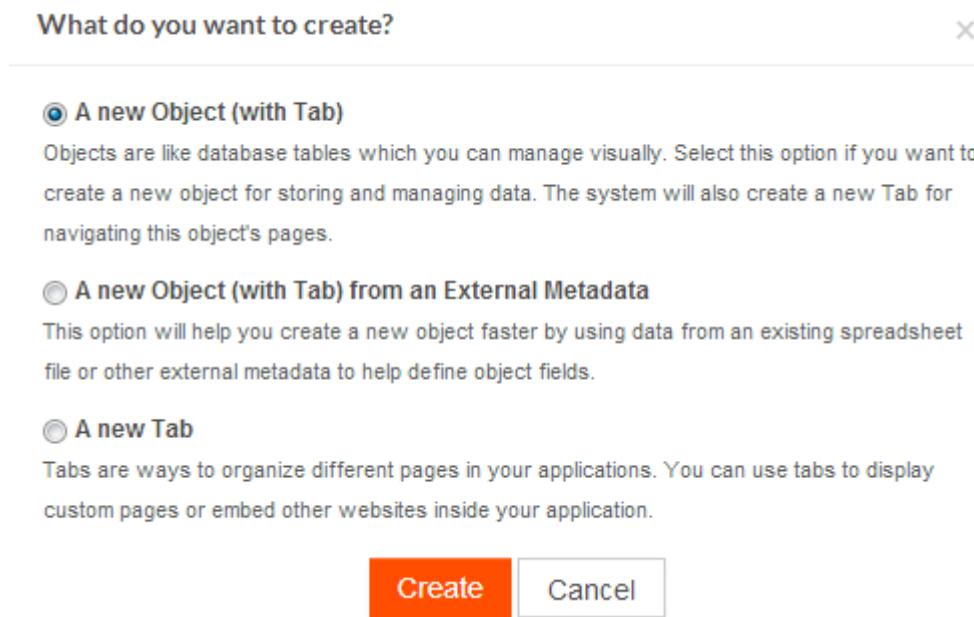
To create a new object definition:

1. Launch the **What do you want to create?** dialog in one of the following ways

- From within an application:
  - Click the **New** tab that displays to the right of the existing object tabs:



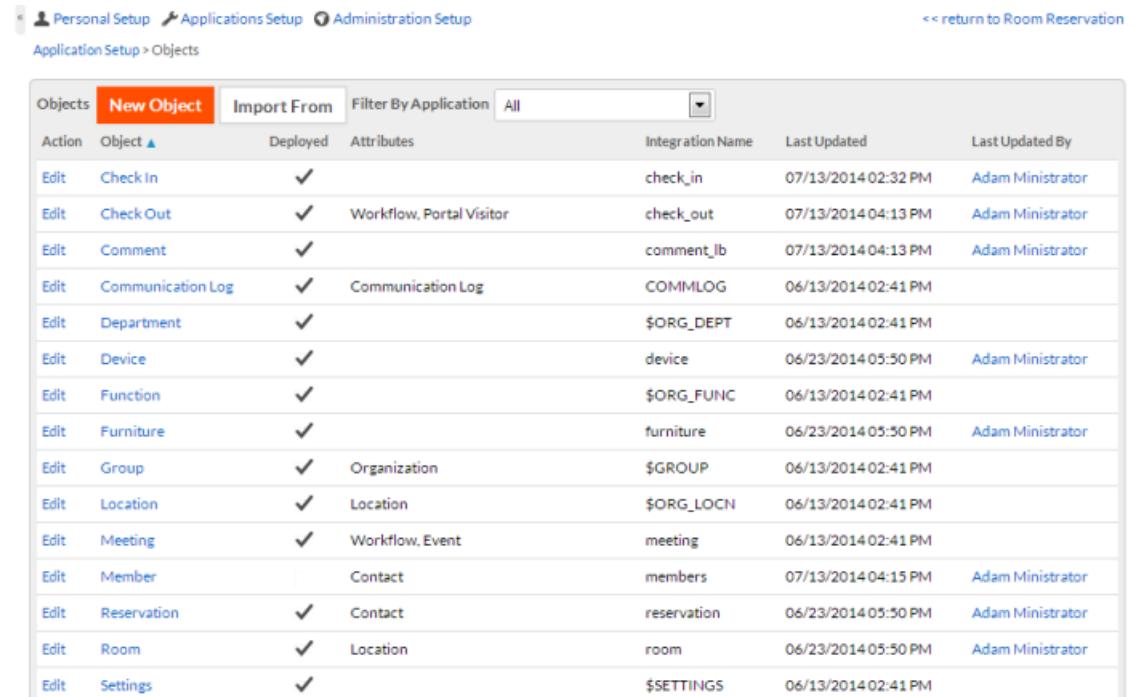
- The **What do you want to create?** dialog displays:



- Leave the default selection, **A new Object (with Tab)**

- From the list of all objects:
  - From the **Applications** dropdown list, select **Setup**.
  - In the **APPLICATIONS SETUP** area, click **Objects**.

A list of available objects displays:



The screenshot shows a table titled 'Objects' with a 'New Object' button highlighted in orange. The table has columns for Action, Object, Deployed, Attributes, Integration Name, Last Updated, and Last Updated By. The 'Object' column lists various system objects like Check In, Check Out, Comment, etc. The 'Deployed' column contains checkmarks. The 'Attributes' column shows details like 'Workflow, Portal Visitor' for Check Out. The 'Integration Name' column lists names like check\_in, check\_out, comment\_lb, etc. The 'Last Updated' and 'Last Updated By' columns show the date and time of the last update and the user who made it.

Action	Object	Deployed	Attributes	Integration Name	Last Updated	Last Updated By
Edit	Check In	✓		check_in	07/13/2014 02:32 PM	Adam Ministrator
Edit	Check Out	✓	Workflow, Portal Visitor	check_out	07/13/2014 04:13 PM	Adam Ministrator
Edit	Comment	✓		comment_lb	07/13/2014 04:13 PM	Adam Ministrator
Edit	Communication Log	✓	Communication Log	COMMLOG	06/13/2014 02:41 PM	
Edit	Department	✓		\$ORG_DEPT	06/13/2014 02:41 PM	
Edit	Device	✓		device	06/23/2014 05:50 PM	Adam Ministrator
Edit	Function	✓		\$ORG_FUNC	06/13/2014 02:41 PM	
Edit	Furniture	✓		furniture	06/23/2014 05:50 PM	Adam Ministrator
Edit	Group	✓	Organization	\$GROUP	06/13/2014 02:41 PM	
Edit	Location	✓	Location	\$ORG_LOCN	06/13/2014 02:41 PM	
Edit	Meeting	✓	Workflow, Event	meeting	06/13/2014 02:41 PM	
Edit	Member		Contact	members	07/13/2014 04:15 PM	Adam Ministrator
Edit	Reservation	✓	Contact	reservation	06/23/2014 05:50 PM	Adam Ministrator
Edit	Room	✓	Location	room	06/23/2014 05:50 PM	Adam Ministrator
Edit	Settings	✓		\$SETTINGS	06/13/2014 02:41 PM	

**3. Click New Object.**

The **New Object** screen displays:

## Object: New Object

**Save** **Cancel**

## Object Properties

Define a singular and plural name for this object definition. These names will be used throughout your account to refer to one or more records of this type.

Singular Name  Example: Project

Plural Name  Example: Projects

What are Records of this object called? The Record Name field is used in pages, views, selectors, and search results that reference records of this type. Once this object has been created you can customize the display format of the Record Name field using merge fields.

Record Name  Example: Project

The Integration Name is used to reference this object via merge fields and the Web Services APIs. This name must be unique. Be careful when changing the Integration Name as it may affect existing templates, formulas and integrations.

Integration Name  Example: Project

Description

## Optional Object Properties

Select the appropriate properties based on how you want records of this object type to behave:

Property	Description
<input type="checkbox"/> Audit Trail	Select this option to enable creation of Audit logs when values of selected fields are changes or by invoking triggers and API. In addition create Audit logs when record is: <input type="checkbox"/> Viewed <input type="checkbox"/> Created <input type="checkbox"/> Edited <input type="checkbox"/> Deleted
<input type="checkbox"/> Flagging	Select this option if you want the ability to flag records for follow-up. Flagging applies uniquely to each user.

2. Fill in the fields as directed by the explanatory text. Required fields are indicated with a red bar. When you supply a **Singular Name** and move out of that field, the other fields are populated automatically with default values that can be changed. The object definition includes optional properties, attributes, and permissions. If you do not know which of these you might need, you can come back and add them later.
3. When you are satisfied with the object definition, click **Save**.

If you leave the box checked to create a new tab, the following screen displays:

Personal Setup Applications Setup Administration Setup

Application Setup > Objects > New

Tab: New Tab

Save Cancel

Properties

Tab Name: MyNewObjects

Description:

Parent Tab: None (top level tab)

Add to Applications

Select All Rollbase Organization Management

Room Reservation Library\_Test

User Roles | Select All | Select None

Administrator	<input checked="" type="checkbox"/> View
Portal Visitor	<input type="checkbox"/> View

Individual Users | Select User

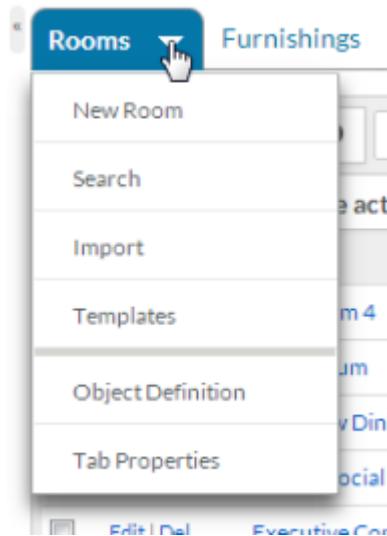
Save Cancel

4. Edit the **Properties** as desired. If you select a **Parent Tab**, the tab for this object will not appear in application tabs. Instead, it will be available from the drop-down menu of the parent.
5. In the **Add to Applications** section, select the applications to which you want to add this object.
6. In the permissions box, select the roles and users you want to be able to view this object.
7. Click **Save**.

## Viewing and Editing an Object Definition

View an object definition in either of the following ways:

- Navigate to the object's tab in any application in which it is used.
1. Click the arrow next to the object name:



**2. Select **Object Definition**.**

- Navigate to the list of all objects:
  1. Navigate to **Setup**.
  2. In the **APPLICATIONS SETUP** section, click **Objects**.
  3. Find the object you want to view and click its name.

The object definition displays in the view page:

The controls on the object view page provide the following functionality:

1. The shaded ribbon at the top contains links to subcomponent sections, where you can create, edit, or delete object subcomponents, such as **Fields**, **Relationships**, and **Pages**.
2. The **Edit** button takes you to a screen where you can edit the following:
  - Deployment status
  - Object properties and attributes
  - Permissions
3. The **Delete** button allows you delete this object definition, but the object must first be undeployed.
4. The **More actions** menu allows you to do the following:
  - Clone
  - Run triggers
  - Add a condition to the object

## Adding Fields

This topic describes how to create a field from the object definition. You can also create basic fields using the **Page Editor**. To create advanced Field types or Portal Field types, create the Field from the Object definition's Page in the Setup area, as shown here.

For performance reasons, the amount of certain types of fields you can add to a particular object is limited by Rollbase. Private Cloud users can control these limits as described in [Adding Columns to a Private Cloud Database](#) on page 478

To add a field to an object definition, navigate to the **Fields** section and click **New Field** or **New Formula Field**. Choose the most appropriate field type based on the kind of data you want to store in that field. For example, for a field that will store prices, you might select currency; for a description field, you might select a text area.

Click **Next** to define properties. The most basic property is the label that is used as an identifier in all Pages, Views, Reports and any other places the field appears in your application.

As shown in the screen below, some attributes are common to all fields:

- **Field Label** (mandatory) is displayed on the left from the field on View, Edit, New Record etc. pages.
- **View Header** (optional) is used in headers of Views and Reports. If not specified, Field Label is used.
- **View Width** (optional) is used to set width (in pixels or %) of columns in Views and Reports. If not specified, browser calculates width of column automatically.

Field Properties

Field properties are global settings that apply to this field wherever it is presented for input or display.

Field Type: Text

Field Label:

You can specify text label to be used in the headers in Views and Reports. This label is optional. If not specified, Display Label will be used.

View Header:

View Width:  pixels or %

Each field has a varying number of additional properties based on its type. For example, Currency fields have a size and a Currency Format property that allows you to select the type of currency to use. Text Areas have a default height and width, as well as a property specifying whether or not to use a rich text editor.

Each field has a set of advanced properties. Different advanced properties are available depending on the type of the field. For example, all text-based fields can be indexed as part of the full text search engine, but image fields cannot. Most Fields can be audited so you can keep a historical log of when a field has changed its value, but text area fields cannot be audited.

After you have defined properties, you will also need to ensure that each field has a unique integration name. Rollbase suggests a name based on the label you entered earlier. The integration name is used to reference this field via merge fields or with Rollbase APIs.

Integration Name

The Integration Name is used to reference this field via merge fields and the Web Services APIs. This name must be unique. Be careful when changing the Integration Name as it may affect existing templates, formulas and integrations.

Integration Name

Field-Level Help

Field-Level Help

If an application has Field-Level Help enabled, this text will be displayed upon clicking the  icon next to this field's label

The **Field-Level Help** box allows you to define help text for the field that will appear in a popup when the user clicks the help icon next to the field.

The last step in creating a field is deciding on which Pages and Views to include the field by. The screen displays a list of Application Pages on the left, a list of Portal Pages in the center (the example shown below does not have Portal Pages) and a list of Views on the right. Check all that apply.

MyText: Add to Pages and Views

**Save** **Save & New** **Cancel**

Add to Pages and Views

Select the pages and views that should include this field. It will be added to the section in each page marked as the "Default New Fields Section". The field will not appear on any pages if you do not select one or more pages below.

You can always add or remove a field from any page or view, or change its location on that page, by editing that page.

<b>Application Pages</b>	<b>Views</b>
<input type="checkbox"/> Select All	<input type="checkbox"/> Select All
<input checked="" type="checkbox"/> Edit User	<input type="checkbox"/> All Users
<input type="checkbox"/> Mass Update	<input type="checkbox"/> ISV Partners
<input checked="" type="checkbox"/> New User	<input type="checkbox"/> Library Employee
<input checked="" type="checkbox"/> Personal Setup	<input type="checkbox"/> Library Web User Management
<input checked="" type="checkbox"/> View User	

## Field Integration Name

Each field has a unique **Integration Name** that is used by formulas, templates, and Rollbase APIs. This name must be unique across all fields in an object definition. When you create a new field, Rollbase automatically assigns an **Integration Name** derived from display label; you can change it at any time. Integration names for System and automatically created fields cannot be changed. Progress recommends using a naming strategy for fields. Changing the **Integration Name** will affect all existing templates, formulas and integrations that use that field.

A field **Integration Name** must conform to the following rules:

- Cannot be more than 20 characters long.
- Must start with a letter.
- Cannot include space ‘ ‘ or pound ‘#’ characters.
- Must be unique (case-insensitive) across other field integration names in the same object.

- Must not be one of the reserved system integration names:

```
startdate, due date, act, id, id2, ids, srcid, destid, objdefid, cloneid,
returned, view, actionid, relid, relatedid, gridid, gridrows, griddeleted,
country, state, priority, commtype, category, isactive, Fieldname, tag
```

## Enabling Field-Level Help

Field-level help allows you to enable popup-style help for any field. You can enable this behavior per application as follows:

1. Navigate to the application settings, and edit the application definition to set the **Enable field-level help for this application** property.
2. Enter help text in the **Field-Level Help** box. You can use HTML tags to format the text, such as <br/> to control line breaks.

On object pages, a ? icon displays next to fields for which help has been defined. Roll your cursor over the icon or click it to display the help text.

The screenshot shows a 'New Room' application page. At the top, there are buttons for 'Save' (orange), 'Save & New', and 'Cancel'. Below this is a 'Room Information' section with fields for 'Room', 'Reservation', and 'Phone Number'. The 'Phone Number' field has a question mark icon. A tooltip appears when the cursor is over the question mark, containing the text: 'Enter the phone extension for this room, leave blank if the room has no permanent phone.' To the left of the 'Phone Number' field is a 'Furnishings' tab, and to the right is a 'Furniture' tab. The 'Description' field is at the bottom right.

## Field Actions

When viewing the list of Fields on an Object definition you can see several links in the Action column. Each Field exposes an Edit link that allows you to make changes to that Field's definition. Other actions are discussed in this section.

### Cloning Fields

You can clone any Field except System-level Fields (such as ID) and other Fields created automatically by the system. When cloning a Field, Rollbase gives you the option of adding the cloned Field to a different Object definition. Properties of original field (including permissions, validation script, and DOM event handlers) will be copied to a new field.

### Deleting Fields

You can delete any Field except for system-level Fields (such as ID) and other Fields created automatically by the system.

Note: Fields created by Object attributes and relationships will be deleted automatically if the attribute is removed or the relationship is deleted.

When you confirm that a Field should be deleted, the system will:

- Delete the Field definition from the Object definition.
- Delete all of that Field's data for all existing records.
- Remove the Field from all Pages, views and reports.

The system will not modify your formulas, templates, integration links and API calls containing a reference to the deleted Field. It is your responsibility as the Application developer to ensure a deleted Field is no longer used.

## Replacing a Picklist

If you need to change the value of a picklist, you should not directly edit that value by editing the picklist Field because Rollbase assumes that you want to delete the old value and create a new one (not necessarily in the old one's place). In this case, values assigned to existing records will be lost if they used the old value. In order to preserve these assigned values you need to use the Replace action, rather than editing the picklist Field.

Select the value to replace and enter the new value. This will replace the old value in all existing records with the new value you define.

Replace Picklist Value	
Value to replace	Small
Replacement Value	Medium
Integration code	M

## Converting Field Types

In the course of application development, you may want to change the type of a Field you have already created. Some Fields do allow you to change the Field type at any point in the future, as listed below:

- Text Fields can be converted into Email, Picklist, Radio Buttons, Text Area, or URL Fields.
- Currency Fields can be converted to Decimal or Integer Fields.
- Date Fields can be converted to Date/Time Fields.
- Date/Time Fields can be converted to Date Fields.
- Decimal Fields can be converted to Currency, Percent, Integer, or Text Fields.
- Email Fields can be converted to Text or Text Area Fields.
- Integer Fields can be converted to Currency, Decimal, Percent, or Text Fields.
- Percent Fields can be converted to Decimal, Integer, or Text Fields.
- Picklist Fields can be converted into Multi-Select Picklist, Radio Buttons, or Group of Checkboxes Fields.
- Multi-select Picklist Fields can be converted into Group of Checkboxes Fields.

- Radio Buttons Fields can be converted into Picklist, Multi-Select Picklist, or Group of Checkboxes Fields.
- Group of Checkboxes Fields can be converted into Multi-Select Picklist Fields.
- URL Fields can be converted to Text Area or Text Fields.
- Auto-Number Fields can be converted to Text Area or Text Fields.

When a text Field is converted into a Picklist or Radio Buttons Field, Rollbase will:

- Analyze the value of the text Field for each record of that Object type.
- Create values for all unique values found in the text Field.
- Replace original text values with pointers to the newly created values.

## Field Level Permissions

You can restrict access to a particular Field to certain user roles by hiding it completely (unchecked the "View" box) or disallow editing of the Field's value (check the "Read Only" box). If set, these restrictions will be applied on top of all other Rollbase permissions described in [Security and Access Control](#) on page 375.

## Field Validation

On most Field types you can define your own server-side validation logic by writing custom JavaScript that makes use of record-level Field values, as well as related Field values, to return a custom error messages if your required conditions are not met.

To define custom validation, enter your JavaScript expression in the text area. After your users submit data to the server, this expression will be evaluated. If that evaluation yields a string value, that value will be considered an error message and the user will be returned back to the data entry Page with the error message displayed (all form data will be preserved). For example, the following validation ensures that the value of an amount charged Field is no more than 100 less than the value of a total due Field:

```
if ({!total_due}>{!amount_charged}+100) {
    return "The amount charged is too small. Please make sure the amount charged
    is no more than 100 less than the total due.";
}
```

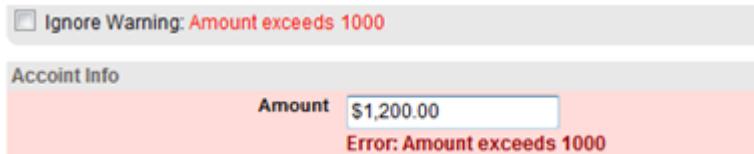
In some situations when a record is edited, you may want to access the value of a Field before it was updated by the user, as well as the value after it was updated. To do this, use the `#before` suffix in the merge Field. For example:

```
if ({!balance#before}-{!balance}<100) {
    return "The balance change is too large. Please make sure the new balance
    is no more than 100 less than the initial balance.";
}
```

When the user is saving data (for new or existing record) and field's validation formula is evaluated into error message the saving process cannot continue: the user is redirected back to new/edit page and red error message is displayed below the field.

The screenshot shows a form titled 'Account Info'. There is a single input field labeled 'Amount' with the value '\$1,200.00'. Below the input field, a red horizontal bar displays the error message 'Error: Amount exceeds 1000'.

This behavior changes if you select an option "Treat this Validation Rule as a Warning". In this case at runtime the user can check "Ignore Warning" box and proceed with saving despite of the warning.



Ignore Warning: Amount exceeds 1000

Account Info

Amount \$1,200.00

Error: Amount exceeds 1000

## JavaScript Event Handlers

Field-level events allow you to define custom JavaScript code that executes when a DOM event is invoked. You can attach JavaScript to any of the following DOM events for a Field: onchange, onclick, onfocus, onblur, onmousedown, onmouseup, onmouseover and onmouseout.

By calling JavaScript functions defined in your Pages or in hosted JavaScript code libraries, you can use Field-level events to create a variety of dynamic Page behavior. In addition, you can use the Rollbase AJAX API to add a further level of dynamism to your Pages. For more information about client-side scripting in Rollbase, see [Programmatic Client-side Customization](#) on page 247.

## Adding Fields to Pages, Views, Reports

Any time you create a new field, a dialog will prompt for the pages and views on which to add the field. You can add or remove existing fields from object pages or create new fields with the **Page Editor**, see [Editing Pages](#) on page 224. You can also edit views and reports to add, move, or remove fields from appearing in them.

## Deleting an Object Definition

To delete an object, you must first un-deploy it by editing the object definition and un-checking the **Deployed** setting. Note that deleting an object definition also deletes:

- All records of this object type.
- All associated sub components, including fields, pages, and templates.
- All associated menus.
- All relationships with other Objects.



**Warning:** You cannot undo the deletion of an object Definition.

---

# Views

Object definitions contain view components. A view is a result set, a way to display a list of object records to act on individually or in bulk. Each object definition can have an arbitrary number of associated views. When you create an object, a list view of all objects is created for you. If you select certain properties or attributes, Views display object fields in columns. Most columns are sortable. Normally the record name field is included as a column, so users can click on each record as desired to be taken of that record.

The total number of records in the view is based on the permissions of the viewer. In complex cases, such as when **Location**, **Department**, and **Function** functionality is enabled, the number of records is not shown to avoid over extending resources.

You can change the default view used for any generic or object page by editing the page, selecting the view component and then selecting the desired view in the **Default List View** field in the **Properties** box. If the **Default View** was selected, the system adds "Default View" link. By clicking on that link you can always restore the default view and its sorting options.

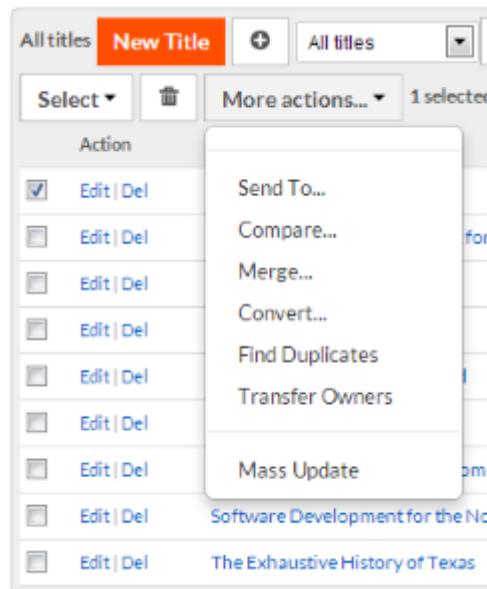
The following shows a typical view:

You can navigate through pages of records using the buttons in the shaded area on the top right. Click on column headings to sort records by column. The left column checkbox in the table of records allows you to select multiple records on which to perform an action. Selections will be maintained as you navigate through pages of records.

The view header contains the following buttons and menus from the shaded section top left to right:

- The top row of includes:
  - The view name, such as All titles, as shown above.
  - The **New** button for creating a new record.
  - The **Quick Create** (+) button, which opens a dialog to create a new record without leaving the view page. You can edit the **Quick Create** Page to specify which fields should appear in the dialog.
  - A menu for selecting another view.
  - View actions from the drop-down menu next to the gear icon include:
    - **Edit this View** provides a quick way to make changes to settings such as name, columns, sorting, grouping and filtering.
    - **Edit Row Colors** allows you to set color codes for rows in the current view. A
    - **Create New View** provides a quick way to create another view for this type of objects.
    - **Clone This View** provides a quick way to create a copy of the view.
    - **Delete This View** provides a quick way to delete the view. Rollbase requires that at least one view exists for each object type, so you cannot delete a view if it is the only one available.
    - Export to XLS, CVS or Google. XLS exports the entire view to Excel (XLS) format. CSV exports the entire view to a comma-separated value flat file (CSV). The Google option opens the currently selected view in a Google Docs spreadsheet. Note that your Google account must be set up in your personal settings to use the export to Google functionality. See [Personal Setup](#) on page 422

- **Filter** opens a section that allows you to dynamically add filters to select records from the view without changing the view itself.
- The controls on the second row of the header work with selected records and include:
  - **Select** picklist providing a way to manage your current set of selections on the current page of the view as well as across all pages.
  - **Delete** allows you to move all of the currently selected records to the Recycle Bin. Administrators can recover records from the Recycle Bin if they are deleted by mistake.
  - A **More Actions** menu with features available for group operations.



The items in the **More Actions** menu vary depending on the object definition and can include:

- **Tag** for objects with tags enabled, allows you to add keywords to selected records so you can easily find them in the future. See [Tagging Records](#) on page 148 for more information.
- **Send To** allows you to email the selected records.
- **Compare** allows you to compare the selected records.
- **Merge** allows you to merge the selected records.
- **Convert** allows you to convert the records to a different type.
- **Find Duplicates** allows you to find duplicates.
- **Transfer Owners** allows you to transfer owners.
- **Workflow actions**, when enabled. See [Workflow Actions](#) on page 195 for more information.
- **Mass Update** allows you to perform the same change to multiple records.

# Relationships Between Objects

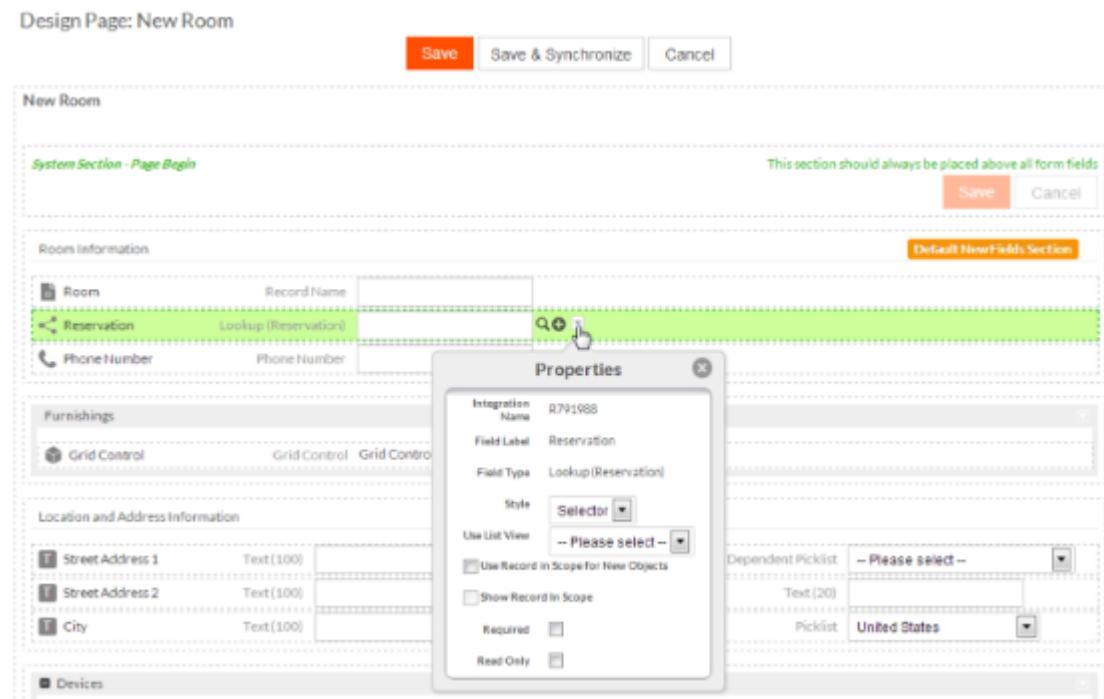
When creating a new relationship Rollbase will also create:

- Lookup Fields for each Object and add them to selected Pages.
- Related View components for each Object and add them to selected Pages.
- Grid Control components in user-selected Pages.

Lookup Fields are used to select one or more related Object records when editing or creating a new record. Fields of this type cannot be created manually; Lookup Fields are created automatically when a relationship is created.

## Page Lookup Fields

By default, Lookup Fields allow users to pick one or more records by displaying a Selector Page in a popup window. However, often times it is faster and more convenient to allow users to select records using a picklist or multi-select picklist. Rollbase allows you to change whether a Lookup Field displays as a popup selector or a picklist/multi-select picklist, via a Page level property.



To change this setting for any Lookup Field on any given Page, follow these steps:

- Edit the Page with the Page Editor.
- Locate and select the Lookup Field.
- Roll your cursor to the left of the lookup and plus icons and click the menu arrow that appears.

- In the Properties box, choose one of the following: 1) "Selector" as the Field's style if you want users to select records from a popup dialog, 2) "Picklist" if you want the lookup Field to act like a picklist (for 1-1 and N-1 relationships), 3) a multi-select picklist (for 1-N and N-N relationships), or 4) "Hidden" for non-editable relationships (such as a relationship between Invoice and Invoice Line Item).
- You can also select the Default List View to be used for this popup selector when users use this Lookup Field. This allows you to choose which sorting and filtering of related records are available to your users when they are selecting related records.
- Check the "Use Record in Scope for New Object" property if you want lookups to be pre-populated for new records (this is the only option to populate hidden lookups).

See [Editing Pages](#) on page 224 for more information on using the Page Editor.

Note: If Lookup field is rendered as picklists it has "Please Select" empty option. Number of items in picklist is limited to 1000 records to avoid uncontrolled growth of HTML page.

## Global Lookup Field Properties

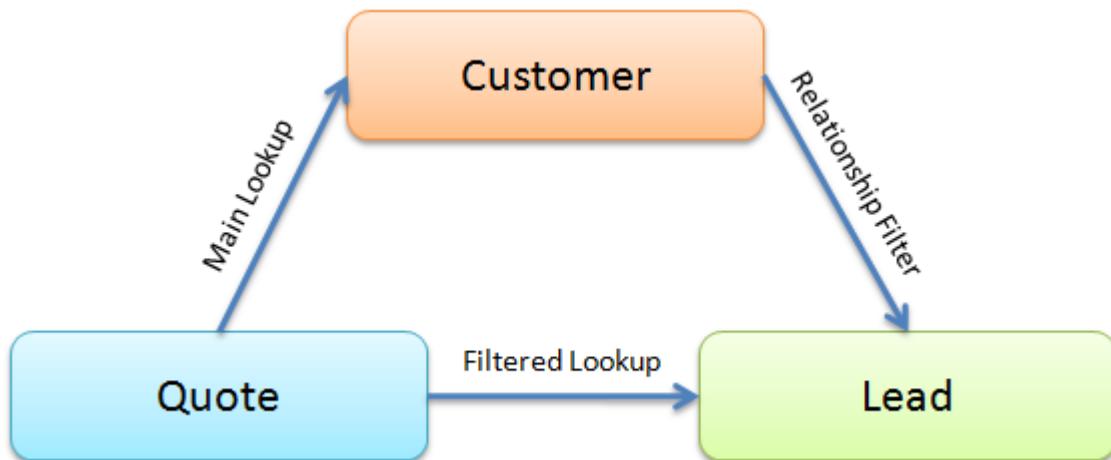
In addition to page-level settings defined in the **Page Editor**, lookup fields have global properties. You can set these properties when editing a lookup field from an object definition:

- Links Alignment: choose how links should be aligned on View pages and in Views: horizontally (default) or vertically.
- Hide "Quick Create" button for this Field: Check this box if you want to hide the button used to create and select a new related record rather than choosing an existing one.
- Main Lookup and Link Lookup: Use this setting to restrict the available choices for related records based on the current selection in another lookup Field (referred to as the "Main Lookup").

The **Quick Create** button is not available (regardless of setting) in following cases:

- If the user does not have permissions to create new record of related type
- In pop-up (like Print Preview) and portal pages
- In "Quick Create" pages
- If relationship has 1-1 or N-1 cardinality and the related object is already selected

Consider an example: A Quote Object has relationships with Customer and Lead Objects and the Customer Object has a relationship with the Lead Object as well. While creating or editing a Quote record, you need to make sure that only leads related to the selected customer record can be selected. In this example, you would choose the Quote-Customer relationship as the Main Lookup and the Customer-Lead relationship as the Link Lookup.



Warning: Due to system limitations you can have at most one main lookup field on the page.

Autocomplete Field: Select the field to be used to find matches when users start typing in the lookup Field. After typing 3 characters Rollbase starts displaying matches automatically for the user to select as desired.

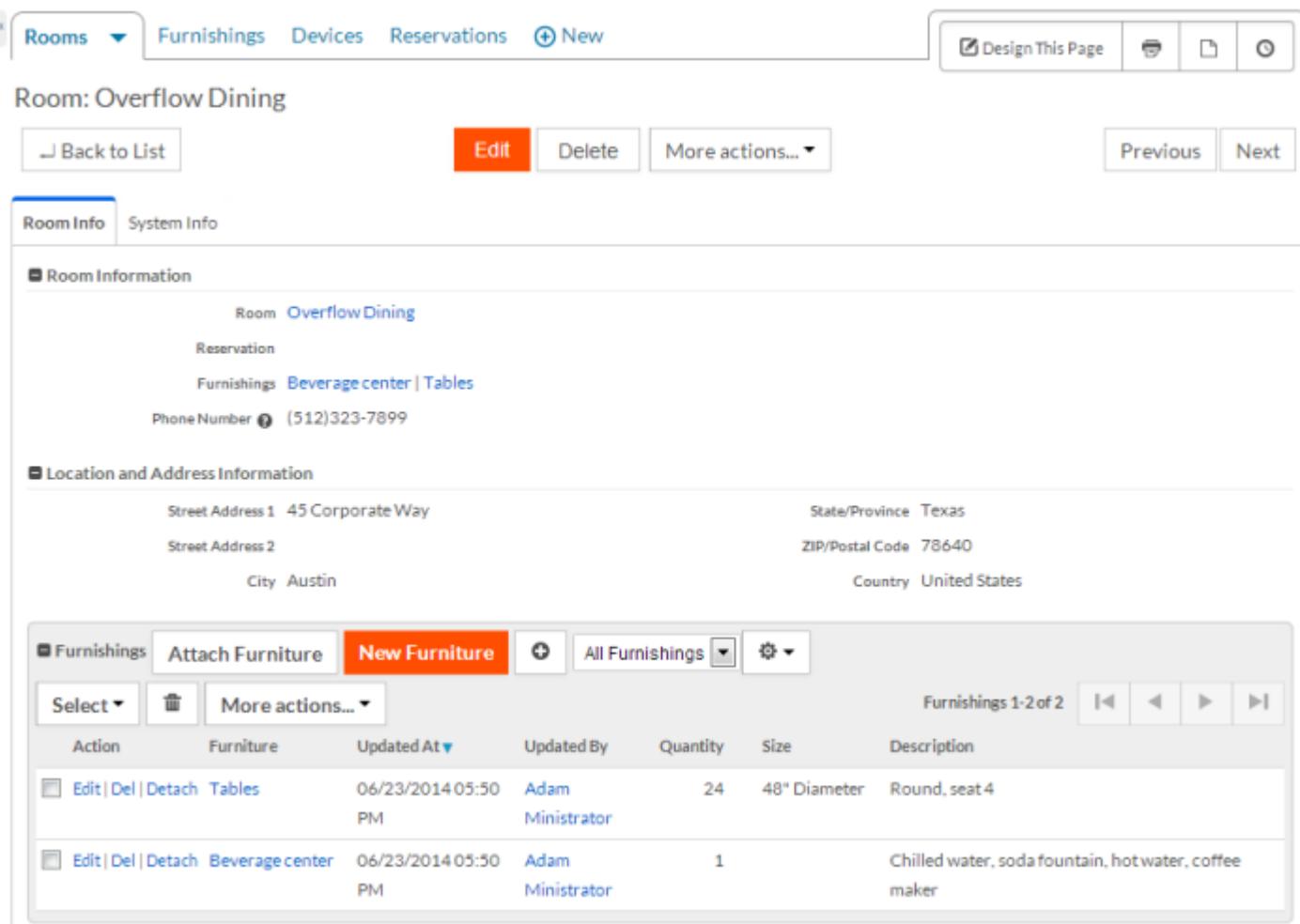
Autocomplete Rule: You can select "Starts With" or "Contains" to change the matching behavior of the autocomplete feature.

Address	<input type="text" value="san"/> <span style="font-size: 2em;">🔍</span> <span style="font-size: 1em;">+</span>
Contact Owner	<div style="border: 1px solid #ccc; padding: 5px; display: none;">           San Francisco [San Francisco]            San Mateo [San Mateo]            Santa Clara [Santa Clara]         </div>

Default Value: You can optionally select a record to be selected in this field by default when users create new records.

## Related Views

Related Views are used to display a list of related records for Objects with 1-many or many-many relationships. For example, if you have an room object with a one-to-many relationship with furnishings, the room view page can contain a related view component:



The screenshot shows a Room record detail page for 'Overflow Dining'. The top navigation bar includes 'Rooms', 'Furnishings', 'Devices', 'Reservations', and a 'New' button. To the right are buttons for 'Design This Page', 'Print', 'Copy', and 'Reset'. Below the navigation is a toolbar with 'Back to List', 'Edit' (highlighted in orange), 'Delete', 'More actions...', 'Previous', and 'Next'. The main content area has tabs for 'Room Info' (selected) and 'System Info'. The 'Room Information' section shows the room name 'Overflow Dining' and its location 'Austin, TX'. The 'Location and Address Information' section shows the address '45 Corporate Way, Austin, TX 78640, United States'. Below this is a grid control for 'Furnishings' related to the room. The grid has columns for Action, Furniture, Updated At, Updated By, Quantity, Size, and Description. It lists two items: 'Tables' (Quantity 24, Size 48" Diameter, Description Round, seat 4) and 'Beveragecenter' (Quantity 1, Description Chilled water, soda fountain, hotwater, coffee maker). Navigation arrows are shown at the bottom of the grid.

Related View components must be placed in their own dedicated sections in Pages. The Page Editor enforces this rule. You can use the Page Editor to define whether or not you want to hide or show controls of the related view such as the "Quick Create" and "New" links. You can also use the Page Editor to define what View should be shown by default. For more information about configuring related views in Pages using the Page Editor, see [Editing Pages](#) on page 224.

Related Views are only shown on View Pages for specific records and automatically filter the data displayed to only show those records that are related to the current record being viewed. For more information on creating and configuring Views, see [Views](#) on page 130.

## Related Grid Controls

While Related Views can be used to view related records on View Pages, Grid Controls can be used to edit, create, or delete related records inline on New and Edit Pages. This is particularly useful for building master-detail input, such as a Quote with Quote Line Items, Invoices, Purchase Orders, Time and Expense entry, etc. Grid controls are useful for any situation in which you need an arbitrary number of line items all editable on the same Page.

For example, the following screen shows a grid control that allows users to add multiple related furniture records to a room record.

Room Information

Room	Atrium Social Corner	<input type="button" value=""/>																							
Reservation	<input type="text"/>	<input type="button" value=""/>																							
<table border="1"> <thead> <tr> <th>Furnishings</th> <th>Add Furniture</th> <th><input type="button" value=""/></th> </tr> <tr> <th>Quantity</th> <th>Furniture</th> <th>Description</th> <th>Size</th> </tr> </thead> <tbody> <tr> <td>24</td> <td>Bistro Chairs</td> <td>Plastic molded</td> <td><input type="text"/></td> </tr> <tr> <td>2</td> <td>Glass Buffet</td> <td>Modern design, countertop height</td> <td>50" X 24"</td> </tr> <tr> <td>6</td> <td>Glass Tables</td> <td>Seats 4</td> <td><input type="text"/></td> </tr> <tr> <td>1</td> <td>Revolving Dessert Case</td> <td>Refrigerated, with 6 glass shelves</td> <td>36" square</td> </tr> </tbody> </table>			Furnishings	Add Furniture	<input type="button" value=""/>	Quantity	Furniture	Description	Size	24	Bistro Chairs	Plastic molded	<input type="text"/>	2	Glass Buffet	Modern design, countertop height	50" X 24"	6	Glass Tables	Seats 4	<input type="text"/>	1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square
Furnishings	Add Furniture	<input type="button" value=""/>																							
Quantity	Furniture	Description	Size																						
24	Bistro Chairs	Plastic molded	<input type="text"/>																						
2	Glass Buffet	Modern design, countertop height	50" X 24"																						
6	Glass Tables	Seats 4	<input type="text"/>																						
1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square																						

When adding and configuring a Grid Control on a New or Edit Page, you can define the Fields shown in each row, attach custom JavaScript event handling code for performing calculations and other operations when Fields are updated or rows are added and deleted.

For more information on setting up and configuring Grid Controls, see [Using Grid Controls to Manage Multiple Records](#) on page 227.

## Working with Records

The following topics describe how to work with records:

- [Cloning and New Record Creation](#) on page 136
- [Protecting Records](#) on page 139
- [Record Conversion](#) on page 140
- [Find and Merge Duplicates](#) on page 143
- [Auditing](#) on page 145
- [Changing the Owner of an Object](#) on page 147
- [Updating Multiple Records](#) on page 147
- [Orphan Records](#) on page 149

## Cloning and New Record Creation

You can clone the currently selected record by selecting "Clone" from the "More Actions..." drop-down box on a record view page.

Rooms ▾ Furnishings Devices Reservations + New

Room: Overflow Dining

Back to List Edit Delete More actions...

Room Info System Info

Room Information

Room Overflow Dining

Reservation

Furnishings Beverage center | Tables

Convert

Clone

Send

Find Duplicates

This displays the New Record Page, where all Fields are pre-populated with values from the original record. You can change these Fields or accept values from the original record. The original record is not affected when you create a cloned record.

If the original record has a relationship with dependent records, cloning behavior is defined in the relationship. When you create or edit a relationship, you can indicate whether related records should be cloned when a parent record is cloned:

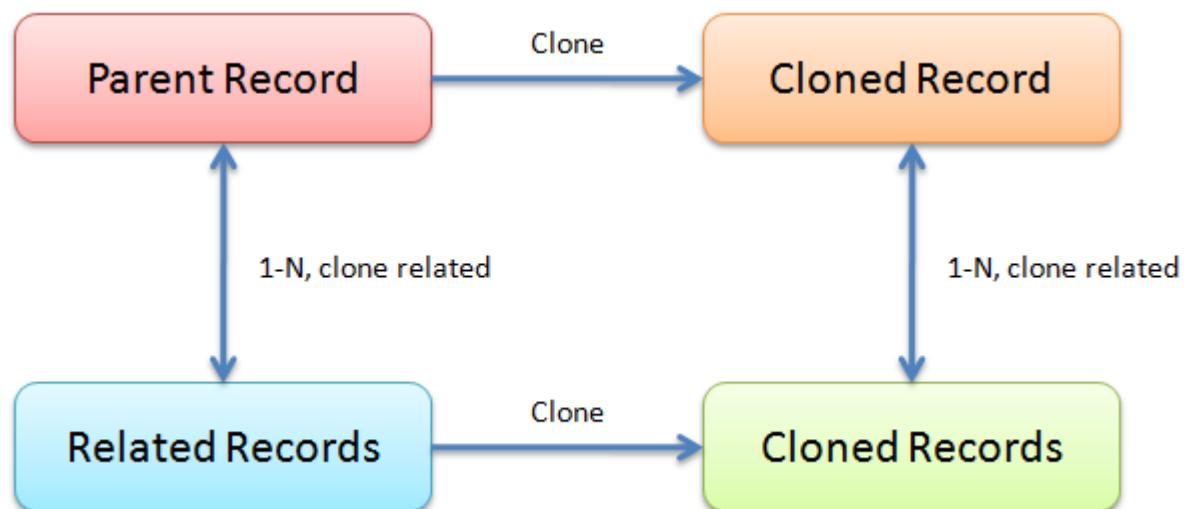
#### Cloning Control

Cloning control allows you to specify whether or not related records should be cloned whenever a record is cloned.

Clone all related Rooms when any Device record is cloned.

Clone all related Devices when any Room record is cloned.

When the "Clone all related..." flag is set, related records are cloned and attached to the resulting cloned records. For example, when you clone a Purchase Order header record that has relationships with Line Item records, it makes sense to clone all of its related line items.



Tip: When a New Record Page contains a Grid Control and a relationship has the "Clone related records" flag checked, the Grid control will be automatically pre-populated with information corresponding to the original related records.

Warning: If the New Record Page has a lookup Field that is configured to explicitly select related records to be attached to a cloned record, this supersedes the cloning mechanism described above.

## Cloning Control

Cloning control allows you to specify whether or not related records get cloned when the parent record is cloned. For example, if a user clones a Quote, we might also want to clone all of the related Quote Line Item records. But the opposite case is not true; if a user clones a Quote Line Item, we do not want to clone the related Quote record. Rollbase allows you to control this kind of behavior for each relationship.

### Cloning Control

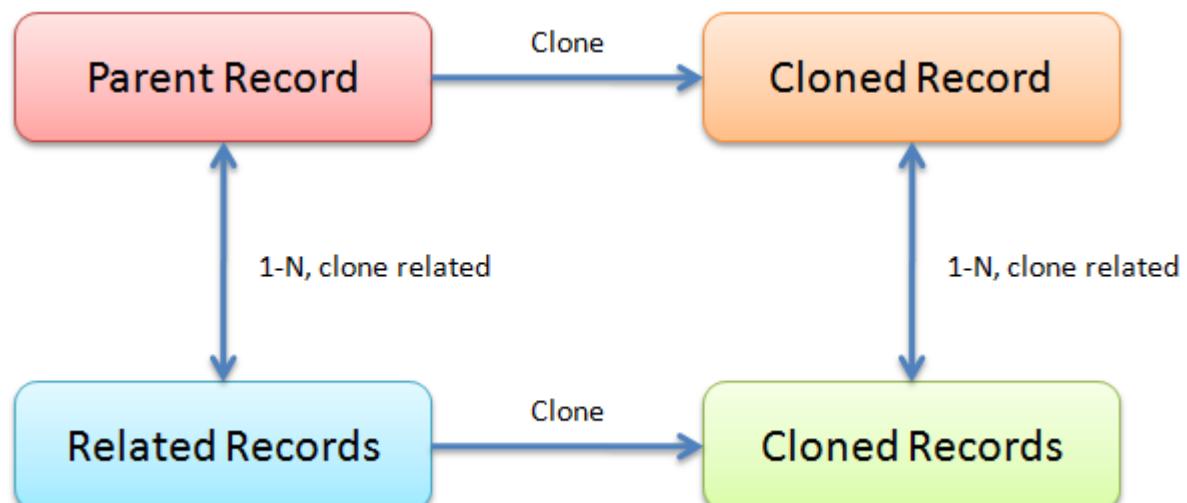
Cloning control allows you to specify whether or not related records should be cloned whenever a record is cloned.

- Clone all related Rooms when any Device record is cloned.
- Clone all related Devices when any Room record is cloned.

## Attaching Related Records to a Newly Created Converted Record

If source and destination Objects both have relationships with the same Object and that relationship has the "Clone all related..." setting enabled, then converted records can be attached to newly created cloned related records from the source record.

For example, consider Purchase Order and Invoice Object definitions that both have relationships to a Line Item Object definition. When a Purchase Order is converted into an Invoice, Line Items from the Purchase Order are cloned and attached to the Invoice.



To enable this conversion, check the box in the Clone Matched Related Objects section:



Tip: If you want to attach related records from source records to destination records without cloning, you can do so by mapping the destination Lookup Field to the source Lookup Field.

## Protecting Records

Rollbase provides a number of ways to protect records from being modified. One way is to set user permissions. For information on user permissions, see [Security and Access Control](#) on page 375. This section describes two other ways to protect record data:

### Locking Records

In some cases, you may want to "lock" a record to prevent it from being edited or deleted. To use this feature, enable the "Lockable" attribute on the Object. This adds a new checkbox called "Is Locked".

When you set this Field to "checked" (either by directly editing a record, via a mass update, Trigger, etc.), the record is moved into a "locked" state. This means:

- "Edit" and "Delete" buttons on the record View Page are disabled.
- "Edit" and "Del" links are not shown.
- A lock icon is shown as the value for the "Is Locked" Field.
- Related Lists shown on the record View Page do not display links to create, attach, edit, or delete related records.
- "Edit" and "Delete" buttons on the View Page for related records are disabled.

To remove locks on records, you can use a Trigger or the API to set the Is Locked checkbox to false. A new Trigger type called "Unlock" is created when the "Lockable" attribute is set.

Workflow actions are still available when a record is in the "locked" state. If you wish to disable them you can do so by adding a formula condition to each workflow action to return false if the record is locked:

```
!{ !isLocked}
```

### Condition Formulas

If locking mechanism is insufficient and you want to enable/disable "Edit" and "Delete" functionality based on record values you can do that by using Condition Formulas.

Select "Condition Formulas" from "More Actions" drop-down on Object View page. Now you can define formula which returns true or false depending on record's fields. If formula returns false for particular record - links and buttons to edit that record will be disabled. Example:

```
"{ !status#code }" != "CC"
```

This formula will disable editing (including Inline editing and Mass update) for records with status code "CC".

Similarly you can define a formula which will enable/disable "Delete" buttons and links.

When using Condition Formulas it is important to remember that:

- Permissions to edit/delete records (see [Security and Access Control](#) on page 375) apply first regardless of Condition Formulas
- Locking mechanism apply first regardless of Condition Formulas
- Condition Formulas do not apply to workflow actions and all types of API

## Record Conversion

You can convert records from one object type into records of another object type. You can initiate conversion in the following ways:

- From a List View for one or more selected records. For information on views, see [Views](#) on page 130.
- From a record View Page: Select "Clone" from the "More Actions..." drop-down list in the upper-right corner.
- Through triggers and workflow actions. For information on Workflow and Triggers, see [Triggers and Workflows](#) on page 179.

The procedures are similar:

1. Select the destination object type.
2. You can save the mapping for future use (Conversion Maps are stored as part of the source object definition).
3. Map the destination fields to source fields, formulas, constant values, or default values.
4. Perform the actual conversion.

All fields of the selected destination object are displayed except read-only (including template) fields. A drop-down box with a list of selectable source fields for direct conversion is displayed on the right of each destination Field. The system will ensure that data types of source fields offered for mapping are compatible with destination fields. Some fields may be selected by default if the display names of source and destination fields match. You can change these if desired.

You can also map a destination field to a constant value (i.e. a string, number, or particular record for lookups), or an expression. An expression can be a simple formula built from source field tokens the same way as other formulas are constructed (see [Adding Business Logic](#) on page 151). The mapping user interface does not provide helpers to build these expressions.

Example of a mapping expression:

```
{!total} - {!amount}
```

Tip: Expressions may include server-side API calls (see [Adding Business Logic](#) on page 151).

Picklist values can be transported from source to destination if:

- Picklists in source and destination objects are shared (have the same set of items)
- Source and destination picklist items have the same integration codes

If a destination field provides a default value, you can map a destination field to its default value (as automatically happens with new record creation).

**Important:** Deleted source records will be moved to the Recycle Bin without deleting dependent records. This setting has no effect if a conversion map is used in workflow action or trigger.

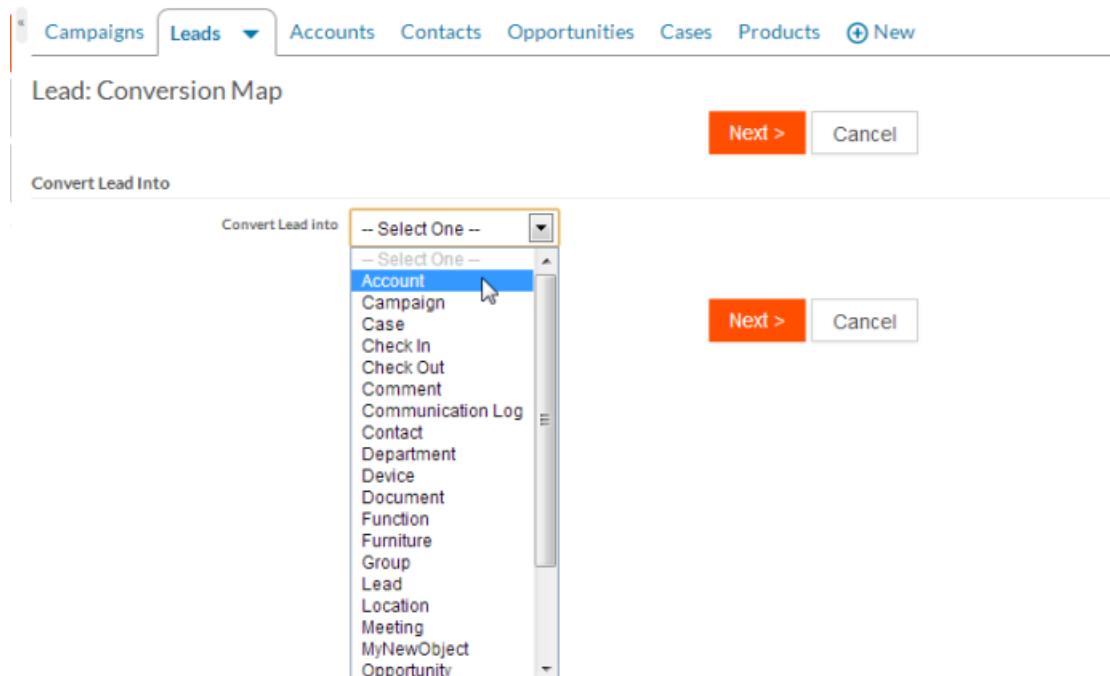
**Note:** You can also manage conversion maps from the source object definition's page in setup. In addition to name you can set Integration code for your map.

## Converting Records

To convert objects of one type to another type, follow these steps:

1. Navigate to the list of objects. One way to do this is as follows:
  - a) Select the application that contains the type of objects you want to view.  
The application object tabs display.
  - b) Select the tab for the object type of interest.
2. Select the objects to convert:
  - To convert all, from the **Select** drop-down menu, select **All Records**.
  - To convert a subset, check the box to the left of each record to convert.
3. From the **More Actions** drop-down menu, select **Convert**.

The **Conversion Map** screen displays. The following example shows the screen that appears when converting a lead from a sample CRM application to an account.



4. Select the destination object type. Available destination types include all deployed objects except the source object. Click **Next**

The screen to map object fields displays:

Lead: Conversion Map (Account)

Selected Records

Lead Wheliness, Tom

Select Saved Map

No maps have been saved to convert Lead into Account

Map Account Fields to Lead Fields

Account Fields	Lead Fields
Account Description	Description
Account Fax	Fax
Account Name	Full Name
Account Number	Not mapped
Account Phone	Phone
Account Rating	Rating
Account Site	Not mapped
Account Type	Not mapped
Accounts	Not mapped
Active	Not mapped
Annual Revenue	Annual Revenue
Billing City	Not mapped
Billing Country	Not mapped
Billing State/Province	Not mapped

- Map the fields as desired. Destination Fields marked as "required in all forms" are shown in red. You cannot continue the conversion process until these Fields are mapped. See [Record Conversion](#) on page 140 for more information.
- Optionally, check the box to have the original record deleted after the conversion.
- After checking to make sure that all fields will be mapped as desired:
  - Click **Convert** to convert the selected records.
  - Click **Save Map** to save these settings for a future conversion. After saving, you will be returned to the map so that you can convert the records or cancel.
  - Click **Cancel** to cancel.

## Converting Records to a Different Type

Click the "Convert..." option in the "More actions..." menu in the View header to convert all of the selected records into new records of a different object type. Select a target object type to display the **Conversion Map** Page, where you can map values from the selected record's fields to Fields of the new object type. You can save these conversion maps for future use. When defining a conversion map, you can choose whether or not to delete the source record after the conversion takes place. For more information about conversion maps, see [Record Conversion](#) on page 140.

## Find and Merge Duplicates

This feature allows you to easily find potential duplicate records among a set of records and then merge them with a currently selected record. To use this feature:

1. Navigate to the list of objects. One way to do this is as follows:
  - a) Select the application that contains the type of objects you want to view.  
The application object tabs display.
  - b) Select the tab for the object type of interest.
2. In the list of objects, check the box to select the object you want to check for duplicates.
3. From the **More Actions** menu select **Find Duplicates**.
4. Select the desired criteria to find duplicate records: values of selected fields must be equal to field values in the current record.
5. Click **Find**.
6. If any records matching the selected criteria are found, select the records to be merged and then click "Merge".
7. Finally, select fields from all records to be merged into a final record, in the same way as you would use the merge records feature.

## Compare

Click the "Compare" option in the "More actions..." picklist in the View header or View each selected record side by side for comparison purposes. This is useful prior to merging duplicate records, and you have the option to merge after viewing the records.

Members: Compare Selected Records

Compare Selected Records		
Barry Smith	B Smith	Barri Smith
Check Out [R112444147]		
Check Out [R53356]		
Check Outs		
Communication Log		
Contact Owner		
Created At 07/13/2014 06:27 PM	07/13/2014 07:09 PM	07/13/2014 07:10 PM
Created By Adam Ministrator	Adam Ministrator	Adam Ministrator
Email Address		
Fax		
First Name Barry	B	Barri
ID 112448012	112448013	112448014
Last Name Smith	Smith	Smith
Members Barry Smith	B Smith	Barri Smith
Middle Name		
Mobile Phone		
Phone		
Title		
Updated At 07/13/2014 06:30 PM	07/13/2014 07:09 PM	07/13/2014 07:10 PM
Updated By Adam Ministrator	Adam Ministrator	Adam Ministrator
vCard File Synchronize	Synchronize	Synchronize
<b>Merge</b> <b>Cancel</b>		

## Finding Duplicates

Click the "Find Duplicates" option in the "More actions..." picklist in the View header to find records which may be considered as duplicates to selected group. This is two-steps process.

First, select text fields used to look for identical values (like Record name, Email etc).

Second, select radio button for record with duplicates. Select check boxes for duplicates you'd like to merge with selected record (all boxes are checked by default). Then click "Merge" to start merge process (see above) which will eliminate selected duplicates.

### Member: Merge Duplicates

Select Records to Merge

<input checked="" type="radio"/> B Smith	<input checked="" type="checkbox"/> B Smith	<input checked="" type="checkbox"/> Barri Smith	<input checked="" type="checkbox"/> Barry Smith
--	---	---	---

## Merging Records

In the records list, select the records to merge. Click the "Merge" option in the "More actions..." picklist in the View header to merge two or more records into one resulting record. The merge Page allows you to select which Field value from each selected record to use in the resulting single record.

Important: Merged records are moved to Recycle Bin without deleting dependent records or running triggers.

### Titles: Merge Selected Records

Merge Selected Records

Field The Night the Light Went Out for Georgie Author <input checked="" type="radio"/> Aamaras, D Publisher <input checked="" type="radio"/> Folkways Inc. Title <input checked="" type="radio"/> The Night the Light Went Out for Georgie	The Night the Lights Went Out for Georgie <input checked="" type="radio"/> Aamaras, D <input checked="" type="radio"/> The Night the Lights Went Out for Georgie
---	--

## Auditing

Progress Rollbase automatically creates audit trail entries for records to keep track of a history of changes to Object records and Object definitions. Audit trail records cannot be modified or deleted; they are only deleted when their related Object record is deleted.

To enable auditing on Object records, enable the **Audit Trail** attribute for the Object definition:

Optional Object Properties

Select the appropriate properties based on how you want records of this object type to behave:

Property	Description
<input checked="" type="checkbox"/> Audit Trail	Select this option to enable creation of Audit logs when values of selected fields are changes or by invoking triggers and API. In addition create Audit logs when record is: <input checked="" type="checkbox"/> Viewed <input checked="" type="checkbox"/> Created <input checked="" type="checkbox"/> Edited <input checked="" type="checkbox"/> Deleted

Audit trail entries are displayed in a list view component which is placed in the "System Info" tab by default in an object's View Page. You can move or remove it using the Page Editor.

Audit trail records are automatically created in the following cases:

- When a record is viewed and "Keep a log..." option is enabled.
- When a record is created or updated and the Object has the Audit Trail attribute enabled.
- When a record is deleted and the Object has the Audit Trail attribute enabled. In this case the audit trail entry will be recorded in the associated User's audit trail.
- When a Field has the "Track all changes..." attribute enabled and the content of the Field has been changed (in this case the audit trail record is created without regard to the Object level Audit Trail attribute).
- When an email related to an Object record is sent. In this case, the View link in the Action column allows viewing a copy of the email message that was sent.

You can create a Workflow Trigger of type "Create Activity Trail Record" to generate custom template-based audit trail entries when a record is created or updated.

Note: This audit trail entry is created when the email message is actually sent, typically with a small delay.

The following screenshot illustrates several types of Audit Trail entries associated with an Object record:

Audit Trail   Show All			
Action	Date/Time	Content	User
View	04/08/2010 09:37 AM	Email message "Address Info" sent to pvorobiev@rollbase.com	Pavel Vorobiev
	04/08/2010 09:34 AM	Custom Audit Trail Record created at 04/08/2010 09:32 AM	Pavel Vorobiev
	04/08/2010 09:34 AM	Address "Santa Rosa" has been updated.	Pavel Vorobiev
	04/08/2010 09:32 AM	Value of City field was changed from "Santa Clara" to "Santa Rosa".	Pavel Vorobiev
	04/08/2010 09:32 AM	Address "Santa Rosa" has been updated.	Pavel Vorobiev
	04/08/2010 09:31 AM	Address "Santa Clara" has been updated.	Pavel Vorobiev

The Audit Trail list view component shows the 20 most recent entries. Use the Show All link to see more (up to 100) entries in a pop-up window. You can also export all entries to XLS or CSV format from this window.

Audit Trail entries are also created when you make changes to an Object Definition or Application. You can see these records on the bottom of the Object View and Application View Pages.

Title: The Wind Billows

Back to List
Edit
Delete
More actions... ▾
Previous
Next

title Info System Info

System Information

Created By	Adam Ministrator	Created At	10/02/2014 03:11 PM
Updated By	Adam Ministrator	Updated At	10/02/2014 03:11 PM
Tags	ID 9026607		

Audit Trail Show All ○

Action	Date/Time ▾	Content	User
10/02/2014 03:27 PM		Viewed by Adam Ministrator	Adam Ministrator
10/02/2014 03:24 PM		Viewed by Adam Ministrator	Adam Ministrator
10/02/2014 03:11 PM		Title "The Wind Billows" has been created.	Adam Ministrator

Finally, some types of administrative Audit Trail entries related to a Customer in general can be found on the bottom of the Administration Setup Page. These records include, for example, a copy of email messages sent as result of large asynchronous data imports.

Audit Trail   Show All			
Action	Date/Time ▾	Content	User
View	04/08/2010 10:03 AM	Email message "183 Addresses have been created." sent to <a href="mailto:pv@rollbase.com">pv@rollbase.com</a>	Pavel Vorobiev
	03/25/2010 11:37 AM	Application Opus 2 deleted	Pavel Vorobiev
	01/26/2010 03:47 PM	Application Opus 1 deleted	Pavel Vorobiev
	01/16/2010 07:22 PM	Database creation completed	

## Send Email

Click the "Send Email" option in the "More actions..." picklist in the View header to send an email to all of the selected records. This option is only available for Objects with the Contact attribute.

The Send Email Page allows you to select a predefined email template (e.g., you can select an email template associated with the Object definition). The "Reply To" Field allows you to select the sender: either the current user's email address, or the "Default Email Sender" address specified in Setup > Administration Settings > Account Settings.

The "Send To" Field allows you to specify the recipient's email address. The default selection is "Record's address," which uses the email address Field in the Contact attribute. To change this behavior, select "Specified address" to specify one or more recipient email addresses (using merge Fields such as {!email} {!#CREATED\_BY.email}), as well as CC and BCC options. You can manually type in addresses in the "To", "CC" and "BCC" Fields if you select the "Specified address" option.

**Note:** You can enter several addresses separated by space, comma, or semicolon. You can use template tokens for Formula fields of String type which return one or more valid email addresses in "To", "CC" and "BCC" Fields. For information about using templates and formulas, see [Adding Business Logic](#) on page 151.

The top of the Page shows a list of recipients that can be used to verify that your selection is correct. Links at the bottom of the Page allow you to add up to three email attachments.

When sending emails to a group of recipients you can check "Do not send duplicate emails to the same address" option. This may be useful if you're sending a large number of emails which may have duplicate "to" addresses.

Note: Each recipient will receive a separate email. The Send Email option sends individual emails to each selected record, rather than one record with each recipient copied.

Important: Users often get confused about "From" and "Reply To" fields in Rollbase-generated email messages:

To change field "From" in Rollbase-generated email messages open Account Settings page and change "Default Email Sender". For information about advanced setup and administration, see [Advanced Setup and Administration](#) on page 421.

Field "Reply To" is typically current user's address (unless system address is selected). This field will be used by default if recipient hits "Reply" button.

These two fields are used because of spam rules employed almost everywhere.

## Changing the Owner of an Object

Click the "Transfer Owners" option in the "More actions..." picklist in the View header to quickly change the ownership of the selected records from one user to another. This option is only available if the Object definition has at least one relationship with the User Object.

The screenshot shows a 'Transfer Ownership' dialog box. At the top, there are 'Transfer' and 'Cancel' buttons. Below that, a section for 'Selected Records' shows 'Leads' with three names: Alexey BURDAKOV, Benjamin Horowitz, and Amoretta HOEBER. A note says 'Red = Required Information'. The main section is 'Define Transfer', which includes a 'New Owner' field with a search icon and a 'Fields To Transfer' section with a checkbox for 'Lead Owner'.

## Updating Multiple Records

You can create as many Mass Update pages as you need for each object definition. Each Mass Update page will appear as an option in the **More actions** menu. For more information on Mass Update pages, see [Pages, the Page Editor, and Grid Controls](#) on page 223.

Click the **Mass Update** option in the **More actions** menu to update all of the selected records at once. By default, this page contains no fields. Click **Design This Page** to add fields. By default, fields will not be modified unless the end-user enters data into them.

## Tagging Records

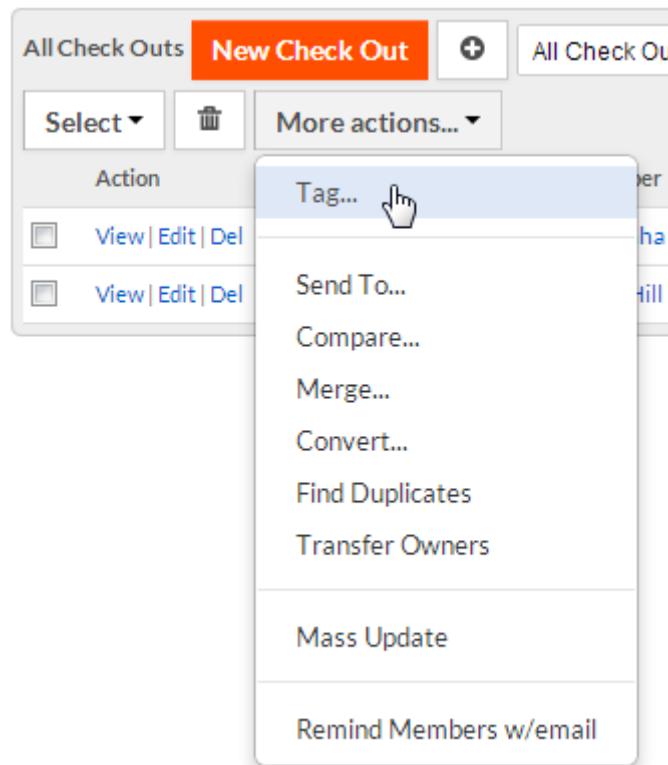
A tag is a label used to describe one or more records. Records tagged with keywords show up as matches for searches on any of those keywords. In addition, when you view a record with tags, you will be able to see all the tags affixed to that record, along with how many different users tagged that record with the same keywords. Clicking on a tag automatically performs a global text search for that keyword and displays the search results.

Tags must be enabled per object in the **Optional Object Properties** section of the object definition:

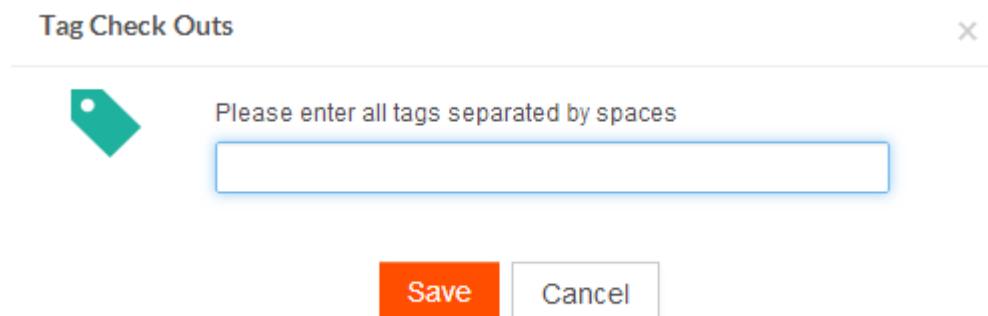
Optional Object Properties	
Select the appropriate properties based on how you want records of this object type to behave:	
Property	Description
<input checked="" type="checkbox"/> <b>Audit Trail</b>	Select this option to enable creation of Audit logs when values of selected fields are changes or by invoking triggers and API. In addition create Audit logs when record is: <input type="checkbox"/> Viewed <input checked="" type="checkbox"/> Created <input type="checkbox"/> Edited <input type="checkbox"/> Deleted
<input type="checkbox"/> <b>Flagging</b>	Select this option if you want the ability to flag records for follow-up. Flagging applies uniquely to each user.
<input type="checkbox"/> <b>Viewed Tracking</b>	Select this option if you want to visually differentiate records that have been viewed from those that haven't. Records that have not been viewed will be shown in bold. Viewed/Unviewed tracking applies uniquely to each user.
<input type="checkbox"/> <b>Dependent</b>	Select this option if you want to make this object a dependent object. As a dependent object, records of this type can only exist as related records. For example, consider objects representing Questions and Answers. Normally an Answer will not exist without a Question. In this case Answer is an example of a dependent object. Another example is Quotes and Line Items, where Line Items cannot exist independent of a Quote.
<input checked="" type="checkbox"/> <b>Show "Tag" Option</b>	Allow adding search tags to this object's records.
<input checked="" type="checkbox"/> <b>Reports Enabled</b>	Reports can be created on this object

To add tags to records from a view:

1. Select the records to tag.
2. From the **More Actions** dropdown list, select **Tag**:



The **Tag** dialog displays:



## Orphan Records

Orphan records control allows you to specify whether or not related records get deleted when the parent record is deleted. For example, if a user deletes a Quote, we may also want to delete all of the related Quote Line Item records. But the opposite case is not true; if a user deletes a Quote Line Item, we do not want to delete the related Quote record. Rollbase allows you to control this kind of behavior for each relationship.

**Orphan Records Control**  
 Orphan records control allows you to specify whether or not related records should be deleted whenever a record is deleted.

Delete all Lead records when related Company record is deleted  
 Delete all Company records when related Lead record is deleted

## Recurring Calendar Events and Tasks

You can replicate Calendar event and task records using a recurring date pattern. To do that on Record View page use "More Actions" drop down and select "Recurring Events" or "Recurring Tasks" option.

**Create Recurring Series of Events** Red = Required Information

<b>Meeting</b>	Staff Meeting
<b>Event Time</b>	09:00 AM
<b>Start Date</b>	1/1/2012 <input style="width: 20px; height: 15px;" type="button" value="..."/>
<b>End Date</b>	12/31/2013 <input style="width: 20px; height: 15px;" type="button" value="..."/>
<input type="checkbox"/> Run "On Create" triggers for newly created Meetings	
<b>Frequency</b>	<input type="radio"/> Daily <input checked="" type="radio"/> Weekly <input type="radio"/> Monthly <input type="radio"/> Yearly
Every <input type="text" value="1"/> weeks on <input type="checkbox"/> Sunday <input checked="" type="checkbox"/> Monday <input type="checkbox"/> Tuesday <input type="checkbox"/> Wednesday <input type="checkbox"/> Thursday <input type="checkbox"/> Friday <input type="checkbox"/> Saturday	
<input style="width: 80px;" type="button" value="Create"/> <input style="width: 80px;" type="button" value="Cancel"/>	

Choose Start and End dates, then select recurring options: Daily, Weekly, Monthly, or Yearly. The system will clone selected Calendar Event with recurrent dates according to selected pattern.

Important: Number of created recurring events cannot exceed 300 limit.

To view and manage recurring instances of Event or task: use Page Editor and add "Recurring Events" component to View page.

---

## Adding Business Logic

---

After an application contains the appropriate objects and relationships to capture data, you will likely want to add logic that uses or transforms that data. Rollbase supports this through templates and formulas. Templates allow you to define a structure for calculating and displaying information. Templates contain tokens, which are variables that the runtime replaces with values from the object currently in scope. Formulas are interpreted as Javascript expressions, they describe how to calculate values. If a template contains formulas, the runtime first parses the merge tokens to obtain the values. Once the values are obtained, the runtime evaluates the Javascript expressions using the appropriate values.

For example, suppose an object contains fields for quantity and price and you want to calculate the price of a particular quantity. Using the tokens, `{!quantity}` and `{!price}`, the runtime will handle them in a formula as follows:

- Formula: `{!quantity} * {!price}>`
- Parsed formula values evaluated as Javascript: `100.0 * 5`
- Result: 500

The following table describes the areas of a Rollbase application where you can use templates and formulas and why you might use them.

Templates	Where and How Used	Formulas	Where and How Used
Page HTML and Script components	Create in <b>Page editor</b>	Object formula fields	Object definition

Templates	Where and How Used	Formulas	Where and How Used
Object template fields and integration links	Create in Object definition	Trigger conditions and expressions	Part of trigger definition that determines when trigger will run or what value to use as a result
Object Record Name Template	Object definition to automatically generate record names	Workflow action conditions	Part of workflow definition that determines whether workflow should proceed
Mail Templates	Object and workflow definitions to dynamically generate e-mails	Field validation conditions	Object field definition
Document Templates	Workflows, template-based reports, and Document Template fields to dynamically generate documents	Values calculated in Gauge Components	Part of gauge definition to calculate values to render in the gauge
Report Templates	Report definition to dynamically generate reports	EVAL blocks	In non-formula templates, such as email or document, use to embed a formula

For details, see the following topics:

- [Working with Templates](#)
- [Formulas](#)
- [Triggers and Workflows](#)
- [Reports, Charts, and Gauges](#)
- [Multi-Currency Support](#)
- [Surveys and Quizzes](#)

## Working with Templates

Topics in this section describe how to work with templates.

## Adding Templates to a Page

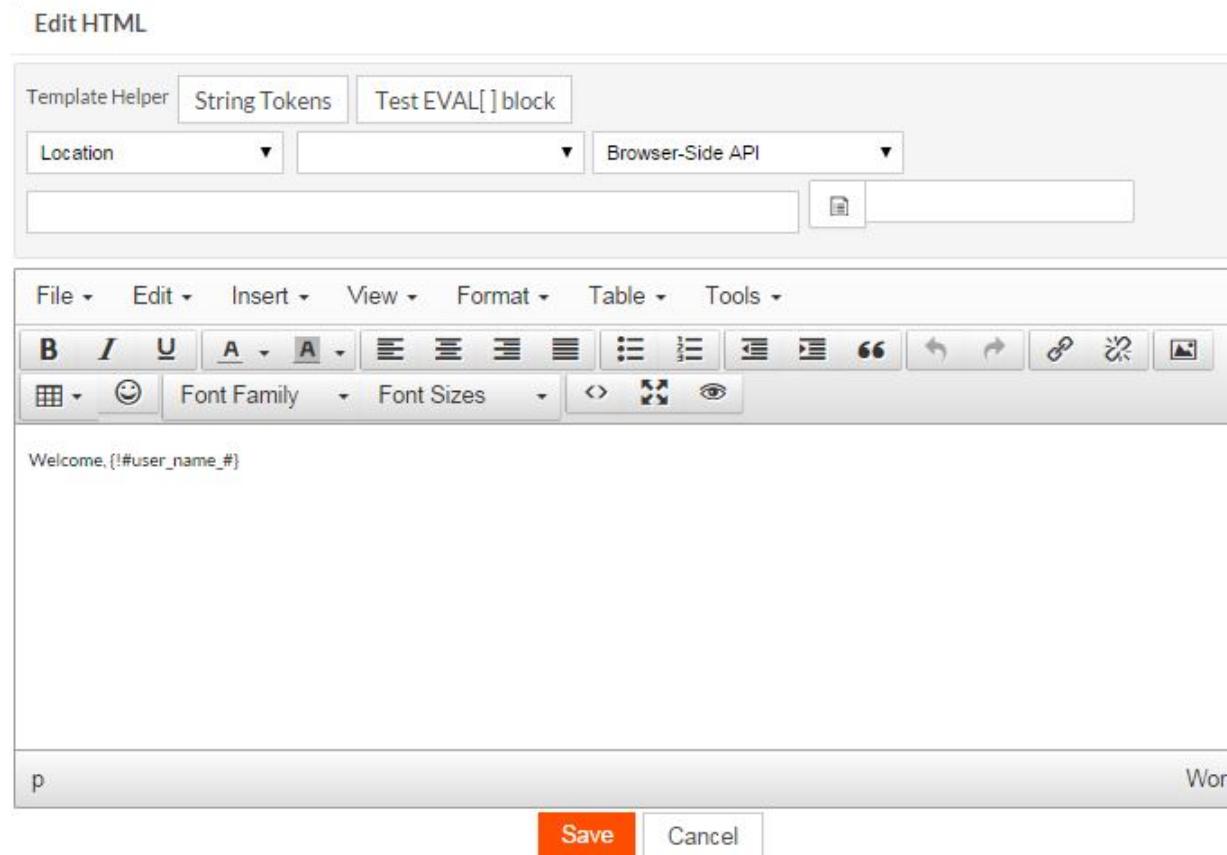
HTML and Script components allow you to add templates and formulas to a page. HTML will be rendered when the end-user views a page. Script components can contain Javascript. See [Adding Business Logic](#) on page 151 for more information about templates and formulas

## Adding HTML Components to a Page

HTML components allow you to customize the way a page displays. They are especially useful for portal pages. You can add formatting with stylesheets and HTML code. The **Template Helper** makes it easy to add data elements, perform calculations, or call Rollbase APIs to perform functions. To add Javascript formulas to an HTML component, insert them in `<script>` elements.

1. Navigate to the page you want to edit and click **Design This Page** to open the **Page Editor**.
2. If the page does not contain a section to hold an HTML component, drag a **New Section** component from the left pane onto the page.
3. In the left pane, select **New HTML Component** and drag and drop it on the page on the right pane.
4. Click **Edit** to display the **Edit HTML** page.
5. In the text box below the formatting bar, create the template for your page.
6. Use the **Template Helper** area of the editor to insert tokens that select values:
  - a) From the left drop-down menu, select the type that you want to retrieve values from.
  - b) From the second drop-down, select the field value of interest to display the token required to extract that value.
  - c) From the field below the drop-downs, select the token value and paste it in the template body.

For example, the following shows how to obtain user's name:



7. To select a particular instance of the values represented by a token, after selecting a merge token, follow these steps:

- a) Click the **Select ID**  icon to display the list of available IDs:

- b) Select the appropriate value, which will display in the **Template Helper**:

- c) Copy the ID and replace the #value portion of the token in your template body.

For example, using the example shown in the screen shot, a token of `{industry#77165406}` would be replaced with "Software" when the runtime parses the template.

8. Click **Save** to save the template and return to the page.
9. Save the page.

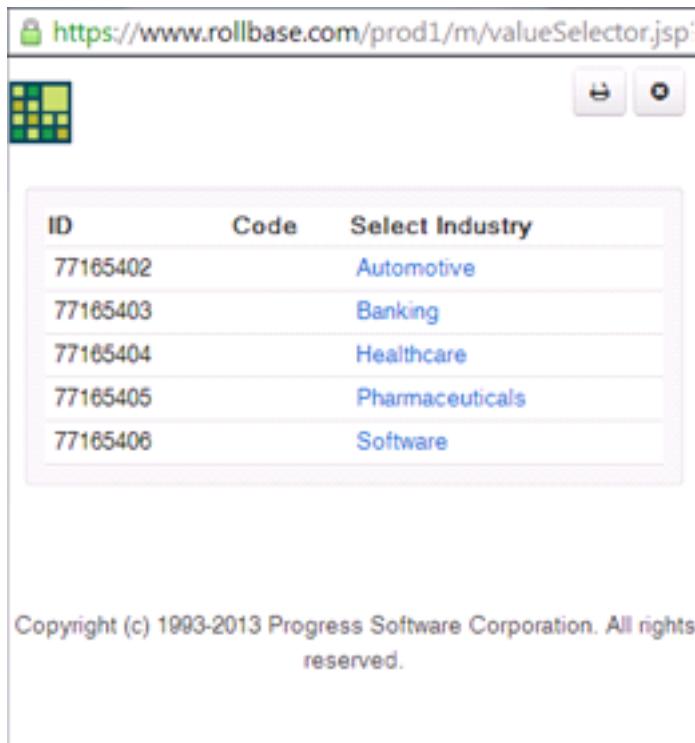
## Adding Script Components to a Page

Script components allow you to add Javascript formulas to a page. See [Adding Business Logic](#) on page 151 for more information about templates and formulas

1. Navigate to the page you want to edit and click **Design This Page** to open the **Page Editor**.
2. If the page does not contain a section to hold a script component, drag a **New Section** component from the left pane onto the page.
3. Add a script component by dragging and dropping it on the page.
4. Click **Edit** to display the **Edit Script** page.
5. Insert the script for your page.
6. Use the **Template Helper** area of the editor to insert tokens that select values:

- a) From the left drop-down menu, select the type that you want to retrieve values from.
  - b) From the second drop-down, select the field value of interest to display the token required to extract that value.
  - c) From the field below the drop-downs, select the token value and paste it in the template body.
7. To select a particular instance of the values represented by a token, after selecting a merge token, follow these steps:

- a) Click the **Select ID**  icon to display the list of available IDs:

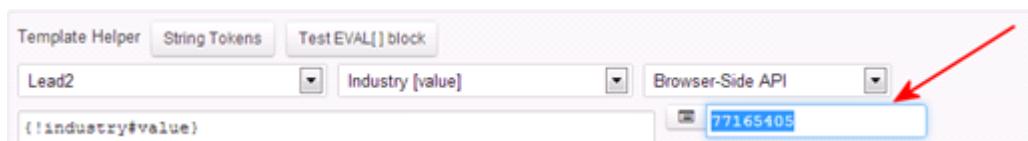


https://www.rollbase.com/prod1/m/valueSelector.jsp

ID	Code	Select Industry
77165402		Automotive
77165403		Banking
77165404		Healthcare
77165405		Pharmaceuticals
77165406		Software

Copyright (c) 1993-2013 Progress Software Corporation. All rights reserved.

- b) Select the appropriate value, which will display in the **Template Helper**:



Template Helper   String Tokens   Test EVAL[] block

Lead2   Industry [value]   Browser-Side API

{!industry#value}   77165405

- c) Copy the ID and replace the #value portion of the token in your template body.

For example, using the example shown in the screen shot, a token of `{industry#77165406}` would be replaced with "Software" when the runtime parses the template.

8. Click **Save** to save the template and return to the page.  
 9. Save the page.

## Adding Template Fields and Integration Links to an Object

Template Fields are similar to Template-based HTML components, but can be used in Views and Page layouts because they are part of an object definition. Integration Links require URL encoding. The Rollbase Template engine automatically selects the correct encoding rule, but you must enter URLs using the correct syntax. You can use Template fields in combination with other field types to deeply customize the Rollbase user interface (see the example below).

Template Fields have an option to hide the display label that is normally shown on the left in Pages. This feature is especially useful for creating a single piece of JavaScript code to be added to multiple Pages.

1. In an **Object definition** page, scroll down to the **Fields** section, click **New Field** and select **Template** or **Integration link** as the **Field type**.
2. If you chose **Template**, enter HTML for the field's body and embed any Template merge tokens as needed using the merge field helper.
3. If you chose **Integration link**, enter link URL.
4. Preview the resulting field by clicking the **Preview Template** or **Preview Link** link.

In this example, a related field on the Order Object points to the email address of a related User. This related field will display a link to send an email to that user and this is correct behavior. To add a link to send an email using Order fields to a different related email address, create a Template Field with the following HTML Template:

```
<a href='..../m/send.jsp?act=clean&id={!id}&objDefId={!#OBJ_ID.order}&to={!relatedEmail}'>{!relatedEmail}</a>
```

This Template will render a link to a "Send Email" Page using fields from the current Order record, but using the email address from the related field.

## Creating a Record Name Template

Every Object record in Rollbase has a special field labeled **Record Name**, which is used as a link to a record's view Page and in other cases when the record's name is displayed. By default this field allows direct user input. However, in many cases it is more convenient instead automatically populate this field's value from other fields using a **Record Name Template**.

If you create an Object with the "Contact" attribute, the following **Record Name Template** will be set by default: `{!firstName} {!lastName}`

If you want to use the value of a checkbox in a **Record Name Template**, use the `#EVAL[]` helper as shown here: `#EVAL[ {!club_member} ? "Member" : "Non-Member" ] {!title}` Important limitation: In **Record Name Template** you can only use non-dependent fields that have stored values for each record. Fields that brings values from other records (such as lookups or related fields) cannot be used in record name templates.

If your Object has an Auto-Number field (such as Invoice Number), it is often useful to use that field in the Record Name Template.

To set up a **Record Name Template**, follow these steps:

1. Edit the Object Definition.
2. Find the **Record Name Template** property.

3. Select one or more merge tokens and paste them into the **Record Name Template** field. Add separator characters as desired.
4. To update all existing records using the new Template, check the box below the Template called **Update names of all existing object records using this template**.

## Template Token Syntax

There is no need to type tokens manually, since the merge field helper allows you to select and copy all available tokens. However, you may find it beneficial to learn the syntax that can be quite useful in troubleshooting issues. Generally, a token has a form similar to:

```
{![Prefix].TokenName#[Suffix]}
```

- **Prefix** (optional) is the integration name for the relationship, if any. For example: R123456. For hierarchical relationships, it additionally includes #C for the child side of the relationship and #P for parent side of the relationship.
- **TokenName** (mandatory) represents the integration name of the field.
- **Suffix** (optional) includes additional instructions on how to render a given token.

**Important:** If you do not specify a suffix, the field is rendered as HTML for HTML Templates and plain text for text Templates. For example, in HTML Templates, a checkbox will be rendered as an image (a checked or unchecked box) and an object's name field will be represented by a link to a record view page.

Several unique merge tokens can be used to build links within Template components (use actual Object integration name instead of objName). For example, a URL generated by **#REPORT** token can accept filtering parameters:

- **filterName**: name of field to filter
- **filterValue**: value of field (only records with this value will be shown in the report)

Example: `{#!REPORT.123456}&filterName=R45678&filterValue={!id}`

All fields from the currently logged-in user and current Customer are available for use in Templates:

- `{#!CURR_USER.name}`: name of current user
- `{#!CURR_CUSTM.name}`: name of current customer

You can also use tokens for:

- Company-wide settings (see [Using Company-wide Settings](#) on page 424)
- Current portal user (for portal Pages)
- Shared images (from any Object)
- Helpers

The following list commonly used and advanced Template tokens:

## Common Tokens

The following tokens are commonly used in templates and formulas:

FIELD TYPE	SUFFIX	MEANING
Any type	#before	Field's value before update (for triggers, see below)
Any text field	#url	URL-encoded text (for use in URL templates)
Text field	#value	Plain text (ignore template-specific encoding)
Picklist, Status, or Role	#id	ID of field's value or comma-separated list of IDs
Picklist, Status, or Role	#value	String value of field or comma-separated list of string values
Picklist, Status, or Role	#code	Integration code of field's value or comma-separated list of integration codes
Text Area	#html	Raw HTML value (for rich-text text areas)
Text Area	#text	HTML-encoded text value
Lookup	#id	ID of first related record or -1 if no related records exist
Reference	#id	ID of referenced record or -1
Reference	#value	Object name of referenced record
Record Link (name field)	#text	Record's name as plain text
Record Link	#link	Link to View Page (for HTML Templates)
Record Link	#url	URL to View page
Checkbox	#value	"true" or "false"
Decimal, Currency, Percent	#value	Unformatted number
Date, Date/Time	#iso	Date or date/time formatted in ISO 8601 format. Example: 2011-10-13T
Date, Date/Time	#js	Date or date/time formatted in JavaScript format. Example: Thu Oct 13 2011 08:06:48 (PDT)

FIELD TYPE	SUFFIX	MEANING
Document Template	#id	ID of selected Template
Document Template	#link	Link to file generated from Template
Document Template	#url	URL to file generated from Template
Email Template	#id	Id of selected Template
Email Template	#link	Link to preview email Template
Email Template	#url	URL to preview email Template
Shared Image	#html	HTML with IMG tag (for HTML Templates)
Shared Image	#url	URL to image
File Upload	#html	Link to uploaded file
File Upload	#url	URL to uploaded file

## Advanced Tokens

The following tokens provide advanced functionality that you can use in templates and formulas. Rollbase replaces the token with the content described below:

Token	Will be replaced with ...
{ !#TODAY }	Today's date in the user-selected format
{ #SESSION_ID }	User's session ID
{ #PAGE_ID }	ID of current page
{ !#HOST_NAME }	Host name of the server running the Rollbase instance
{ !#LINK. <i>objName</i> }	Link to the object's view page assigned to current user
{ !#LINK. <i>objName</i> # <i>id</i> }	ID of the object's view page assigned to current user
{ !#LINK. <i>objName</i> #12345 }	URL to the object's page specified by the original ID (only available for view, new record, and edit pages and only if more than one page of each type exists (e.g. at least two view pages))

Token	Will be replaced with ...
{ !#LINK. <i>objName</i> #generic }	URL to the object's generic page assigned to current user
{ !#LINK. <i>objName</i> #template }	URL to the templates page
{ !#LINK. <i>objName</i> #import }	URL to the object's import page
{ !#LINK. <i>objName</i> #search }	URL to the object's search page
{ !#LINK.123456#tab }	URL to the generic or web tab specified by the original ID
{ !#OBJ_ID. <i>objName</i> }	Object definition ID for the specified object
{ !#CURR_USER. <i>fieldName</i> }	Value of the specified <i>fieldName</i> for the currently logged in User
{ !#CURR_CUSTM. <i>fieldName</i> }	Value of the specified <i>fieldName</i> for the current Customer
{ !#SETTINGS. <i>fieldName</i> }	Value of the specified <i>fieldName</i> for Company-wide Settings
{ !#ISV_VALUE. <i>fieldName</i> }	ISV customers only: Value of the specified <i>fieldName</i> for an ISV account. The fields must be marked as <b>Make this field available for related ISV customers</b> , see <a href="#">Using the ISV Partner Application</a> on page 554
{ !#PORTAL. 688851.#id }	ID of the portal with the specified original ID
{ !#PORTAL.688851.705908#id }	ID of the portal page with the specified original ID
{ !#PORTAL.688851.705908#url }	URL of the portal page with the specified original ID. This token can be used in the user interface.
{ !#PORTAL.688851.705908#link }	Link to the portal page with the specified original ID
{ !#PORTAL.688851.705908#emailUrl }	URL of the portal page with the specified original ID. This token can be used in emails.
{ !#PORTAL.688851.logout#url }	URL to the logout page for the portal with the specified original ID
{ !#PORTAL.688851.logout#link }	Link to the logout page for the portal with specified original ID
{ !#PORTAL.688851.emailUrl }	URL to Portal's main page which is safe to use in emails

Token	Will be replaced with ...
{ !#REPORT.123456#body }	The HTML body of a text-based report with the specified original ID
{ !#REPORT.123456#url }	The resource URL that generates the report with the specified original ID.
	<b>Note:</b> This is not available for tablular reports.
{ !#REPORT.123456#body? fieldName=fieldValue }	The HTML body of a text-based report filtered by a field value. The <i>fieldName</i> parameter specifies the integration name of a field by which to filter, <i>fieldValue</i> specifies the value on which to filter, such as <i>country=USA</i> .
{ !#FILTER }	HTML with content of current report filters. For template-based reports only.
{ !#CURR_VISIT.name }	Display name of the current portal user
{ !#CURR_VISIT.id }	ID of the current portal user
{ !#CURR_VISIT.loginName }	Login name of the current portal user
{ !#UID }	Unique string ID of objects imported from an external database or an OpenEdge Service.

## Iterating through Records

Templates can iterate through a list of records related to the current record. To accomplish this, include a portion of the Template between `#LOOP_BEGIN` and `#LOOP_END` tokens and that portion will be repeated for each record related to the current record.

**Note:** Having too many loops in Templates and Formulas may result in performance degradation, because Rollbase will retrieve many records from the database to compute the results. Use Template loops with caution. In many cases Group Functions or the Query API are better alternatives.

For example, the following Template code renders a list of related items:

```
<ul>{ !#LOOP_BEGIN.R8011504#8108944}
  <li>Item: { !R8011504.R8011504} Amount: { !R8011504.amount}</li>
{ !#LOOP_END.R8011504}
</ul>
```

To create a Template loop in the Template Editor:

1. In the left dropdown, select the Object type related to the current record
2. Select and copy the #LOOP\_BEGIN token.

If you have several Views for a related record, you will have a choice of the View to use for iteration that allows an easy way to filter and sort related records in Templates.

3. Create a portion of the Template to be repeated for each related record.
4. Select and copy the #LOOP\_END token.

---

**Note:** The #LOOP\_BEGIN token uses original ids of Views that will be preserved when your Template is published as a part of an Application and installed in another Tenant.

---

Rollbase does not support nested loops (loops within a loop). If you need nested loops consider using an API instead. For example, to use template tokens from parent record while looping through related record use object integration name (of parent record) as prefix, try the following:

```
<ul>
  { !#LOOP_BEGIN.R8011504#8108944 }
  <li>Bill: { !order.name } Amount: { !R8011504.amount }</li>
  { !#LOOP_END.R8011504 }
</ul>
```

## Loop Through Specific Number of Records, Comments, and Activity Trails

Optionally, you can include a maximum number of records to iterate through in parentheses at the end of the LOOP\_BEGIN token. This Template will iterate through the first 10 related records, ordered according to the selected View:

```
{ !#LOOP_BEGIN.R8011504#8108944(10) } ... { !#LOOP_END.R8011504 }
```

Using similar syntax you can loop through related Comments and Activity Trail records. The following example displays two last Activity Trail records:

```
<ul>{ !#LOOP_BEGIN.$ACT_TRAIL(2) }
  <li>{ !$ACT_TRAIL.content }</li>
{ !#LOOP_END.$ACT_TRAIL }</ul>
```

## Loop through All Records

You can loop through any records of a certain Object within a template, not just through related records. You must explicitly specify object name as prefix to template tokens used in loop through all records. For example, use { !order.name }, not the unqualified { !name }. You can limit the number of records to display by adding the (numRecords) token at the end of #LOOP\_BEGIN. If not specified, the number of records is limited to 100. Merge field helpers do not provide support for looping through all records.

To accomplish this, follow these steps:

1. Prepare a List View which sorts and filters records the way you desire (e.g., to render the most recent records and sort these records by "Updated At" in descending order).
2. Record the Original Id of that List View (you can find it on the Object definition's View page).
3. Use that Original Id in the Template:

```
{!#LOOP_BEGIN.all#originalViewId} ... {!#LOOP_END.all}
```

Use the Object's integration name as prefix for merge fields used inside loops through all records. Example:

```
{!#LOOP_BEGIN.all#479484}  
{!order.name}  
{!#LOOP_END.all}
```

Instead of {!name} here we use {!order.name} so that the merge field is replaced by the related record's name rather than that of the parent object record.

## Using EVAL Blocks

EVAL blocks allow you to use simple JavaScript-based expressions in a Rollbase Template. To debug an EVAL block in the **Template Helper**, click **EVAL[] Block**. Syntax for an EVAL block is as follows:

```
#EVAL [  
    JavaScript returning String ]
```

The following example loops through records but renders <img> tag only for records with amount > 1000:

```
<ul>  
{!#LOOP_BEGIN.R8011504#8108944}  
    <li>#{EVAL[ {!R8011504.amount}>1000 ? "<img src='important.gif'>" : "" ]}  
        Bill: {!order.name} Amount: {!R8011504.amount}</li>  
{!#LOOP_END.R8011504}  
<ul>
```

## Mail Templates

Mail Templates provide convenient way to generate title, body, and attachments for template-based emails manually from within Rollbase (for single record or group of records). Mail Templates can be used in/with:

- **Email Template** fields, see [Email Template Field](#) on page 841.
- **Send Email** workflow actions, see [Workflow Actions](#) on page 195
- Workflow triggers, see [Trigger Overview](#) on page 179

---

**Note:** Only users with access to the **Templates** menu can manage mail and other templates. See [Private Attribute](#) on page 398.

---

Template marked as "Private" have important limitations:

- They cannot be used in Triggers and Workflow Actions (since these components are shared by all users)

- They cannot be published as part of Application
- Once set, a **Private** flag can only be modified by an Administrative user.

The following screen shot shows an example of an email template definition:

## Creating an Email Template

To create an email Template:

1. Open the **Object definition** page (See [Viewing and Editing an Object Definition](#) on page 121).
2. From the object attributes ribbon, select **Templates** to navigate to the templates area of the page.
3. Click **New Mail Template** to define a new Mail Template.
4. Provide a display name for the Template.
5. Check the **Private** box if you want this Template to be hidden from other users (checked by default for non-admin users).
6. Select a Template **Format**: plain text or HTML.
7. Optionally, enter an **Integration Code** (used by API to find this template).
8. Specify a **Subject** line for this Template. The subject may include any tokens.
9. Specify the body for this Template. The body may include any Template merge field tokens and loops.
10. Click **Save**.

## Email Address Templates

When configuring Email Triggers, Email Actions, or sending a group of emails the system allows using template syntax to choose email's recipients: To, CC, BCC. Select available field of type "Email Address" from "Use Field" drop-down list. That will append selected field's token to "Send To" text box. You can re-arrange tokens or move them to another text box. Use space " ", as separator between tokens. Important: Email Addresses template only supports tokens for "Email Address" fields. Other tokens will be ignored. Remember: Single email will be sent and received by all email addresses specified in "Send To", CC or BCC box. If you wish to send individual email messages - do not specify multiple email addresses.



## Document Templates

You can create Rollbase templates by uploading text or binary documents. Document Templates can be used to generate documents such as Quotes, Purchase Orders, Invoices, etc. Supported document formats include:

- Microsoft Word (DOC and DOCX)
- RTF
- Microsoft Excel 2003 (XLS)
- CSV
- HTML
- XML
- PDF forms

- Plain text (.txt)

Document templates can be used in/with:

- Document Template** fields, see [Document Template Field](#) on page 841
- Template Document** workflow actions, see [Workflow Actions](#) on page 195
- Workflow **Triggers**, see [Change Workflow Status](#) on page 189
- Template-based **Reports** (see [Working with Reports](#) on page 204)

Text-based templates (TXT, CSV, and HTML) can be edited directly from Template Edit page without having uploaded new file. Note that template tokens must be considered as single, non-modifiable symbols. If you embed any formatting information (visible or invisible on the screen) inside the token, the parser will not be able recognize the token. Example: `{!name}` will not work and will be ignored. If you see that the parser cannot recognize your token (i.e. it does not get replaced), delete this token and type or paste it in again.

## Creating a New Document Template

Prepare a file in one of the supported formats which includes Template tokens and loops, as explained above. And then, create a new Document Template using the following steps:

- Open the **Object definition** page (See [Viewing and Editing an Object Definition](#) on page 121).
- From the object attributes ribbon, select **Templates** to navigate to the templates area of the page.
- Click **New Document Template** to define a new Document Template. The following page

Define Document Template

Template Name	<input type="text"/>
Integration Code	<input type="text"/>
Render as PDF	<input type="checkbox"/> (for HTML templates only)
Flatten	<input type="checkbox"/> (for PDF Forms only)
Populated Form	
<div style="border: 1px solid #ccc; padding: 5px; display: flex; justify-content: space-around;"> <span>Template Helper</span> <span>String Tokens</span> <span>Test EVAL[ ] block</span> </div> <div style="border: 1px solid #ccc; padding: 5px; display: flex; justify-content: space-around;"> <span>Meeting</span> <span><input type="button" value="▼"/></span> </div> <div style="border: 1px solid #ccc; padding: 5px; height: 40px; margin-top: 10px;"></div>	
<div style="display: flex; justify-content: space-between; align-items: center;"> <span>Upload File</span> <span><input type="button" value="Choose File"/></span> <span>No file chosen</span> </div> <p>Supported template formats: DOC, DOCX, XLS, HTML, RTF, CSV, XML, TXT, PDF</p>	

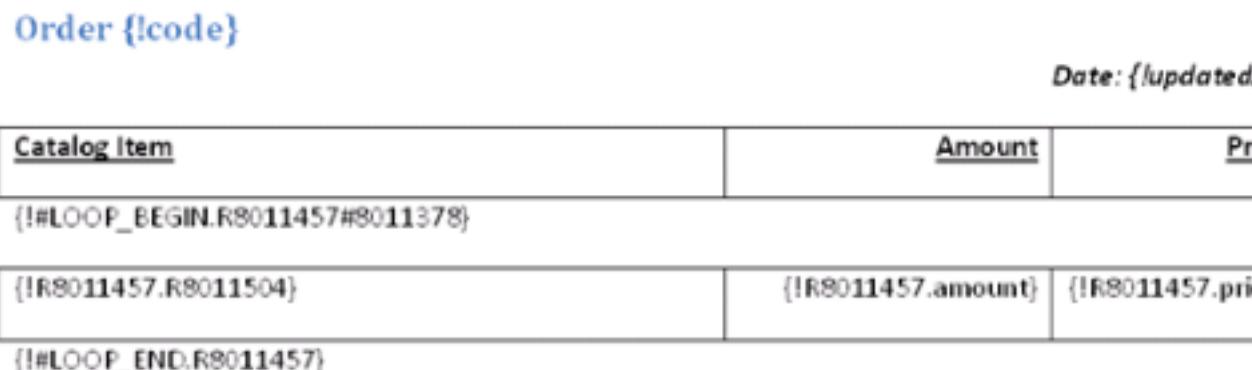
- Specify the following details:

- Template name:** Display name of the template.
- Integration code:** A code used by APIs to find this template. This is an optional field.
- Render as PDF:** For HTML Templates, select this option if you want the resulting HTML document to be converted into PDF format. Choose additional options for PDF document. For more information on PDF Report options, see [PDF Report Options](#) on page 209. This is an optional field.

- **Flatten Populated Form:** For PDF Templates, select this checkbox to make the resulting PDF non-editable. This is an optional field.
5. When modifying an existing text-based template (XML, TXT etc.) you have an option to edit text directly on web page rather than uploading a new file. For convenience the system displays:
- Helpers for available template tokens.
  - Preview for JavaScript Hosted files.
6. Click **Browse** to select and upload the prepared file to Rollbase.
7. Click **Save**.

## Microsoft Word Templates

You can upload Microsoft Word 97-2003 documents (with a DOC or a DOCX extension) and use them as Rollbase Document Templates. Important: On Private Cloud Rollbase servers document templates are only available when third-party software is purchased and installed (see [PDF Converter](#) on page 446). The screenshot below shows an example of a Word Document Template that includes tokens from a base record (Order) that utilizes different styles and includes a loop through related items. The body of the loop represents a Word table with a single row:



The screenshot shows a Microsoft Word document template. At the top, it says "Order {code}" and "Date: {updated}". Below this is a table with three columns: "Catalog Item", "Amount", and "Price". The "Catalog Item" column contains the token "{!#LOOP-BEGIN.R8011457#8011378}". The "Amount" column contains the token "{!R8011457.amount}" and the "Price" column contains the token "{!R8011457.price}". Below the table, there is a closing token: "{!#LOOP-END.R8011457}".

Catalog Item	Amount	Price
{!#LOOP-BEGIN.R8011457#8011378}		
{!R8011457.R8011504}	{!R8011457.amount}	{!R8011457.price}
{!#LOOP-END.R8011457}		

The screenshot below shows the parsed Template populated with real data. You can see that original tokens have been replaced with actual values and the styles of the Word document have been preserved. The Date field uses the Date format selected by the current user. The loop through related records now includes a table row for each related record (Catalog Item).

## Order O-2010-00002

Date: 01/28/2010 01:19 P

Catalog Item	Amount	Price
iPhone	100	\$300.
Office Chair	78	\$150.

## Writable PDF Forms

**Note:** For Private Cloud users, PDF templates are only available when the appropriate software is purchased and installed (see [Third Party Software You Can Install](#) on page 446).

PDF templates enable you to use application data to fill fields in writable PDF Forms. You first need to upload the writable PDF form and then map form fields to object fields of simple expressions. Use the **Map Form Fields** button (available only for PDF Document Templates) to bring up the mapping page. The example below shows mappings for an Object with Location and Contact attributes and PDF fields in standard IRS W-9 form. Please note that one field is mapped to a single token `{!streetAddr1}` while another is mapped to a string template: `{!city},  
{!state#code} {!zip}`. It is not always easy to figure out how particular PDF field should be used: their names may not be informative. We recommend that you first map all fields to easily recognizable tokens like "A", "B" etc. than preview result of mapping. In addition to Preview mode for resulting document PDF templates offer preview mode for fields mapping.

**Note:** A writable PDF form will remain writable after data is populated. Select the **Flatten Populated Form** checkbox on the **Edit Template** dialog to make the resulting PDF non-editable.

Map PDF Form Fields

Template Helper | String Tokens

Select Token Group	Select Merge Token	Copy Merge Token
Person		
topmostSubform[0].Page1[0].Address[0].f1_04_0_[0]	{!streetAddr1}	
topmostSubform[0].Page1[0].Address[0].f1_05_0_[0]	{!lcity}, {!state#code} {!lzip}	
topmostSubform[0].Page1[0].c1_01[0]		
topmostSubform[0].Page1[0].EmployerIdentif[0].f1_10[0]		
topmostSubform[0].Page1[0].EmployerIdentif[0].f1_11[0]		
topmostSubform[0].Page1[0].f1_01_0_[0]	{!name#text}	

The following was populated from a sample record:

**W-9**

Form (Rev. December 2011)  
Department of the Treasury  
Internal Revenue Service

**Request for Taxpayer Identification Number and Certification**

Give Form to the requester. Do not send to the IRS.

Name (as shown on your income tax return):  
John Smith

Business name/dissociated entity name, if different from above

Check appropriate box for federal tax classification:  
 Individual/sole proprietor    C Corporation    S Corporation    Partnership    Trust/estate  
 Limited liability company. Enter the tax classification (C=C corporation, S=S corporation, P=partnership) ►  Exempt payee  
 Other (see instructions) ►

Address (number, street, and apt. or suite no.):  
123 Broadway

City, state, and ZIP code:  
Middletown, AL 95678

List account number(s) here (optional):

Requester's name and address (optional):

Print or type  
See Specific Instructions on page 2.

## Communication Logs

Communication Log records are used to keep track of email messages, phone calls and other forms of communication. Unlike Audit Trail entries, you can add Fields to the Communication Log object definition and modify its Pages and Views.

[Communication Log](#) | [Send An Email](#) | [New Communication Log](#)

View (Related Records): [All Communication Logs](#) | [Edit View](#) | [New View](#) | [Clone](#)

Subscribe: [RSS](#) | [Export XLS](#) | [CSV](#) | [Google](#)

[Tag](#) | [Delete](#) | [More actions...](#) | [Select...](#)

Communication Logs 1-1 of 1

Action	Type	Subject	To	Updated At	Updated By
<a href="#">Edit</a>   <a href="#">Del</a>	Email	Test Message	tbulba@yahoo.com	04/08/2010 10:10 AM	Pavel Vorobiev

When an object with "Contact" attribute is enabled on new or existing object the system automatically creates relationship with Communication Log and adds list of related Communication Log records to object's View page. Otherwise you can create a relationship with a Communication Log object using the **New Relationship** link. For more information on relationships, see [Relationships Between Objects](#) on page 132.

Communication Log records can be edited or deleted if the User attempting to do so has sufficient permissions. Click the "Send Email" link to send a template-based email, with the email address pre-populated with the address of the base record. Click "New Communication Log" to create a new log record for a phone call, conversation, or other documented communication.

Please note the following:

- The "New Communication Log" page must include hidden field "Related To". Do not remove this field from page.
- The relatedTo field must be populated with valid parent record ID when creating Communication Log record through.
- Incoming gmail messages can be converted into Communication Log (see [Incoming Gmail](#) on page 367).

## Localization

Rollbase provides full support for localization, including various date formats, currency formats and multi-lingual support.

## New Record Template

In cases where users will need to create similar records repeatedly on a regular basis, you can use the Record Templates feature to streamline this process. New Record templates are available from the Templates menu for a particular Object definition, or from an Object definition's setup page.

To create a New Record template, define its name and select an existing record to be cloned.

### Account: New Record Template

#### Define Record Template

Template Name

Record to Clone

If you have defined one or more New Record templates for your Object definition, a drop-down list of these templates is displayed on the New Page for that object definition. Selecting one of them will initiate cloning of the record using the selected New Record template.

You can also use New Record templates in Create Related Record workflow actions. For information on workflows, see [Workflow Overview](#) on page 194.

# Formulas

Formulas are text templates sent to the JavaScript engine after parsing (i.e. after all merge field tokens have been replaced with real data). The JavaScript engine executes the formula and returns a single value to the caller.

## Simple Formula Example

This simple formula represents an expression:

```
{ !amount}*{ !price}*(1-{ !discount_proc}/100)
```

In more complex cases, you can define intermediate variables, flow-control operators, etc. For example, the following formula calculates monthly payments on a mortgage:

```
var q = { !mortgage_rate}/12/100; var A = parseInt('{ !amount}',10); if (A<=0
|| q<=0)
    return null; var N= parseInt('{ !loan_type#code}',10); if (N<=0) return A*q;
return
    A*q/(1+Math.pow(1+q, -N));
```

In such a case, the formula's body will be wrapped into a JavaScript function and the resultant function will represent the formula's result.

## Including Your Own Functions in a Formula

You can include your own functions in a formula:

```
function formatNum(x) {
    if (x instanceof Number && !isNaN(x))
        return x.toFixed(2);
    return "";
}

function calcStr() {
    var y={ !amount}*{ !discount};
    return formatNum(y);
}

calcStr();
```

Important: If you are using custom-defined functions, your Formula cannot be automatically wrapped and hence cannot include a return statement outside of a function. In this case place exactly one function call at the end of your Formula as shown above. The result of this call will be used as the result of the entire Formula. The Formula's resulting value must be of a certain type that depends on where the Formula is used:

## Creating and Processing Errors

To interrupt normal flow of JavaScript execution you can throw exceptions. An error message will be displayed on the screen if UI is involved. For example:

```
if ({ !update_blocked})
    throw "Update is blocked";
```

On the other side if you wish to process errors generated by part of your formula (for instance, Server-side API) you can use try / catch block. Example:

```
try {
    rbv_api.update(...); // Call which may throw Exception
}
catch (e) {
```

```
// Process Exception without terminating further execution
}
```

## Writing and Debugging Formulas

In the Formula Helper section, you can select a Template token and paste it into the Formula's body in the same way as done with Templates. The Select ID control can be used for picklists, lookup and status fields when you want to select an ID or integration code for use in your Formula. Loops through related records can be used in Formulas in the same way as in Templates. The loop's body will be replicated during parsing using data from each related record. Click the Validate Formula button for quick validation of the Formula's syntax. This will process the Formula on an arbitrarily selected Object record. For Formula validation and debugging, you must have at least one Object record created.

---

**Note:** Formula validation only checks JavaScript syntax validity. For more detailed debugging, use the Debug Formula button.

---

Click the Debug Formula button for detailed Formula debugging. Select the Object record to run the formula on in the Lookup window. As illustrated in the following screen shot, the debugger shows the following information:

- Original formula
- Formula parsed using the selected record's data (with line numbers on the left for convenience)
- Debugging output (if any)
- Result of Formula calculation
- Error message (if any)

If a particular line in parsed formula is causing an error that line will be highlighted.

## Original Formula

```
if (={!total} > 1000)
    return {!total}*{!discountLarge}/100;
else
    return {!total}*{!discountSmall}/100;
```

## Parsed Formula

```
function wrapper() {
if (1200.0 > 1000)
    return 1200.0*12.0/100;
else
    return 1200.0*10.0/100;
}
wrapper();
```

Formula return type: **Currency**

## Result

144

**Note:** For simple operations like SUM and COUNT, looping through related records should be replaced by much more efficient Group Functions as described in [Group Functions](#) on page 177

## Formula Return Types

The return type of a formula must be appropriate for its context. The following table outlines the return types of the formulas:

Type of Formula	Return Type
Formula fields created as part of an Object definition Selected by user per field. Can be one of the following:	Decimal Currency Integer String Boolean Date Date/Time
Conditions used in triggers and workflow actions	Boolean
Expressions used in triggers to change a field's value	Same as target field's type
Conditions used in field-level validations	String (error message) or null (no error)
Values calculated in gauge components	Numeric
#EVAL[ ] helpers that may be embedded in any Template	Any, will be converted into a string

## Examples of Valid String Tokens

String tokens such as `{!lastName}` will be replaced by values from text fields before sending the Formula to the JavaScript engine. Unlike numeric tokens that can be used in the Formulas as is, text tokens typically must be enclosed in quotes to produce meaningful results. Using an example for a record where Ellison is the last name and John is the first, the following table shows valid ways of using the name in a formula.

Token Usage	JavaScript	Results
<code>{!amount}*{!counter}</code>	<code>20.5*2</code>	Valid JavaScript expression
<code>{!lastName} Ellison</code>		Since there is no variable "Ellison" this will cause a JavaScript error
<code>"{!lastName}"</code>	<code>"Ellison"</code>	Valid JavaScript expression
<code>"{!lastName}, {!firstName}"</code>	<code>"Ellison, John"</code>	Valid and efficient concatenation of two tokens
<code>"{!lastName}"+ ", "+"{!firstName}"</code>	<code>"Ellison, John"</code>	Valid, but an inefficient way to concatenate tokens - see efficient example above

## Using Dates in Formulas

For the Date return type, Formulas can use the JavaScript Date class. Formulas also can optionally return the number of milliseconds between midnight of January 1, 1970 and the specified date. Typically you will create an instance of JavaScript Date class and call `getTime()` on it. The following Formula returns the current date shifted forward by 24 hours (calculated using the current user's time zone):

```
var d = new Date(rbv_api.getCurrentDate());
return d.getTime() + 24*60*60*1000;
```

---

**Note:** Avoid using the default constructor `new Date()` since it will return date in server's time zone which may be very confusing. `Date("{!date_field}")` will create a Date Object corresponding to your date field

---

### Example of Date Usage in Formulas

You can use the standard JavaScript Object Date in formulas. The following Formula shows how to calculate the difference in days between the Rollbase Date field and the current date taking into account the user's time zone setting:

```
var dt = new
  Date("{!date_field}");
var today = new Date(rbv_api.getCurrentDate());
var day = 24*60*60*1000;
// Length of 24 hours on milliseconds
var days1 = Math.floor(dt.getTime()/day);
// Rounded down
```

```
integer var days2 = Math.floor(today.getTime()/day);
return days1 - days2;
```

---

**Note:** String representations of date fields in Formulas (e.g. the merge field `{!date_field}` above) automatically uses the correct time zone from the current user. To ensure that current date uses the correct time zone as well, use the `getCurrentDate()` API.

---

## Example Using Images to Represent Record Status

To create an HTML icon with an image that reflects the current status of a particular record, follow these steps:

1. Create several **Shared Icon** fields to hold your icons.
2. Create a Formula field with return type `string` and the following formula:

```
if ("{!status#code}"=="CRE") return "{!order.created_icon#html}";
else if
    ("{!status#code}"=="SHI") return "{!order.shipped_icon#html}";
else if
    ("{!status#code}"=="CAN")
return "{!order.cancelled_icon#html}"; return "";
```

3. Add this formula field to Views and View Pages as needed. It will generate HTML with an image field which depends on the record's status

---

**Note:** Please note that the formula above uses status integration codes rather than IDs. This approach ensures that the Formula above can be published as part of application and installed without requiring changes. Never rely on IDs in formulas if you plan on publishing your application.

---

## Formula Execution Limits

You can use any valid JavaScript code in your formulas, including arrays, for loops, etc. However, to protect our computational resources, the total execution time of any formula is limited to 3000 ms. If server-side script takes too long to execute, it will be aborted by the system. The length of a parsed server-side script cannot exceed 10KB. This is normally not an issue and can only become an issue if you're using loops through a long list of records. Try to avoid using long loops.

---

**Note:** Rollbase Private Cloud customers may change the limits specified above. For information about configuring Private Cloud, see [shared.properties](#) on page 505.

---

For obvious reasons, server-side formulas cannot make use of the Document Object Model (DOM) or third-party JavaScript libraries. However, core JavaScript Objects (Math, String and Date) are available for server-side scripting. If a formula or template is used within another formula or template, it will be evaluated before being used. However, this useful feature can result in endless recursion (consider Formula field `my_formula` with body `{!my_formula}+1`). To prevent endless recursion, the system limits maximum recursion level (number of times formula is called from within another formula) to 10.

## Group Functions

Group functions are used to calculate the value of an expression from groups of related records. Rollbase Formulas support four types of group functions:

Function	Result	Parameters
<code>#CALC_SUM.R8011457( expression   condition )</code>	SUM of expressions for records where condition is true.	Expression is mandatory and must be numeric. Condition is optional (default to true).
<code>#CALC_COUNT.R8011457( expression   condition )</code>	Counts all records for which expression is not null and condition is true.	Expression is optional (default to 1) and must be numeric. Condition is optional (default to true).
<code>#CALC_MAX.R8011457( expression   condition )</code>	MAX of expressions for records where condition is true.	Expression is mandatory and must be numeric. Condition is optional (default to true).
<code>#CALC_MIN.R8011457( expression   condition )</code>	MIN of expressions for records where condition is true.	Expression is mandatory and must be numeric. Condition is optional (default to true).

You can also run Group Functions on the entire set of an Object's records. For that use the Object's integration name instead of a specific Relationship name, such as:

```
#CALC_SUM.invoice( amount | true )
```

When writing expressions and conditions for Group Functions do not use tokens from the Select Merge Token box; rather, use the Group Token box for fields from related records.

**Note:** Inside a group functions' body, you must not use tokens in `{! . . }` format or equate hard-coded numeric or string values with the special tokens. For example, the TODAY token specifies the current time, but you cannot get yesterday's time using `(TODAY-1)`. This results in an error.

Group functions use SQL syntax rather than JavaScript syntax for expressions and conditions. In simple cases you may not notice a difference. For Group Function conditions you can use the following special tokens:

- TODAY for the current time
- WEEK for 12PM of last Sunday
- MONTH for 12PM of 1st day of the current month
- QUARTER for 12PM of 1st day of the current quarter
- YEAR for 12PM of 1st day of the current year
- CURR\_USER for id of the currently logged in user

You can also use integration codes from picklists and status fields. Examples of Group Functions:

- Maximum value of amount field among related records created in current quarter:  
#CALC\_MAX.R8011457( amount | createdAt>=QUARTER )
- Count number of related records with address1 field not null: #CALC\_COUNT.R8011457( address1 )
- Sum amount field for all invoice records with due\_date after January 1st current year:  
#CALC\_SUM.invoice( amount | due\_date>=YEAR );

## Typical Mistakes in Formulas

The following sections summarize some typical mistakes and inefficiencies in formulas.

### Include Tokens in Quotes

Do not forget that Template tokens like {!name} are not JavaScript variables. They are replaced by actual values before being sent to the JavaScript engine. While numeric values can be used as is, strings are typically enclosed in quotes to form a meaningful result.

Original Formula	Parsed Results	Comments
counter += {!amount}; counter += 100;	Correct	You can treat numerical tokens as variables
if ({!country#code} == "US") If (CA == "US")	JavaScript error	CA variable does not exist
if ("{!country#code}" == "US") If ("CA" == "US")	Correct	You must always enclose string tokens in quotes or make them a part of larger string.

### Unnecessary Quotes

You can often use the fact that Template tokens will be replaced with real, non-quoted values to your advantage by simplifying token concatenation. You don't have to use JavaScript to concatenate tokens - the Template parser can do it for you.

Formula	Comments
var ln = "{!lastName}"; var fn = "{!firstName}"; return ln+", "+fn; var ln = "Ellison"; var fn = "John"; return ln+", "+fn;	Correct but inefficient
return "{!lastName}, {!firstName}"; return "Ellison, John";	Correct and efficient

### Avoid Loops in Formulas

Although you can use Template loops in Formulas, be aware that they can result in very long JavaScript after parsing; that script could be aborted by Rollbase due to Formula length and execution time limitations. Consider this example:

```
{!#LOOP_BEGIN.R12345}
// Do some processing
```

```
rbv_api.setFieldValue(...);  
{ !#LOOP_END.R12345}
```

The code inside the loop will be replicated for each record in the loop. A much more efficient approach would be to use a JavaScript for() loop:

```
var ids = rbv_api.getRelatedIds("R123456", {!id});  
for (var i=0; i<ids.length; i++) {  
  var ordered = ids[i];  
  rbv_api.runTrigger("order", orderId, "trUpdate");  
}
```

In this example, the Formula obtains a list of related IDs, loops through them all and invokes a trigger on the corresponding related record. You may also use Query API to extract data instead of loops through data records. Warning: If a number record in a loop is large you may hit timeout limit for formula execution.

## Using Comments

You can use valid JavaScript comments in your formulas. However please keep in mind that:

- Comments enlarge total length of formula and may potentially lead to hitting limit on overall formula length.
- Avoid using // -style comments since they may lead to error should carriage-return in your formula accidentally be lost. Use /\* \*/ style for comments in JavaScript formulas.

# Triggers and Workflows

Rollbase triggers provide a way to add user-driven and programmatic business logic based on record changes, or at scheduled times. Workflows support a structured user-driven flow of records through the user interface.

To use workflows, you must first enable the **Workflow** attribute. While triggers are often used together with workflows, they do not have to be used with workflows and the **Workflow** attribute does not have to be enabled on an object to use triggers.

Watch [Web App Trigger Examples](#) to see how triggers can update fields in related records and create new records.

---

**Note:** In addition to the built-in trigger types explained in this documentation, Rollbase Private Cloud users can develop custom Triggers using Java code.

---

## Trigger Overview

Triggers can perform automated validation, notification, and data manipulation. You can configure triggers to fire upon events such as record creation, update and deletion; as a result of a workflow flow action; and they can be run manually.

The triggers available for an object depend on that object's properties and components. For example, some are only available when object attributes such as workflow or audit trail are enabled. Others depend on components that you must create first, such as a conversion map or template. The timing options available also depend on the type of trigger you are creating.

Triggers are associated with an object definition. You can view or create new triggers from the **Triggers** component table. Depending on the type of trigger, its formula can calculate and set values, or specify a condition that causes the trigger to fire.

To conditionalize a trigger, specify a formula that returns a boolean value. If the value evaluates to false or null for a particular record, the trigger will not run on that record.

For example, to run triggers only for large orders, you could use a formula such as:

```
{ !amount } > 10000
```

A single line of JavaScript such as that shown above that evaluates to true, false, or null does not require a return statement. However, multi-line conditions such as those shown in other examples do require a return statement.

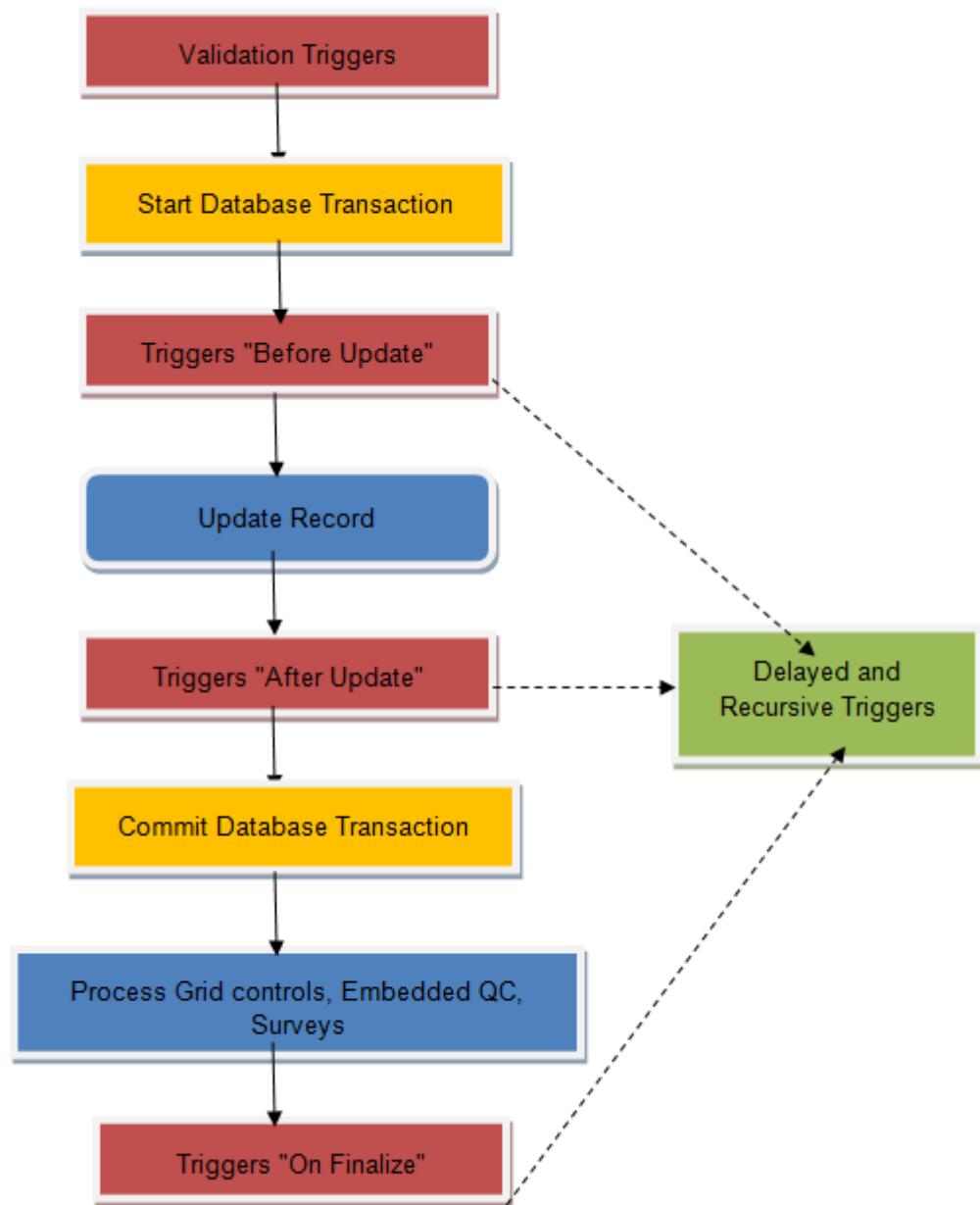
When defining a trigger you can also specify:

- Whether or not the trigger is deployed. Un-deployed triggers will not run.
- An integration name, which is good practice and allows you to invoke the trigger using the [rbv\\_api.runTrigger\(\)](#) on page 585 method.
- Whether to run a trigger only when a particular field has changed its value (this will only affect **After Update** triggers). You can use template and formula fields in this condition, allowing you to determine whether or not the trigger should fire based on changes to a group of fields instead of a single field.
- If a trigger performs an update of an existing record or creates a new record, you can specify whether dependent triggers (before/after update, before/after create) should also run. Important: Use this option with caution, since it may lead to inefficient nested trigger invocations.

Built-in triggers provide the following capabilities, which are described further in the linked content:

- [Send Email](#) on page 186
- [Create Audit Trail Record](#) on page 187
- [Validate Record Data](#) on page 187
- [Unique Fields Combination](#) on page 187
- [Update Field Value](#) on page 187
- [Change Workflow Status](#) on page 189
- [Create New Record](#) on page 189
- [Attach Related Record](#) on page 189
- [Create Template Document](#) on page 190
- [Run Triggers on Related Record](#) on page 190
- [Object Script](#) on page 190
- [HTTP Triggers](#) on page 191 Send **GET**, **POST**, or **SMS** messages

The graphic below illustrates the general order in which Rollbase runs triggers when you specify a timing option. See [Trigger Timing Options](#) on page 182 for more details.



The following topics provide guidance on using triggers and the general steps for creating them:

- [Trigger Rules and Restrictions](#) on page 182
- [Trigger Timing Options](#) on page 182
- [Best Practices for Trigger Formulas](#) on page 183
- [Delayed and Repeating Triggers](#) on page 184
- [Creating a Trigger](#) on page 185
- [Example: Set Field Based on a Workflow Status Change](#) on page 193
- [Debugging Complex Triggers](#) on page 193
- [Debugging Delayed Triggers](#) on page 194

## Trigger Rules and Restrictions

Triggers can work in combination. One trigger might invoke or rely on the results of others. The following rules ensure that dependencies between triggers do not result in endless recursion:

1. The total number of triggers invoked in a single update, create, or delete operation is limited to 100 for immediate triggers and 20 for delayed Triggers (these numbers may vary for Private Cloud customers).
2. Per triggering event, the same group of triggers, for example those timed to occur after an update, will not run twice on the same record.
3. Per triggering event, each trigger only runs once on each record.
4. Total execution time for a group of triggers, such as those timed to occur on create, cannot exceed 30 seconds (this limit may vary for Private Cloud customers). Time of HTTP calls (POST and GET) is excluded from this limit.

Use the Trigger Debugger to verify that grouped triggers work as you expect. See [Debugging Complex Triggers](#) on page 193 for more details.

## Trigger Timing Options

When you create or configure a trigger, you can optionally select when you want it to run during the lifecycle of a record. The interface only displays valid options for the type of trigger you are creating. Timing options that fire before a record is committed, such as a trigger to validate data, are different for those available for triggers that run after commit. In some cases, options show but are disabled. For example in a new record trigger, all timing options are shown but, **Before Create** and **After Delete** are disabled, because they are not applicable when creating a new record.

Multiple triggers can have the same timing option. In this case, the order in which they run is determined by the order in which they appear in the object definition **Triggers** table. The following table briefly describes each **Timing Option**:

Timing Option	When Run
<b>Before Create</b>	Before record is created
<b>After Create</b>	After record is created (most commonly used timing)

Timing Option	When Run
<b>Before Update</b>	Before record is created
<b>After Update</b>	Before record is updated (most commonly used timing)
<b>Before Delete</b>	Before record is deleted
<b>After Delete</b>	After record is deleted
<b>On Finalize</b>	After input from grid controls, embedded quick create, and survey components (if present on UI page) are processed. This option only applies when an individual record is created or updated on a UI page, not through an API.
<b>On Login</b>	When a user or portal user logs in
<b>On Logout</b>	When a user or portal user logs out

When a record is created, Rollbase:

1. Creates a new record without committing it into database.
2. Runs all triggers with **Before Create** timing.
3. Creates a new record in the database.
4. Runs all triggers with **After Create** timing.

When a record is updated, Rollbase:

1. Runs all triggers with **Before Update** timing (please note that record is not updated at this point of time).
2. Performs the actual record update in the database.
3. Runs all triggers with **After Update** timing. When a record is attached or detached using UI links in a related list component, the system runs **After Update** triggers on both sides of relationship.

When a record is deleted, Rollbase:

1. Runs all triggers with **Before Delete** timing.
2. Moves the record to Recycle Bin.
3. Runs all triggers with **After Delete** timing.

## Best Practices for Trigger Formulas

Triggers use formulas to calculate conditions, change field values, or perform other logic. To write efficient formulas, follow these suggestions:

1. Use the `return` keyword only in complex formulas with intermediate variables:

- Efficient:

```
({!amount} > 1000)
```

- Inefficient:

```
if ({!amount} > 1000)
    return true;
else
    return false;
```

2. Avoid looping through related records. Use group functions instead (looping through related records can seriously affect performance).
3. Always use integration codes instead of IDs for status and picklist items (this applies to template fields and formula fields as well). Logic based on IDs will not work correctly in applications published to other tenants because the IDs change.

## Delayed and Repeating Triggers

If you would like a trigger to be delayed relative to a particular date and time, you can specify delay criteria.

Note:

A trigger can be delayed relative to the current time or value of any date or date/time field for the selected record. The example above shows a trigger that will run seven days after the last record's update. You can also specify criteria to run a particular trigger periodically, including maximum number of times to run.

The example above shows a Trigger that runs every week starting from the last record's update, up to 3 times.

You can view stored delayed triggers by clicking **Queue** on an object view page. A queue displays at most 1000 stored triggers and shows when they will run. It can be filtered by the selected object record.

Queued Events: Automated Trial Expiration Warning (7 days before)			
Action	ID	Record Name	Will Run At
Del	19760439	wesco	09/17/2011 11:41 AM
Del	19776775	VEZUS - uslužno trgovski obrt	09/17/2011 11:39 PM
Del	19780904	Sigma	09/18/2011 01:15 AM

When setting triggers to run after a delay or repeat, keep the following in mind:

- Be careful when using delayed or repeated triggers to avoid unnecessary recursion.
- For delayed triggers, the timing options before or after update or creation are irrelevant.

- Object Script running in delayed or recursive triggers requires the appropriate permissions be set for the Server API role. For information about security and access control, see [Security and Access Control](#) on page 375.
- Delayed and recursive triggers do not run if a Customer tenant is expired or inactive.
- The **Debug Formula** button (in the trigger formula editor) for delayed or recursive triggers uses the Server API role to calculate permissions.
- If a **Date** field is used as relative point for a delayed trigger, its value is interpreted as 12AM in the user's time zone. If a **Date** or **Date/Time** field has no value, no delay will be set and the trigger will run immediately.

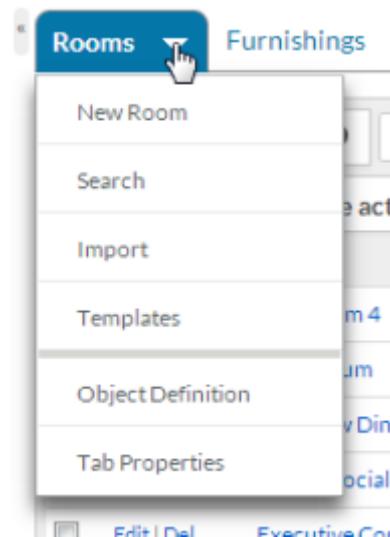
See [Debugging Delayed Triggers](#) on page 194 for more on debugging delayed triggers.

## Creating a Trigger

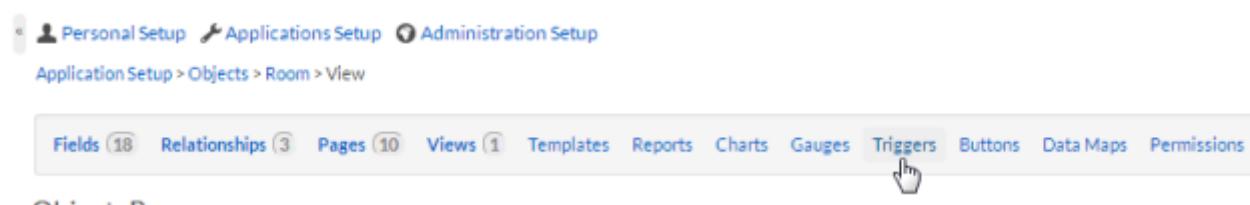
This topic covers general steps to create a trigger. For specific examples, see [Example: Set Field Based on a Workflow Status Change](#) on page 193, or watch [Web App Trigger Examples](#) to see how triggers can update fields in related records and create new records.

1. Navigate to the **Object Definition** screen. For example:

From an application, click the arrow next to the object name and elect **Object Definition**:



2. In the ribbon of object components, click **Triggers**:



The **Triggers** component table displays:



3. Click **New Trigger**.

The **New Trigger** screen displays:

4. Select the trigger type and click **Next**.

The options available on the next screen depend on the type of trigger you select.

5. Finalize the definition of your trigger using the controls on this screen:

- **Deployment Status** — If checked, the trigger will run. If unchecked, the trigger will not run.
- **Trigger Timing** — Causes the trigger to fire during events in a record lifecycle. See [Trigger Timing Options](#) on page 182 for details.
- **General Properties** — Enter a name and an integration name. If you selected **Before Update** or **After Update** timing, choose the field(s) for which an update should fire the trigger.
- **Trigger Properties** — Read the **Tip** to understand what type of formula the trigger expects. Use the **Template Helper** and formula editor to create and validate the expression. See [Best Practices for Trigger Formulas](#) on page 183 for more information.
- **Trigger Delay Time** — Optionally, set a delay time. See [Delayed and Repeating Triggers](#) on page 184 for details.
- **Recursion** — Optionally, set times for the trigger to repeat. See [Delayed and Repeating Triggers](#) on page 184 for details.

6. Click **Save**.

The **Triggers** table in the **Object Definition** contains the new trigger. You can control the order in which triggers fire by reordering them in this table.

## Trigger Types

The following sections describe the types of triggers you can create:

### Send Email

This Trigger sends an email based on an Email Template. Alternatively, you can select an Email Template field to use per record (this provides a way to dynamically determine which email template to use based on a field value in the record).

Warning: You must create at least one Email Template for a given Object definition to create this type of Trigger.

Apart from the common Trigger settings described above, you must select the recipient email address (and, optionally, CC and BCC recipients). This may be a specific address (you can use the popup selector to choose one), or an email address stored in one of the record's fields (use the "Use Field" helper). You must also select the Email Template or Email Template field to use.

Tip: You can also type in text and Formula fields which return one or group of (separated by space, comma, or semicolon) valid email address.

You can also specify whether the "Reply To" field in a trigger-generated email should contain the email address of the current user, the default auto-reply email address (defined in the Account Settings page), or explicitly specified email address.

## Create Audit Trail Record

This Trigger creates a new **Audit Trail** record based on a template defined by you and attaches it to the current or related record. To use this Trigger, select an Object (either the current record's object type or that of a related record) and specify the template to populate the audit trail record's text.

Warning: the **Audit Trail** option must be enabled for a given Object definition to create this type of Trigger.

## Validate Record Data

This Trigger validates the record's data using an expression. If the expression results in an error (i.e. a non-null value expected as a String message), the data operation is terminated and an error message is displayed. This is similar to the situation where a Field validation fails when creating or modifying a record. To use this Trigger, provide a formula expression to validate. The formula returns a string error message when validation fails, or an empty string or null if the validation succeeds. For example:

```
if ({!amount} > 100000)
    return "Order is too large: {!amount}";
else
    return null;
```

Another example:

```
if ("{!email}" == "" && "{!phone}" == "")
    return "Either email or phone must be specified";
else
    return null;
```

Note: You can choose the "Treat this trigger as a Warning" option to allow trigger execution to continue even if there is an error message. In this case if validation fails at runtime the user will have a chance to check an "Ignore Warning" checkbox and proceed with saving data despite the warning. This is similar to custom field validation (see [Field Validation](#) on page 128).

## Unique Fields Combination

This Trigger validates that a certain combination of fields is unique across all records. If the combination is not unique (another record contains the same values for each of the fields specified here), the operation is terminated and an error message is displayed. To use this Trigger, select:

- A combination of fields that must be unique
- The error message to display (e.g., "Author and title combination must be unique")
- Option to ignore fields' combination with NULL values.

## Update Field Value

This Trigger updates the value of a field in the current record or a related record using a formula. To use this Trigger select:

- A record to update: current or related record(s).

- A field to change (depends on the previous selection).
- Formula to calculate the new field's value: See below for expected return types.

Example 1: update text field with value depending on numeric field:

```
({!amount} > 1000) ? "Big Order" : "Small Order";
```

Example 2: update lookup field with multiple values (use JSON array):

```
[ 12345, 56789 ]
```

Important: If this formula evaluates to or returns null, the field's value will NOT be changed.

In addition, the following options are available when configuring:

- Access Control Policy: Most Trigger types do not check Permissions; however, the Update Field Value Trigger (along with "Change Workflow Status" and "Create New Record" Trigger types described below) does check update and create permissions for the current user. When configuring a trigger of this type you can choose what Rollbase should do when access is not allowed:
  - Do nothing: (ignore permissions)
  - Skip this Trigger: Do not run the trigger
  - Throw an error: This will terminate entire update or create transaction
- Dependent Triggers. Most Triggers do not result in running other Triggers; however, the Update Field Value Trigger (along with "Change Workflow Status" and "Create New Record" Trigger types described below) does have the option to run dependent Triggers after this one is completed by checking the checkbox called "Run dependent triggers after this one is completed".

Warning: If you use this option, we highly recommended using the Trigger debugger (described below) to ensure expected results.

The formula used in this Trigger must return a value that is used to update the selected field on the current or related record(s). The expected return value depends on the type of the field to be updated, as in the table below:

Field Type	Expected Value
Text fields	String
Numeric fields	Number
Checkbox	true or false
Process	Process ID
Status	Status ID or integration code
Single picklist	Picklist item ID or integration code
Multiple picklist	Picklist item ID as number or comma-separated string, with the picklist item's IDs

Field Type	Expected Value
Lookup	Related record ID as number, JSON array of IDs, or comma-separated string with related records IDs
Template	Select Template ID or integration code
Date or Date/Time	JavaScript Date Object or number of milliseconds as returned by the JavaScript <code>getTime()</code> function
Time	String representing time in user selected format. Example: "2:05 PM"

## Change Workflow Status

This Trigger changes the Workflow status of a record. To use this Trigger, select a target Workflow status. Use a Trigger Condition Formula to make this change optional. Also, refer to the [Update Field Value](#) on page 187 for more details.

Note: You must create at least one Workflow Status before you can use this type of Trigger.

## Create New Record

This Trigger creates a new record from the current record using a Conversion Map. To use this Trigger, select a Conversion Map (which determines the type of the resulting record). Also, refer to the notes in [Update Field Value](#) on page 187 for more details.

Note: You must create at least one Conversion Map to convert the current record into another record to use this type of Trigger.

ID of newly created record is available for other triggers in update chain through shared value named "newID\_objectName". The following Object Script example reads ID of newly created record and creates Activity Log record:

```
var newID = rbv_api.getSharedValue("newID_entity");
rbv_api.createActivityLog("entity", {!id}, "New record created: "+newID);
```

## Attach Related Record

This Trigger attaches the current record to a Related Record (i.e. establishes a relationship between two Records). To use this Trigger select or define:

- A relationship between current Object and a target Object.
- A formula that calculates the ID or array of IDs of the related record (instead of a condition formula). If this formula returns a null or negative number, this Trigger is ignored.

The following formula example uses Query API to determine ID of record to be attached:

```
var arr = rbv_api.selectQuery("SELECT id FROM contact WHERE status=?", 'Ready');

var ids = new Array();
for (var k=0; k<arr.length; k++) {
  ids[k] = arr[k][0];
```

```
    }
    return ids;
```

Note: You must create at least one relationship for the current Object in order to use this type of Trigger.

## Create Template Document

This Trigger creates a template-based document for the current record and (optionally) sends it in an email as an attachment. For instance, you can use this Trigger to generate an Invoice and send it to a customer. To use this Trigger select:

- A document template or Template picklist field (which selects a document template based on the value of that field for a given record) to use.
- The File Upload field where to store the resulting generated template-based document.
- An Email template and email address to send resulting document (optional).

You can create a Document Template field that generates a new document each time you view it. This Trigger is similar, but it generates and stores the template document in a particular field and its contents do not change even if other field values change).

Note: You must create at least one Document Template and at least one File Upload field before you can use Triggers of this type.

## Run Triggers on Related Record

Typically, Triggers do not invoke other Triggers; however, some Triggers offer an explicit option to do so. This Trigger type allows you to explicitly invoke Triggers with specified Timing options on related records. To use this Trigger select:

- The Relationship between the current Object and a target Object (if the relationship is multiple, triggers will be invoked on multiple records).
- A Timing option to on target object to determine which triggers to invoke (e.g., "On After Update") or explicitly select a group of related triggers to run.

Warning: If this Trigger is used, we highly recommend using the Trigger Debugger (described below) to ensure expected results.

## Object Script

Object Script triggers run JavaScript, allowing manipulation of multiple fields and records in one trigger via server-side Query API calls. To use this Trigger, enter a formula that returns no value, but invokes one or more Rollbase API calls (see [Server-side API](#) on page 559 for more information). The following example modifies the field "amount" and prints a message displayed in the trigger debugger:

```
rbv_api.setFieldValue("invoice", {!id}, "amount",
    {!amount}*(1-!discount)/100));
rbv_api.print("amount after discount: "+
    rbv_api.getFieldValue("invoice", {!id}, "amount"));
```

Progress recommends that only experienced Rollbase developers use object script triggers. See [Debugging Complex Triggers](#) on page 193 for more information.

You can use all types of JavaScript in Object Script triggers including flow control constructions. To terminate flow of Object Script trigger, as well as all subsequent triggers, and rollback entire transaction you can throw a JavaScript exception:

```
if (something_is_terrifically_wrong)
    throw "Cannot perform operation - aborted";
```

## HTTP Triggers

You can use triggers to send HTTP Get and Post requests and send SMS messages.

### Processing Responses from HTTP Triggers

After the HTTP response is retrieved by GET, POST, or SMS Trigger, the system stores two values and makes them available for subsequent Triggers through the `getSharedValue()` API (see [rbv\\_api.getSharedValue\(\)](#) on page 609):

Name	Value	Example
ReturnStatus	HTTP Status	<code>rbv_api.getSharedValue("ReturnStatus")</code>
ReturnBody	HTTP Body	<code>rbv_api.getSharedValue("ReturnBody")</code>

For example:

1. Create an HTTP GET or HTTP POST trigger.
2. Create an [Object Script](#) on page 190 and configure it to run immediately after HTTP trigger. Use the following API calls in Object Script trigger:

```
var code = rbv_api.getSharedValue("ReturnStatus");
var body = rbv_api.getSharedValue("ReturnBody");
```

Now you can analyze HTTP return code and response's body and perform appropriate action.

Tip: You can create powerful integrations by sending HTTP requests and then processing the results in a subsequent Object Script Trigger. Please note that second etc. HTTP call will override shared values set on the first call. Store these values in intermediate variables if you need to preserve them.

### Send HTTP Post

This Trigger sends an HTTP POST request to a specified URL. It can be used to send any XML (i.e. SOAP) or other POST request to a third party containing information and instructions gathered from Rollbase record data. To use this Trigger, select:

- A Document Template to generate the request with an .XML extension
- The target URL to send the HTTP POST request to
- Encoding for selected template: XML, URL, or other
- The Content Type (text/xml by default)
- Up to 5 HTTP headers (names and values)
- Timeout (in ms) for HTTP request
- Check if you want this Trigger to throw an exception if the HTTP return code is out of the range 200-210 indicating success.

You can immediately debug your Trigger by selecting a record to debug against (click the icon). This sends a generated HTTP POST request to the specified URL and displays results in a popup window.

Note: You must create at least one text-based Document Template before you can use this Trigger.

Important: The HTTP POST Trigger does not add a header or footer to your XML document.

## Send HTTP Get Request

This Trigger sends an HTTP GET request to a specified URL. It can be used to send a REST-style request to a third party. To use this Trigger, select:

- An Integration Link field (which includes a dynamically generated destination URL template).
- A timeout (in ms) for the HTTP request

You can immediately debug your Trigger by selecting a record to run it on (click the icon). This sends the HTTP GET request to the generated URL and displays the results in a popup window.

Result of HTTP GET request is stored in shared variables similar to POST request - see [Send HTTP Post](#) on page 191.

Note: You must create at least one Integration Link field before you can use Triggers of this type.

## Send SMS Message

This Trigger sends an HTTP request to SMS Gateway server, which sends SMS message to recipient. You need to have a valid account with SMS service provider to facilitate this service. To use this Trigger, select:

- Template for URL used to reach SMS Gateway. This template typically should include:
- Login credentials for your SMS account
- Address of SMS sender
- Phone number of SMS recipient
- Text of SMS message
- Request type: HTTP GET or HTTP POST
- Timeout (in ms) for HTTP request

The following example shows a URL template which reaches to Clickatell SMS Gateway (<https://www.clickatell.com/>):

```
http://api.clickatell.com/http/sendmsg?user={#!SETTINGS.sms_user}&password={#!SETTINGS.sms_password}&  
api_id={#!SETTINGS.sms_api_id}&MO=1&from={#!SETTINGS.sms_from}&to={!cell_phone#value}&  
text=Your+verification+code+{!ver_code}
```

Please note:

- SMS credentials (user name, password, API ID, sender's phone number) are stored in Settings object. Template tokens for these fields are used in URL instead of actual values.
- Template token for recipient's phone number uses #value suffix to remove any formatting.
- Explicitly provided text message must use URL encoding.

- HTTP response will include ID of sent message or error code.

## Example: Set Field Based on a Workflow Status Change

Consider the following example: The date when a record's workflow status changes to "Closed" should be captured for reporting purposes. You can accomplish this as follows:

- Create a workflow status named "Closed" and assign it an integration code "C."
- Create a **Date** field to the object called "Closing Date." Only add this field to the view page so it is essentially treated as a read-only field.
- Create a workflow action named "Close" which will change the status to "Closed."
- Create a trigger named "Record Closing Date" with the following configuration:

Options	Description
<b>Timing Option</b>	After Update
<b>On Field Change</b>	Status
<b>Field to Change</b>	Closing Date
<b>Change Value Formula</b>	<code>"{!status#code}"=="C" ? new Date() : null</code>

As soon as the status is changed to "Closed" (regardless of whether this change was invoked by the Workflow action, manual record editing, another Trigger, API call or other source of change), the current date will be stored in the **Closing Date** field.

## Debugging Complex Triggers

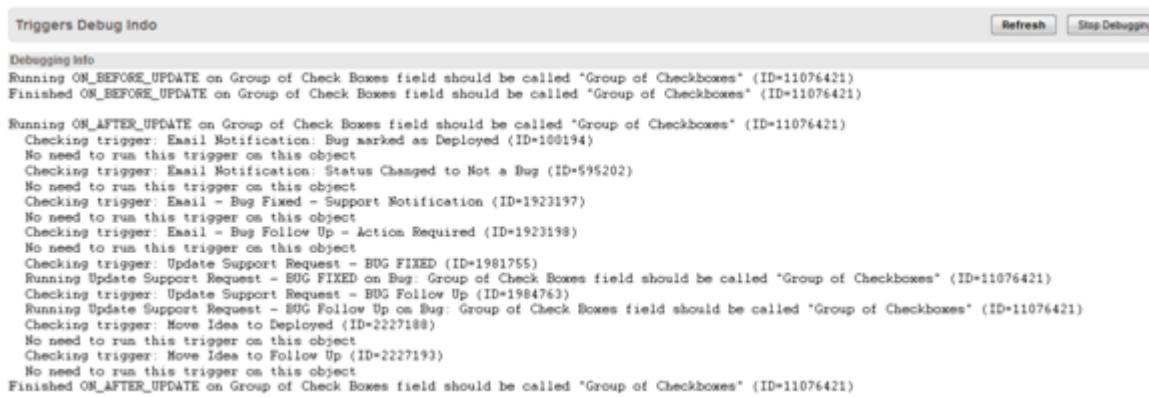
When you use a group of several Triggers that invoke Triggers on related records, it is critical that you use the trigger debugger to ensure expected results. Use `rbv_api.print()` API calls to print additional debugging info, including field values and intermediate formula results.

Warning: Delayed triggers cannot be debugged in the trigger debugger since the user is absent when they're running. To debug these triggers consider running them without any delays.

To use the trigger debugger, simply click the "Debug" link in the Triggers section of the Object definition's page that displays a pop-up Debug window. To use the debugger, follow these steps:

- Without closing the Debug window, open any object tab.
- Perform an operation on a single record: Create, Update, or Delete.
- Come back to the Debug window and click the Refresh button.
- The debugger displays a trace of all Triggers that were invoked on the selected record as well as those invoked on related records. It also displays printouts from any `rbv_api.print()` API calls.

The following screenshot illustrates a debugging trace for both current and related records shown in the Debug window:



The screenshot shows the Rollbase Debug window with the title 'Triggers Debug Indo'. It contains a list of triggers with their descriptions and execution status. The triggers listed are:

- Running ON\_BEFORE\_UPDATE on Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)
- Finished ON\_BEFORE\_UPDATE on Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)
- Running ON\_AFTER\_UPDATE on Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)
  - Checking trigger: Email Notification: Bug marked as Deployed (ID=100194)
  - No need to run this trigger on this object
  - Checking trigger: Email Notification: Status Changed to Not a Bug (ID=595202)
  - No need to run this trigger on this object
  - Checking trigger: Email - Bug Fixed - Support Notification (ID=1923197)
  - No need to run this trigger on this object
  - Checking trigger: Email - Bug Follow Up - Action Required (ID=1923198)
  - No need to run this trigger on this object
  - Checking trigger: Update Support Request - BUG FIXED (ID=1981755)
  - Running Update Support Request - BUG FIXED on Bug: Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)
  - Checking trigger: Update Support Request - BUG Follow Up (ID=1984763)
  - Running Update Support Request - BUG Follow Up on Bug: Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)
  - Checking trigger: Move Ideas to Deployed (ID=2227188)
  - No need to run this trigger on this object
  - Checking trigger: Move Ideas to Follow Up (ID=2227193)
  - No need to run this trigger on this object
- Finished ON\_AFTER\_UPDATE on Group of Check Boxes field should be called "Group of Checkboxes" (ID=11076421)

## Debugging Delayed Triggers

Debugging delayed triggers has additional challenge since there is no one around then they run. The following considerations will be helpful:

1. Always debug triggers in immediate mode first before turning on "delayed" option.
2. Check "Queue" page to see when trigger is scheduled to run.
3. Remember that Object Script on delayed triggers require permissions assigned to "Server API" role. For information on security and access control, see [Security and Access Control](#) on page 375
4. It is advisable to create Activity Log triggers for debugging purposes since regular Debug window is not available for delayed triggers.

## Workflow Overview

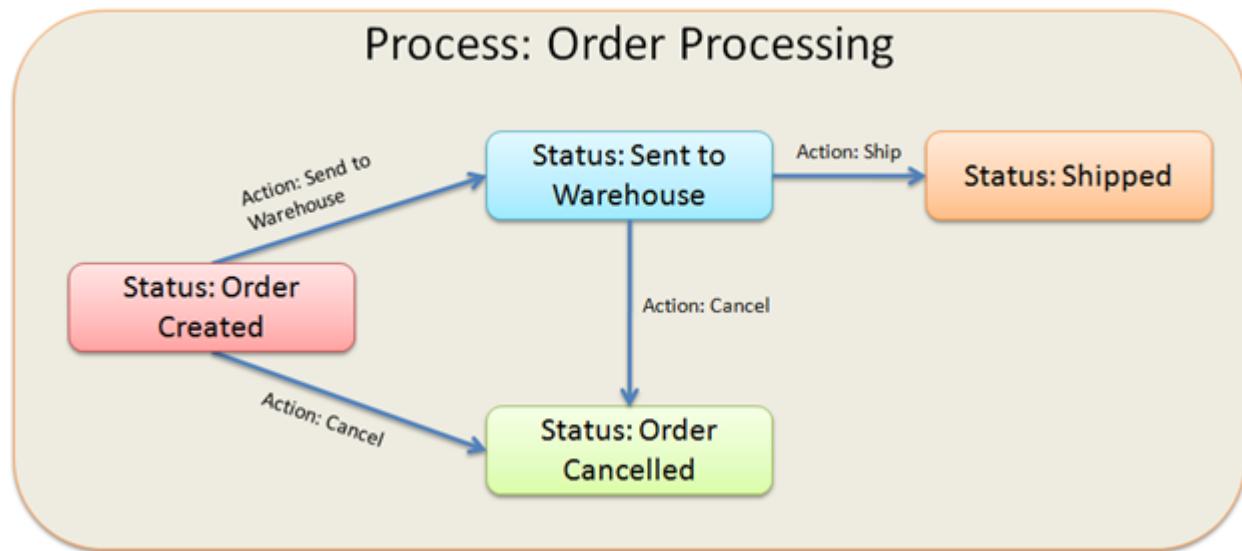
To use workflows, you must first enable the **Workflow** attribute on the desired Object definitions when you create or edit an object. This creates a default record Status (called "Created") and a Workflow Process that you then can modify based on your specific needs. The **Workflow** attribute is in the **Advanced Options** section of an object definition:

Advanced Attribute	Description
<input checked="" type="checkbox"/> <b>Workflow</b>	Objects with the Workflow attribute can be routed through an automated or manual workflow process defined by a set of workflow statuses, actions, and events. Workflow events allow you to create and deploy business rules that are manually or automatically triggered based on specific criteria defined by you. A field called Workflow Status will be added when this attribute is enabled.

It is important to understand three basic Workflow concepts:

- Workflow Statuses: Indicate current state of a record. A current record's status determines the set of currently available Workflow Actions.
- Workflow Actions: An operation that a user can presently perform on a record. The action can change the status of a record, invoke one or more triggers, send an email, etc.
- Workflow Processes: A container for a set of statuses and actions, that as a whole make up a particular workflow process.

The following diagram illustrates a sample workflow process for handling orders. By defining workflow statuses, it is possible to trigger actions based on an order's change in status.



The following sub-section describes each component of a workflow attribute in detail.

## Workflow Status

A status represents the current state of a record. Statuses are similar to picklist values, with a special meaning in Workflow. To define a new Workflow status, you specify:

- Display name (mandatory)
- Integration name/code (has the same meaning as it does in picklist values)
- Whether or not you want to automatically create a new corresponding Workflow Action to enable moving records into this new status. You can also provide a name for that new action (the status name will be used by default).

When you enable the Workflow attribute on an Object, the system generates one default status named "Created." You can then add other statuses as needed. Statuses can be assigned through Workflow Actions, or explicitly by editing an Object record. When editing an object record, statuses are presented in sequential order. You can define this order by clicking the "Reorder" link in the Workflow Statuses section of an Object Definition's View Page.

## Workflow Actions

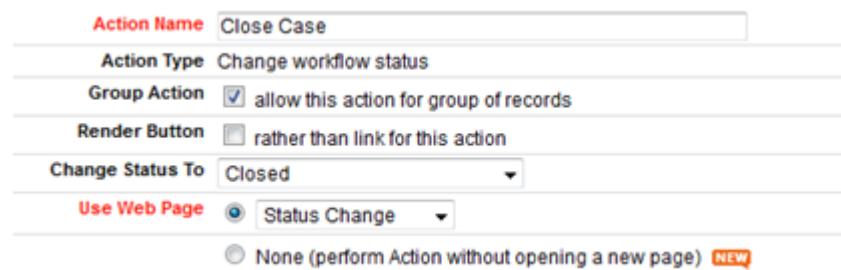
A Workflow Action represents a manual action such as a click or selection that can be performed on a record that changes the record's status, sends an email, invokes a trigger, etc. Rollbase supports several types of actions, as described in the sections below.

### Change Status Action

This action changes the Workflow status of the current record. You can create any number of "Status Change" Pages and use them for different Workflow actions. You can place any editable field from the Object definition on these Pages. This allows you to record various information at the time of a user-driven status change.

To create this Action select:

- New workflow status assigned as a result of this action
- Check if this action can be performed on group of records selected in List View
- Check if this action should be rendered as a button (rather than item in "More Actions" drop-down list) on Record View page. If this option is checked links to this action will be rendered using bold font to draw attention.
- Web page of type "Status Change" to display to the user when changing the status (this allows you to present fields for the user to complete or edit upon status change)
- Alternatively you can choose to run this action without using Status Change page, in one click.
- In addition you can select deployed triggers to run when this Action is completed.



Action Name: Close Case

Action Type: Change workflow status

Group Action:  allow this action for group of records

Render Button:  rather than link for this action

Change Status To: Closed

Use Web Page:  Status Change

None (perform Action without opening a new page) [NEW](#)

## Related Record Action

This action creates a related record of the selected type using a conversion map to either create a record of a different object type or cloning of the current record if desired. To create this type of Action select:

- Web page of type "New" to create a record of selected type (this will be the page the user is taken to when the action is performed)
- Conversion Map to transfer fields from original record to the new one. For information about Conversion Maps, see [Record Conversion](#) on page 140.
- Record Template to create new record (this will be ignored if Conversion Map is selected)

## Send Email Action

This action sends a template-based email using data from the current record. This action is only available if at least one email template exists for the current Object. To create this Action select:

- Reply-To email address
- Destination email address or addresses and/or email address stored in the current record
- Optionally CC and BCC
- Email Template to use
- Alternatively Email Template Picklist (if defined for this Object)
- In addition you can select deployed triggers to run when this Action is completed.

Tip: You can also type in text and Formula fields which return valid email address. Alternatively, you can select the "Email Template" field to use (which gets its value from the record the action is working upon).

## Template Document Action

This action generates a template-based document using data from the current record and presents this document to the user in a popup window. When used as a group action (see below), this action will merge documents from all selected records into one. This action is only available if at least one document template exists for the current Object.

To create this Action select:

- Document template to use
- Alternatively Document Template Picklist (if defined for this Object). This option is not available for Group Actions.

## Run Triggers Action

This action runs a group of selected Triggers on the current record. This action is only available if at least one Trigger exists for the current Object.

To create this Action, select:

- Triggers to run and specify the order you want to run them in.
- Select Status to change (optionally).
- Check if standard "Run Triggers" page should be displayed or Action should be performed without opening new page.

## Workflow Action Properties

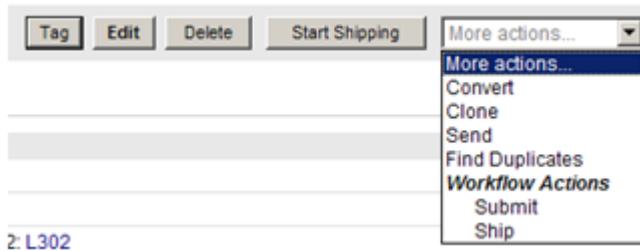
Every Workflow Action allows selecting:

- Whether this action is available for a group of records (see below)
- Whether this action should be rendered as a button rather than an item in a drop-down list
- Whether to change the record's Workflow status(es) when the action is performed.
- A Formula that returns true or false. If the formula returns false, this action will not be available for the current record. For more details on formulas, see [Server-side API](#) on page 559 .

You can reorder Workflow actions to set an order in which they appear in any list.

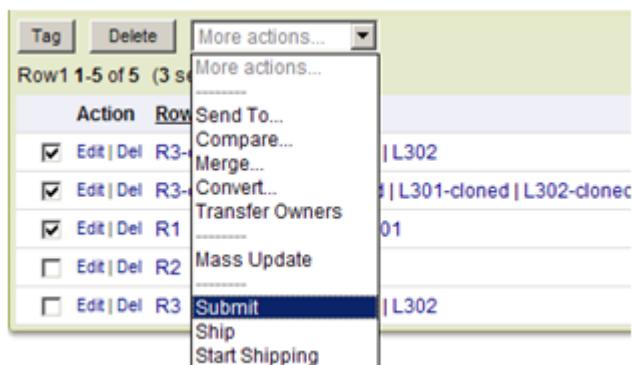
Access to Workflow Actions can be limited only to certain roles or users. For more information about Rollbase Access Control, see [Security and Access Control](#) on page 375.

The read-only field named Workflow Actions renders links to available actions. This field can be added to View pages and as a column in List Views. In addition, actions available to the current user for the current record are rendered in a drop-down box in the header of the View page, or as buttons if this option is selected:



## Group Action Properties

If you check the Group Action box for a particular Workflow Action, it becomes available for a group of selected records in List Views by selecting the action from the "More Actions" drop-down list.



Group actions work the same as single record actions, as follows:

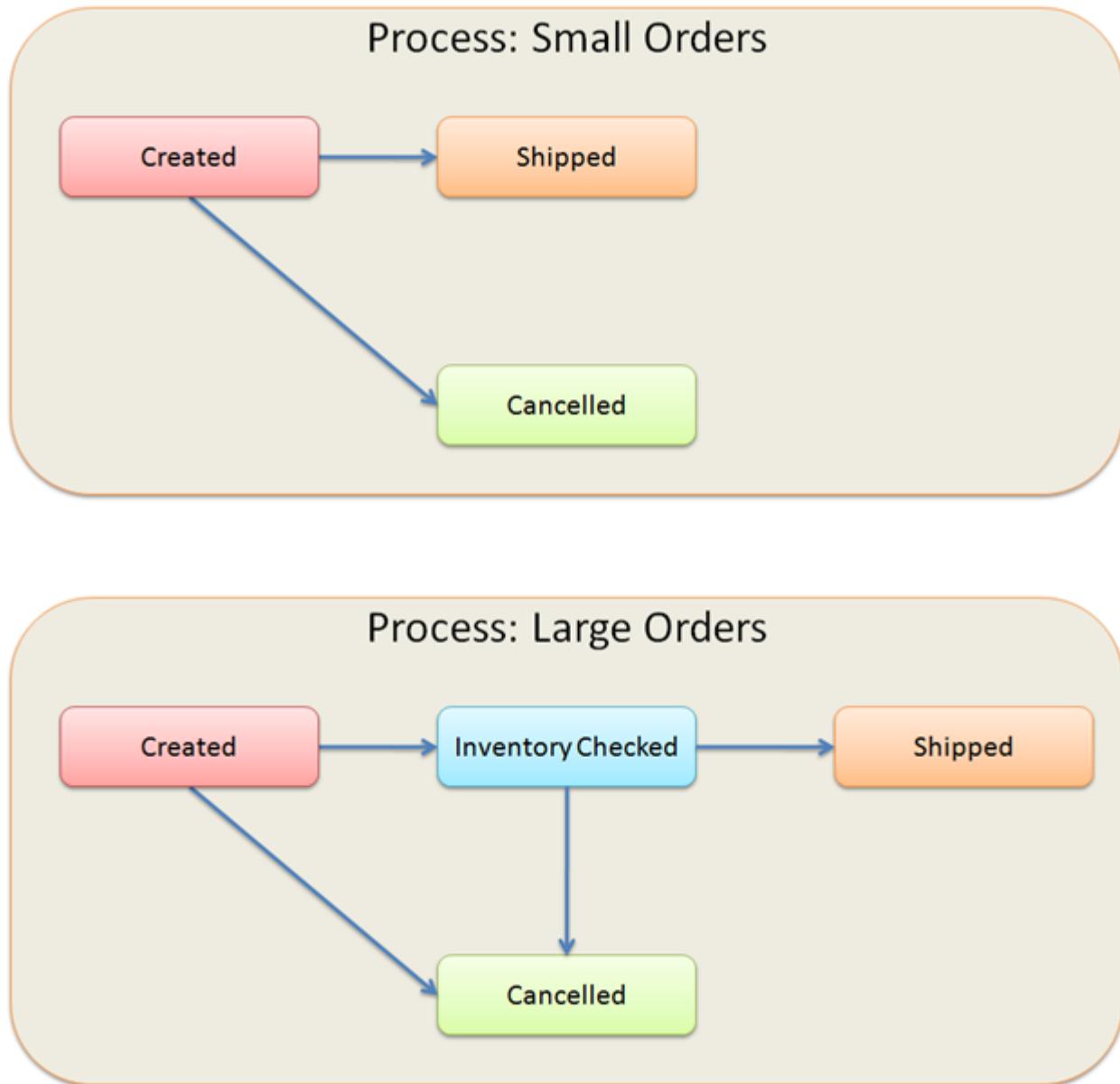
- Status Change: Changes statuses and updates fields on selected records. If the field value is not set, it will remain unchanged.
- Related Records: Creates related records for all selected records.
- Send Email: Sends emails using the selected records.
- Template Document: Generates template-based documents for selected records and merges them into a single document.
- Run Triggers: Runs one or more Triggers on the selected records.

If group workflow action has formula-based condition, selected records will be filtered by using that condition.

## Workflow Processes

A Workflow Process is a container for a group of Workflow Statuses and Actions, moving a record from one status to another. When the Workflow attribute is first set on an object definition, Rollbase creates one Process named "Default." In most cases all records follow the same Workflow and it is sufficient to have only one Workflow Process.

In some complex cases, when Workflow rules are different for different records, you can define more than one Process. The diagram below illustrates two different processes for an Order object: Small Orders and Large Orders.



## Creating a Workflow Process

From the default status, Workflow actions can allow movement of a record into another set of statuses that implicitly determines how a particular Workflow Process is designed. You can reorder Workflow Processes to set the order in which they appear in any list.

When you have more than one process, you can use a Trigger to automatically assign the appropriate process to newly created records based on certain criteria (For example, an order's amount value).

To create a new Workflow Process:

1. Open the **Object definition** page (See [Viewing and Editing an Object Definition](#) on page 121).
2. From the object attributes ribbon, select **Workflow Processes** to navigate to the processes area of the page.

3. Click **New Workflow Process** to define a new Workflow process.
4. Specify **Process Name** and a **Default Status** to be used when this process is assigned to a particular record. The newly created process will list under the **Workflow Processes** area.

To design a Workflow process, you must first define **Workflow Actions** and **Workflow Statuses** that can be a part of the process. All **Workflow statuses** can participate in any **Workflow processes** and the [Editing and Viewing a Workflow Process](#) on page 200 describes how to edit and view your the workflow process.

## Editing and Viewing a Workflow Process

The following procedure describes how to edit and view a Workflow Process:

1. Open the **Object definition** page (See [Viewing and Editing an Object Definition](#) on page 121).
2. From the object attributes ribbon, select **Workflow Processes** to navigate to processes area of the page.
3. Under the **Process** column, select a Workflow process to open and edit it in the process view.
4. Select **Edit** to redefine the process attributes, **Delete** to remove the process from the object, **View Diagram** to see a visual representation of the Workflow statuses included in the process.

## Approvals

Rollbase Approvals involve multi-step processes which should be planned in advance to provide the best results. The Approval process starts when a particular record is moved to the Workflow status called "Waiting for Approval," and ends when the record is moved to the status called "Approved" or "Rejected." Rollbase provides a default implementation for this process that you can customize, including the names of these statuses.

To configure an Approval Process follow these steps:

1. [Install the Approvals Application](#) on page 200
2. [Select Approvers](#) on page 200
3. [Set the Approval Attribute](#) on page 201
4. [Set Up the Approval Process](#) on page 201
5. [Optionally, Customize Approvals Process](#) on page 202

### Install the Approvals Application

Install the Approvals Application (Sidebar > Find Apps > Approvals) from the Application Directory. This application has a system Object named Approval. Each approval record contains the following information:

- A reference to the record being approved
- A reference to the user being asked to approve or reject that record (approver)
- The current status of this approval: pending, approved, or rejected

### Select Approvers

Next, select a group of Users who will participate in the Approval process. Make sure the "Approver" checkbox is checked for each of these Users in the User Edit page (if you do not see this box, use the Page Editor to add it to the Page).

## Set the Approval Attribute

Edit the Object you want to participate in the Approval process and enable the "Approval" attribute (which automatically enables the "Workflow" attribute if not previously enabled). This creates the following components:

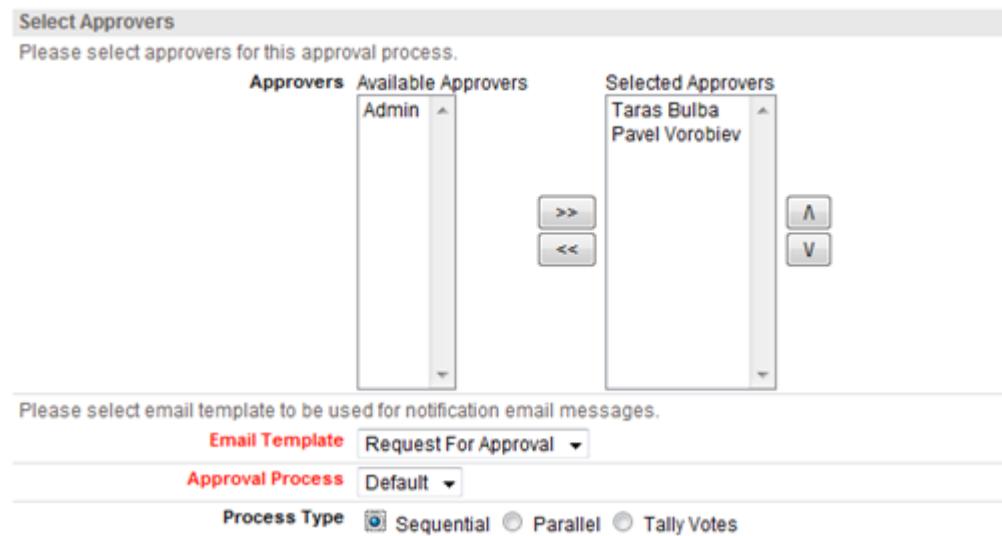
- System statuses: Approval In Progress, Approved and Declined
- Workflow Action Type "Start Approval Process," which is used to start an approval process on a particular record.

Tip: If you want more than one object participate in approval process set "Approval" attribute on each object.

## Set Up the Approval Process

Edit this "Start Approval Process" Workflow Action to complete the configuration of the approval process:

- Assign the "Start Approval Process" action to a Workflow status. Typically use the default Status for your Workflow Process.
- Select a default list of approvers (this can be changed at runtime whenever the approval process is started on a record).
- Select an email template to be used to notify approvers that they have something to approve or reject. You will want to create this beforehand.
- Choose the Workflow Process to use this approval action in (typically Default)
- Choose the approval type:
  - Sequential: Approvers will be notified in sequence. The record will be rejected if any approvers reject it, otherwise it will be approved once all approvers approve it.
  - Parallel: All approvers will be notified simultaneously and can submit their feedback in arbitrary order. The record will be rejected if one or more approvers rejects it, otherwise it will be approved.
  - Tally votes: This works identically to the parallel approval process, but will continue until all approvers approve or reject the record. The record will be approved if a majority of the approvers approve it and rejected otherwise.
- Assign appropriate permissions to run the "Start Approval Process" action.



## Optionally, Customize Approvals Process

To further customize an Approval Process you can:

- Create several Workflow Processes
- Create (by cloning) several "Start Approval Process" Actions to start them.
- Create several email templates as needed.

Once configured, your approval processes will operate as follows:

- Any user with sufficient permissions can invoke the "Start Approval Process" action for a selected record (or a number of records if the "Group Action" attribute is enabled).
- An email to the first approver will be sent (or all approvers for a parallel or tally votes process).
- The email should include a link to the record being approved, with one-click login, as described in [Enabling Single Click Log In](#) on page 387.
- Approvers will also see a list of records currently waiting for their approval in the "My Approvals" tab.
- After reviewing a record submitted for approval, the approver clicks the "Approve or Reject" button, where he/she can:
  - Provide comments
  - Click the "Approve" or "Reject" button
- If the record is approved, the sequential process will continue and an email will be sent to the next approver in the sequence.
- If the record is rejected, the approval process will end and the record will be moved to the "Rejected" status (any approver has veto power, except for the "Tally Votes" process described above).
- When all approvers have approved the record, it is moved to "Approved" status.

Tip: You can associate Triggers to run when a record's Status is changed to the Approved or Rejected status. It is common to configure a Trigger to notify specific Users when a record has been approved or rejected using this technique.

## Record Queues

The Record Queue feature is a type of Workflow process specifically tailored for situations in which a queue of records of a certain type need to be processed by a specific group of Users. Rollbase automatically creates this Workflow process when you add the Queue attribute to an Object definition.

Note: The Queue attribute will enable the Workflow attribute if not already enabled.

When you enable the "Queue" attribute on your Object definition, Rollbase will create the following Object components:

- Two new Workflow Statuses: "In Queue" and "In Process." The former is assigned by default to the first Workflow Process of your Object.
- A Relationship between this Object and the User object (if not already created).
- A new Workflow Action named "Start Processing" with a Status Change Page (more details below). This Page assigns the current User to the record being updated. The "Start Processing" action is available for the "In Queue" Status and can be used for one record or a group of records.



- A View named "In Queue," which shows only records in the "In Queue" Status in the order that they were created.

With these components in place, the Workflow process works as follows:

- New records are created (for instance, leads from a Portal) and placed in the "In Queue" status.
- Any user who has permission to access these records and the "Start Processing" Workflow Action can invoke this action on one or more groups of records.
- That User's name will be associated with the record(s) and the status of each record will be changed to "In Process".

Tip: You can further modify the "Start Processing" action and make the entire Workflow more sophisticated by adding more steps to emulate your business process. The "Queue" attribute helps you start building such a process with minimal effort.

## Reports, Charts, and Gauges

Progress Rollbase data visualization includes viewing data in reports, charts and gauges.

Reports present data in a variety of ways:

- Tabular: Each row represents a data record and each column represents a field. Tabular reports can include up to three layers to show dependent records.

- Document template: You can build custom Microsoft Word, Microsoft Excel, plain text, and XML document templates that are used to display record data in pre-specified locations.
- HTML: You can create custom HTML documents that are used to display record data in pre-specified locations.
- JavaScript: You can write custom JavaScript code to loop over a list of records, perform calculations and analysis, and display results in custom HTML.

Charts are UI components that present summaries of data in visual formats, such as bar charts, column charts, pie charts, donut charts, line graphs, and others. Charts can periodically and automatically refresh themselves without a complete page refresh. Some charts provide drill-down capabilities and interactivity options such as rotation and slice movement.

Gauges are UI components that present a single formula-based value in a way similar to a car's speedometer or a scale display. Rollbase supports several different gauge types for various visual effects. Gauges can periodically and automatically refresh themselves without a complete page refresh.

## Working with Reports

The following sections describe how to work with each of the report types available in Progress Rollbase.

Reports can only be created if the **Reports Enabled** box is checked on the object view page (checked by default).

### Tabular Reports

Tabular reports are the most common type of report. Rollbase renders tabular reports as rectangular tables in which each row represents a data record and each column represents a field value. You can define up to three layers in tabular reports with a list of related records rendered below the parent row.

The screenshot shows an example of a report with three layers of data. Notice that each layer can be expanded together by clicking either **Expand All** or **Collapse All**, as well as individually by expanding or collapsing each row:

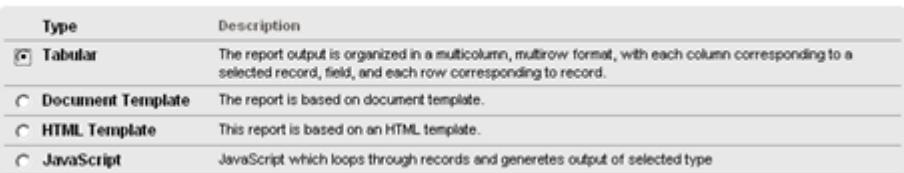
Recent Orders				Export As XLS   Export As CSV
Orders 1-2 of 2				
Order	Buyer	Total	Updated At	
O-2010-00001	Yamaguchi Supply		03/04/2010 02:47 PM	
	Amount	Item Total		Line Item
	12	\$3,600.00		Line Item
	Catalog Item	MSRP		SKU
	Office Chair	\$150.00		X009
	2	\$2,000.00		Line Item
	O-2010-00002	\$1,200.00	03/15/2010 02:51 PM	

You can create a new report in the following ways:

- Click **New Report** in the **Reports** tab in the default Rollbase application.
- Open an object definition, scroll down to the **Reports** section, and click **New Report**.
- Open any page containing a section with a report link and click **New Report** in that section's header.

Only users with the Administrator role have permission to create reports. For more details about user roles, see [User Roles](#).

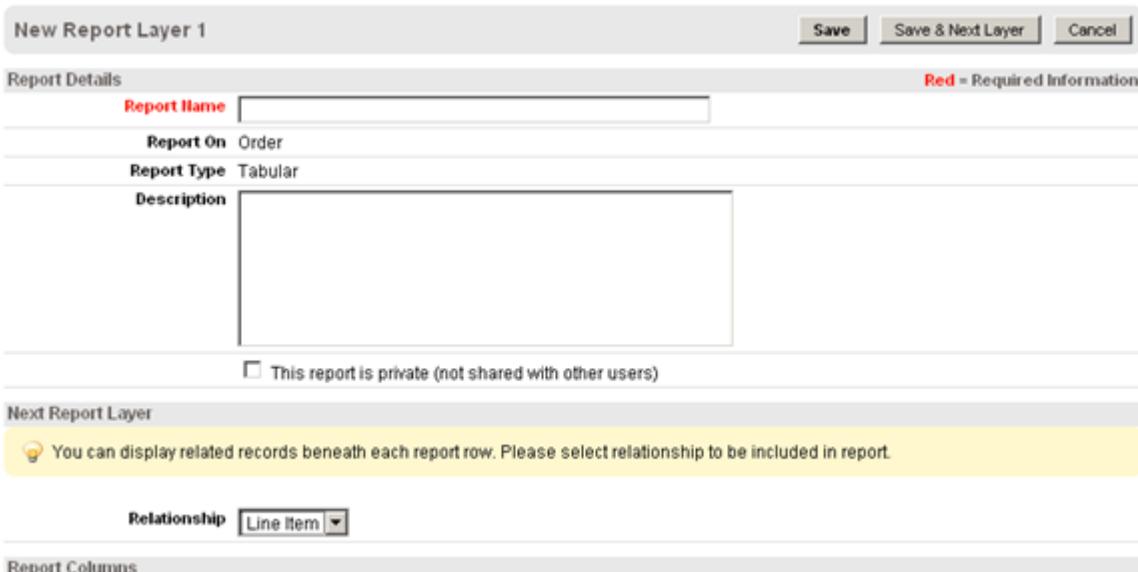
To begin creating a tabular report, navigate to the **New Report** page using one of the methods described above. Select the **Object Type** to report on and then choose the report type **Tabular**:



Select Report Type		
Report Type	Type	Description
<input checked="" type="radio"/>	Tabular	The report output is organized in a multicolumn, multirow format, with each column corresponding to a selected record, field, and each row corresponding to record.
<input type="radio"/>	Document Template	The report is based on document template.
<input type="radio"/>	HTML Template	This report is based on an HTML template.
<input type="radio"/>	JavaScript	JavaScript which loops through records and generates output of selected type

You may report on all deployed objects, system objects, audit trails, system error logs, and log-in history records.

In the next step you can define your report's columns, sorting and filtering conditions for Layer 1, similar to the way you define but with the added option to select a relationship for Layer 2 records. If you do choose a related object to display in a second layer, Rollbase will create Layer 1 and then take you to the Layer 2 edit page. You can create a third layer in a similar manner if desired. In the second and third layers, you can report on comments and activity trail records related to the Layer 1 record (i.e. the parent record).



**New Report Layer 1**

**Report Details**

Report Name  Red = Required Information

Report On Order

Report Type Tabular

Description

This report is private (not shared with other users)

**Next Report Layer**

**Relationship**

**Tip:** By marking a report as private, it becomes invisible to non-Administrator users. This box is checked by default if a non-administrative user creates a new report.

If your object has defined charts, you can optionally select a chart to be included in the report that will be rendered below the report's body.



## Document Template Reports

In addition to tabular reports, you can create reports based on a binary document template almost identical to document templates, except that reports are designed to present information on multiple records rather than a single record. For more information about document templates, see [Document Templates](#) on page 166.

To create a document template report, you have to upload a binary file with the report template. Supported formats are the same as for document templates:

- Microsoft Word (.doc)
- Microsoft Excel (.xls)
- Plain text (.txt)
- XML

To generate the report, Rollbase:

- Scans through all of the object records on which the report is being run (you can use report filters to limit this list).
- Executes template [group functions](#) on that list if they exist in the report.
- Runs template [loops](#) over that list if they exist in the report. See [Iterating through Records](#) on page 162 for more information.
- Generates the report document in the source format.

You can use the EVAL[] block to execute server-side JavaScript in document template reports.

## HTML Template Reports

HTML template reports are similar to document template reports. You don't have to upload a file, but can define your HTML code within the report definition page.

Consider this simple example of an HTML report that prints a list of Lead records:

```
<html>
  <head>
    <h2>List of Leads </h2>
  </head>
  <body>
    <table>
      { !#LOOP_BEGIN.all#154785 }
      <tr>
        <td>{ !name#text }</td><td>{ !type#value }</td>
      </tr>
      { !#LOOP_END.all }
    </table>
  </body>
</html>
```

Note that this report runs a loop through all records using a view with the original ID 154785. Inside the loop, each record outputs a row of an HTML table.

For a summary report, you can use the Rollbase query API in an EVAL[] block.

## JavaScript Reports

JavaScript reports allow you to loop through a list of records and use server-side JavaScript (or formulas) to generate output. You can also invoke complex logic and run queries using the Rollbase query API.

To define a JavaScript report:

- Select the content type of the output (plain text, HTML, XML, or CSV).
- Optionally define JavaScript to generate the report header.
- Select a view to use to loop through the selected object's records.
- Define JavaScript to be executed for each record in the loop.
- Optionally define JavaScript to generate the report footer.

Inside the header, body and footer you can use the `rbv_api.print()` or `rbv_api.println()` API methods to generate the report's output.

Consider this simple example of a JavaScript report that prints a list of Lead records:

Header:

```
rbv_api.println("Hot Leads\n=====");
```

Body:

```
if ("{!lead_type#code}"=="HOT")  
rbv_api.println("{!name#text}");
```

This will generate a text report that looks like:

```
Hot Leads  
=====  
Per Hanseon  
jose del pino  
sawyer joe  
Todd Geist  
Bob Myers
```

If you are using complex business logic to process a report's records, consider running the report asynchronously using a .

## Running Reports

After creating a report, you can use the **Page Editor** to add a report link component to a section in any generic page or object list page:



Users with view permissions on a particular report can run the report by clicking on the report's link. Users with manage reports administrative permission can edit, delete, or set view permissions for a particular report. For more information about Rollbase permissions, see .

Links to template-based reports, including JavaScript reports, can be used on **Portal** pages. Tabular reports are limited to regular pages.

Complex reports, especially template-based ones, might require a long time to execute. In this case, use the Email link to place the report in a queue for asynchronous execution. It will be emailed to you when it is ready.

When running a report, you can use dynamic report filters to filter Layer 1 records. Filtering reports works the same way as dynamic filtering in list views.

When you open the first report page, it renders the report with default filters specified when the report was created, if any. You can dynamically update these filters and click **Run Report** to render the report again with the new filter values. Click **Reset Filters** to restore the default (which might be empty) filters.

If you do not have a dynamic report filters section in your report page, use the **Page Editor** to add it to the page.

When running a template-based report, you can use the following buttons:

- **Run Report** - Displays the report with the currently selected filters.
- **Reset Filters** - Resets the filters to the original report's selection; clears filters if no filters are selected in the report.

Report Filters

Hot Leads

1 Created At greater than QUARTER

2 Amount greater than 500

3 -None-

4 -None-

5 -None-

Filter Conditions: All (AND)

Run Report  Reset Filters

When running a template-based report, you can use the following buttons:

- **Run Report** - Displays the report with the currently selected filters.
- **Print** - Displays the report in a pop-up window ready for printing.
- **Reset Filters** - Resets the filters to the original report's selection; clear filters if no filters are selected in the report.
- **Edit** - Open the report for editing (if the current user has permission to manage reports).

Summary Report

1 Amount greater or equal 1000

2 -None-

Run Report  Print  Reset Filters  Edit

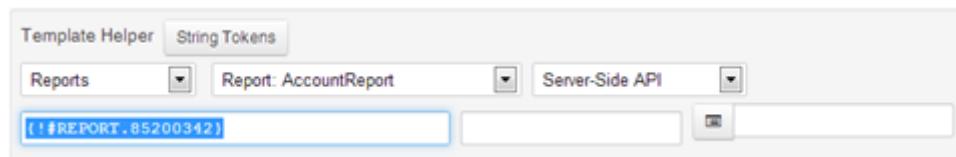
You can export the report data to XLS or CSV formats. When exporting multilayered reports, each record from each layer is exported as a unique row.

## Merging Reports

You can create reports that encompass other HTML-based or JavaScript reports. Use `{ ! #REPORT.123456 }` tokens available inside the template helper section to identify the reports you want to merge (note that these tokens use the original IDs of the reports).

To specify the reports you want to merge in a new report:

1. Create a new HTML or JavaScript report.
2. In the define report template section's **Template Helper**, select **Reports** from the drop-down list at the left as shown in the example below.
3. From the drop-down list in the center, select the name of a report you want to merge. In the example below, Report:AccountReport is selected. The programmatic expression for the report is displayed in the text box below the drop-down lists: `{ ! #REPORT.85200342 }` in the example below. Use this token in the HTML template where you want the report to appear.



4. Similarly, select another report you want to merge from the center drop-down list and copy its programmatic expression to the HTML template.

## PDF Report Options

You can render HTML reports as PDF documents by checking the **Render as PDF** check box. The generated HTML document will be converted into PDF format.

Private Cloud customers: PDF rendering is only available if PD4ML software is installed in your private cloud.

Generated PDF documents have the following attributes:

- **Author** - Your customer name
- **Title** - The name of the report

Use the **PDF Options** section to choose options for PDF document:

- Page size - Defaults to letter.
- Orientation - Defaults to portrait.
- Header's template and alignment
- Footer's template and alignment

Private Cloud customers: PDF headers and footers are only available for the PD4ML Professional edition.

In the PDF header and footer templates, you can use the following special tokens:

- `${page}`  - The number of the current page (starting from 1)
- `${total}`  - The total number of pages in the document

- \${title} - The document's title

In the body of the HTML to be rendered as PDF, use the following PDF-specific tags:

- <pd4ml:page.break> - Separates sections of the HTML report rendered as PDF
- <pd4ml:attachment description="..." src="http://..." icon="Graph - PushPin - Paperclip - Tag" /> - Includes content of the specified URL as an attachment to the PDF.

## Report Batch Jobs

You can create a batch job with a predefined frequency that will generate and email a report as an attachment to a pre-specified email address.

When generating an HTML report in a batch job, you have the option of automatically converting the report to PDF format before emailing it.

For more information about batch jobs, see .

## Charts

You can use the charts in Rollbase to visually represent a snapshot of your data. Charts support a wide variety of styles: bar, column, pie, donut, and many others. Charts can periodically and automatically refresh themselves without a complete page refresh. Some charts provide drill-down capabilities and interactivity options such as rotation and slice movement.

Rollbase offers two engines to render charts:

- FusionCharts (based on HTML)
- Google Charts (based on HTML)

## Creating Charts

You create charts from an object definitions page. Use the following two-step process to build or configure a chart:

1. Go to an object definition page, scroll down to the **Charts** section and then click **New Chart**.  
The **New Chart** page appears:

## Chart: New Chart

**Next >****Cancel**

## Chart Engine

Chart Engine  **Fusion Chart (HTML)**  **Google Chart (HTML)**

## Chart Axes

X Axis Y Axis **Next >****Cancel**

2. Select one of the chart engines, **FusionCharts (HTML)** or **Google Charts (HTML)**.
3. Designate how records of a particular object will be grouped into collections (columns or slices). A set of collections is referred to as the **X Axis** (this refers to a bar-style charts and can be logically extended to other types of charts). You can create these groupings for:
  - Enumeration fields (pick lists, status)
  - Lookup fields (one or more related records)
  - Date fields (dates can be further grouped by week, month, quarter, or year)
  - Numeric fields (allows specification of a range of numeric values)
4. Specify which numeric value will be summarized for each collection. This value is referred as the **Y Axis**. This value might belong to either the parent or to a related object. You can also use the count of parent or of related records.
5. Click Next and specify the following information:
  - A name for the chart and labels for the X-Axis and Y-Axis (not used for pie and donuts charts)
  - The chart style
  - The chart width and height in pixels
6. Optionally, specify the following if you selected **FusionCharts** as your chart engine:
  - Select **Animate Chart** if you want HTML-based animation for this chart when available.
  - Select **Auto-Refresh Chart** if you want this chart to be automatically refreshed without manually reloading the page or report where the chart resides.
  - Select **Disable Drill-Down** if you do not want to allow drill-down to records in chart columns or pie slices.
  - Select **View for Drill-Down** to specify the view to use to display the drill-down data.
7. Specify the **Chart Columns** setting for the selected collections:
  - For numeric fields, select the range and width of each collection.
  - For date fields, select the date's interval (week, month, quarter, or year) range.

- In the **Compute What** field, specify what is to be computed for records that fall in a particular collection: total, average, or percentage of this collection relative to all collections total.

Warning: To preserve system resources, the total number of collections per chart is limited to 200. If you select conditions that generate too many collections, not all collections will be displayed and Rollbase will generate a warning message. Hiding empty collections by checking the **Hide empty columns** check box will improve the visual representation of your chart but will not remove the limitation on the total number of displayed collections.

8. Optionally, use **Chart Filters** to filter and render different types of data on your chart.

## Using Charts

To make a chart available for viewing on a page, use the **Page Editor** to add the chart component to an empty section on a generic or object list view page.

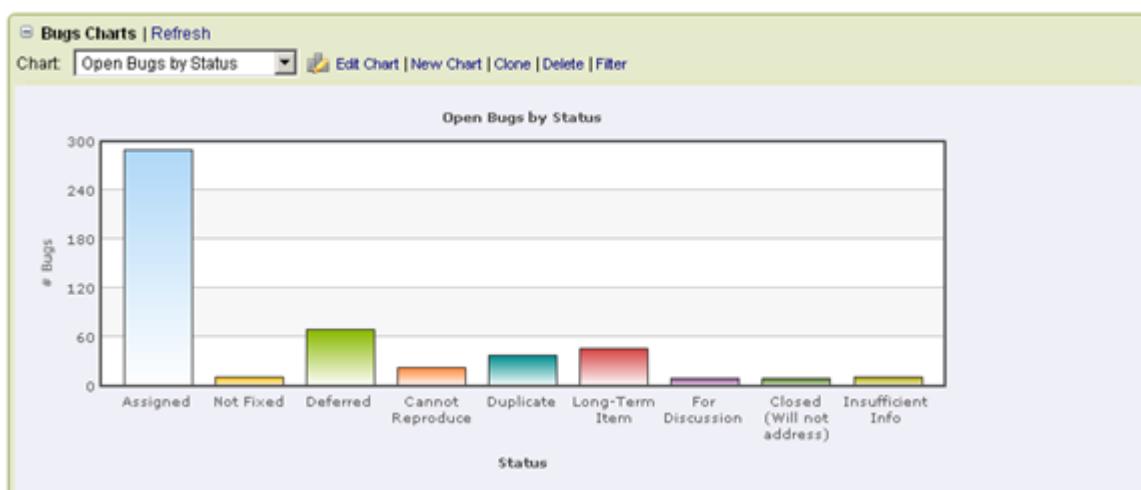
This will result in displaying the selected chart on your page. From this runtime view of the chart, you can use the following options in the section header:

- Select another chart
- Edit the selected chart
- Create a new chart or clone the selected chart
- Delete the selected chart
- Dynamically filter data visualized in the selected chart

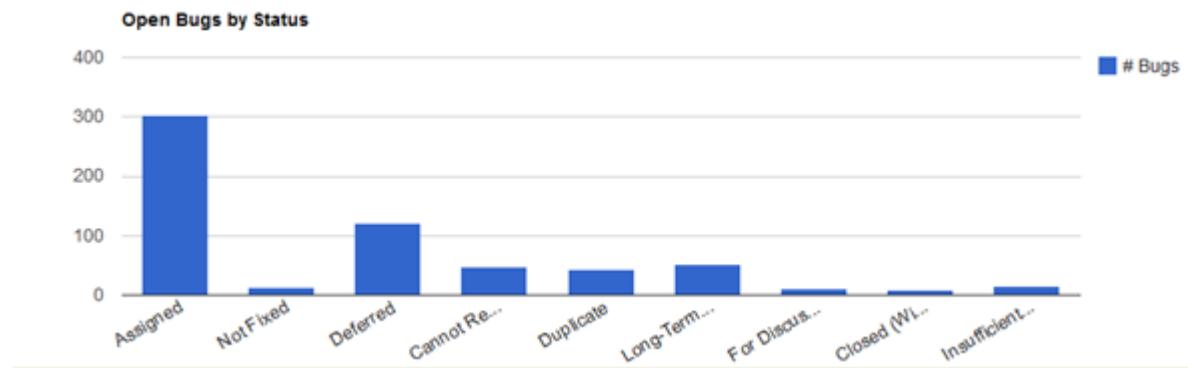
Additionally, you can right-click on the chart and select **Download** to save an XML-based image (.svg) of the chart.

The Administrative permission **Manage Charts** is required to create and modify charts.

An example FusionChart:



An example Google chart:



## Debugging Charts

Because Rollbase generates the underlying functionality for charts, they can be difficult to troubleshoot. A super-admin user logged into a customer tenant can enable logging of the underlying SQL queries, see [Enabling Logging for Charts and Views](#) on page 474. You can use the log file to see if the expected queries are being executed. Since logging can slow performance, it should only be enabled temporarily, for debugging purposes.

## Gauges

You can use gauges in Rollbase to visually represent a single numeric value computed using a JavaScript formula. Gauges resemble familiar devices such as a car's speedometer or gas meter, electronic scale, etc. Rollbase supports several different gauge types for various visual effects. Gauges can periodically and automatically refresh themselves without a complete page refresh.

## Creating and Configuring Gauges

To create a gauge:

1. Go to an object definition's details page and scroll down to the **Gauges** section. Click **New Gauge**.
2. Choose one of presentation styles shown below:



3. Select minimum and maximum values to represent the beginning and end of the gauge's display range. Rollbase renders the computed value by positioning the Gauge's arrow relative to these values. You can also optionally set lower and upper values inside the min/max interval. Rollbase

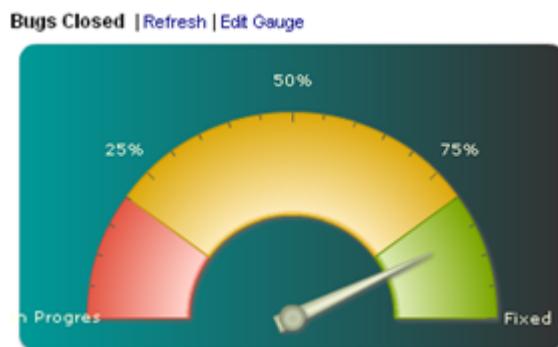
uses these values to visually indicate the Gauge's lower and upper threshold values (i.e. consider the red zone on an automobile's tachometer).

4. Specify the JavaScript formula to compute the Gauge's value. If you use the Gauge to visualize data for a particular record, use tokens from that record and related records. If you use the Gauge to visualize data from a list of records, use a loop to iterate over records, compute and return a final value. You can also use the Rollbase Query API in gauges. For more information about formulas, see [.](#)

## Using Gauges

As with other UI components, you can use the **Page Editor** to add one or more gauge components to a section on a generic, object, or view page. There is no further configuration necessary to start viewing your computed data with professional looking gauges.

You can place up to 3 Gauges in a row in a single section.



Using links above each gauge component, you can:

- Force a gauge component to refresh itself.
- Edit a gauge component.

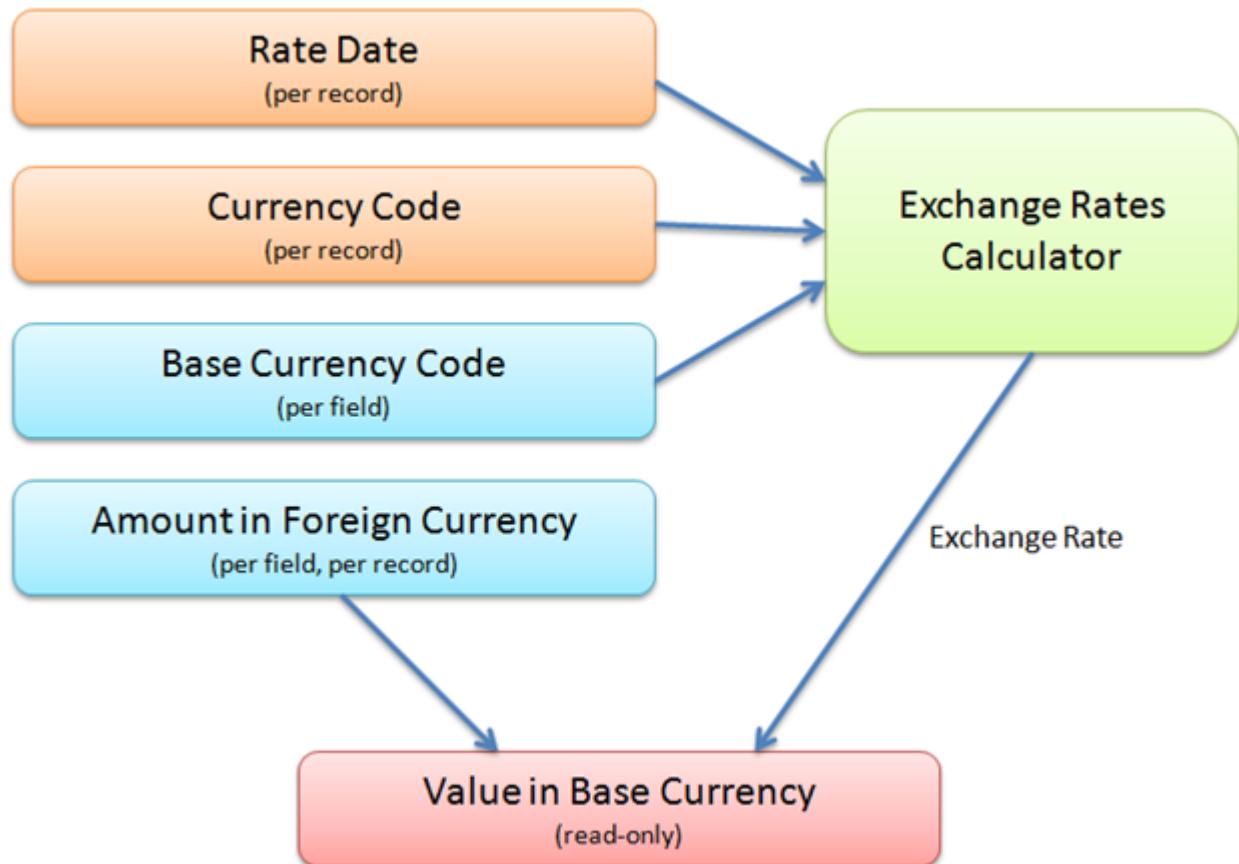
Note: The Administrative permission Manage Charts is required to create and modify gauges.

Warning: Avoid using template `#LOOP` in gauge's formula since that may lead to inefficient computations. Instead, try using group functions or the query API. For more information about formulas, see [.](#)

## Multi-Currency Support

Progress Rollbase supports conversion between monetary values in different currencies. It is common for businesses to keep their accounting books in one currency (let us call this the base currency), while sending and receiving money in several other foreign currencies. In addition, exchange rates between currencies tend to change over time. Rollbase provides a multi-currency framework for all customers.

The following illustrates Multi-Currency functionality:



Note: Although the Base-Currency field is read-only, you can use it in Formula fields and Triggers, as well as to calculate totals in List Views and Reports.

Important: Modifying an exchange rate on a particular date will not affect any Base Currency values that already have been calculated because, unlike Formula fields which are dynamically computed, Base Currency fields are calculated and stored in your database. However, when you edit and save an existing record, this value will automatically be updated using the currently available exchange rate corresponding to the assigned rate date.

To start utilizing multi-currency, follow these steps:

1. [Set Up Currency Codes](#) on page 215
2. [Set Up Exchange Rate](#) on page 216
3. [Enable Multi-Currency Attribute](#) on page 216
4. [Money Amounts in Base Currency \(optional\)](#) on page 216

## Set Up Currency Codes

On the Currency Codes page (Setup > Administration Setup > Currency Codes), enter a list of currency names and currency codes you would like to support. Mark the currency used in your bookkeeping as the default by prefixing it with the symbol {D}. For example:

{D}US Dollar|USD

---

**Note:** Currencies are essentially shared picklist items.

---

## Set Up Exchange Rate

On the Exchange Rates page (Setup > Administration Setup > Exchange Rates), for each relevant date, enter exchange rates as decimal numbers between your selected base currency (typically your bookkeeping currency) and the other supported currencies you defined above.



The screenshot shows a configuration interface for exchange rates. At the top, 'Base Currency' is set to 'US Dollar'. Below it, 'Rates on Date' is set to '05/25/2010'. The table lists exchange rates for five currencies:

Currency Pair	Rate
EUR / USD	1.2584
GBP / USD	1.4472
JPY / USD	0.011137
CAD / USD	0.941354
AUD / USD	0.8263

Note: By entering the exchange ratio between one unit of foreign currency and one unit of base currency, the inverse is handled for you automatically. For example, defining a EUR/USD ratio as x also defines an USD/EUR ratio as 1/x.

You can also use the SOAP API to store exchange rates (see [Rollbase SOAP Methods](#) on page 779). If you will be using multi-currency capabilities on a daily basis you might consider building an automated integration with another web service to retrieve exchange rates and populate them in Rollbase for you.

Note: To managing exchange rates user must have Administrator role or "Manage Exchange Rates" administrative permission.

## Enable Multi-Currency Attribute

Enable the Multi-Currency attribute on the desired Objects that adds two new fields:

- Currency Code, used to specify the type of currency for a particular record (e.g., Invoice)
- Rate Date, used to determine the date for the exchange rate calculation (current date by default). This field ensures that the exchange rate calculation for any given record does not change over time as the exchange rates change.

## Money Amounts in Base Currency (optional)

You can optionally create a Base Currency field corresponding to each new or existing Currency field as desired. In this case the Currency field holds the actual currency amount (determined by the Currency Code field at the record level as explained above) and the Base Currency field holds a read-only amount that is the currency amount from the Currency field automatically converted into the base bookkeeping currency. The Rate Date field, as explained above, determines the date of conversion.

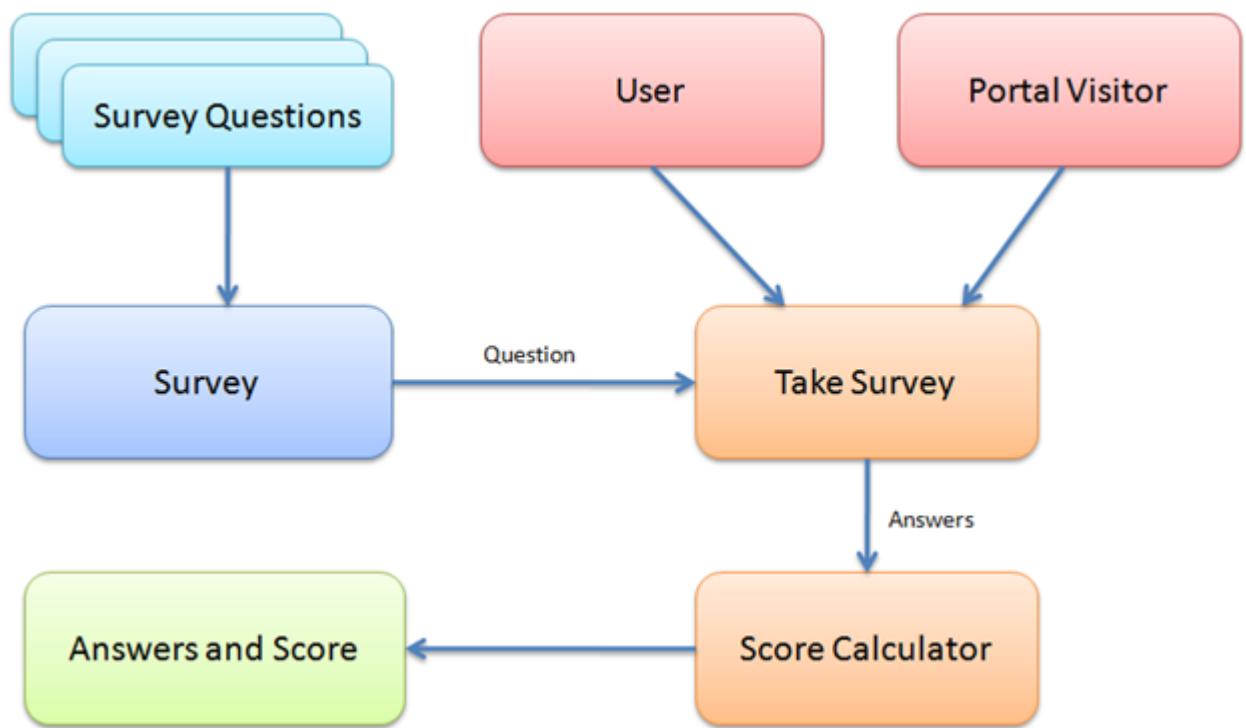
For example, an Invoice record has the following field values:

- Currency Code: British Pound
- Rate Date: 05/25/2010
- Amount (editable): 2000.00
- Base Amount (read-only): \$2,894.40

The exchange rate on 05/25/2010 is set to: GBP/USD = 1.4472. Rollbase calculates and displays the value for the Base Amount (which is configured to use USD) as \$2,894.40.

# Surveys and Quizzes

Surveys provide a way to quickly assemble groups of questions and process answers from users or portal users. Survey questions may include ranking criteria. This way surveys can be interpreted as online quizzes or tests. In more general terms surveys allow to associate different groups of fields with records of the same object. The diagram below illustrates the process of creating and taking a survey.



## Creating a Survey

First create an Object definition with the "Survey" attribute. By default a tab created for this Object includes one extra menu called "Questions".

Tip: If you enable the "Survey" attribute on an existing object you can add the "Questions" menu to its tab.

## Survey Questions Library

Survey Questions form a shared library that can be used by any survey of any object type.

Survey questions are grouped by categories. One category is available by default: Generic. You can create new categories and rename or delete existing categories.

Tip: You can only delete categories that have no questions associated with them.

Click "New Question" next to a particular category to create a new Survey Question. Next select the type of question:

- Text: Users can enter any combination of characters. You can optionally define an input mask to enforce certain kinds of input.
- Text Area: Users can enter multiple lines of text. You can optionally enable a rich text editor so users can enter text as formatted HTML.
- Picklist: Users can select a value from a list of values defined by you.
- Picklist (Multi-Select): Users can select any number of values from a list of values defined by you.
- Radio Buttons: Group of radio buttons with mutually exclusive selection
- Checkbox: Users can select a Checked (true) or Unchecked (false) value.
- Group of Checkboxes: Group of checkboxes with multiple selections
- Currency: Users can enter a dollar or other currency amount.
- Date: Users can enter a date or pick a date from a calendar popup.
- Date/Time: Users can enter a date and a time, or pick a date from a calendar popup (when a date is selected from the calendar popup, that date and the current time are entered into the Date/Time field).
- Email: Users can enter an email address which is checked to ensure that it is a valid format.
- Integer: Users can enter any number (without decimals).
- URL: Users can enter any website address.

Next, select properties for your question that can be different for different types. For instance, Text questions have the following properties:

- Question label
- Short label
- Category (the list of categories can be managed from the "Questions" menu)
- Check if this question must be answered
- Check if you want to publish this question in any application which includes the Survey object
- Size of HTML input box
- Keywords: you can specify keywords and their ranking in the range from -1000 to 1000. A score calculator will award scores for specified keywords in the answer.

Question Properties

Question Type	Text
Question Label	Last Job Title
Short Label	Job Title
Category	Generic
<input checked="" type="checkbox"/> This question must be answered	
<input type="checkbox"/> Publish this question as part of any application which includes Survey object	
Size	
Determines size of HTML input box	
You can specify list of keywords with rankings from -1000 to 1000 for a particular keyword: High#10 Use separate text line for each keyword.	
Keywords	Programmer#500 QA#700 Developer#500

## Adding Survey Questions

To create a particular instance of a Survey you need to create a record of a Survey object and give it a name. Then use the Questions section (created by default on each Survey object's View Page; use Page Editor to add this to a new section if you don't see it) to attach existing questions or to create new questions for this survey.

Questions   New Question   Attach Question   Reorder   Preview						
Action	Order No	Question	Category	Type	Is Copy	Updated By
Edit   Del   Remove	1	Last Job Title	Generic	Text	<input type="checkbox"/>	Admin
Edit   Del   Remove	2	Favorite Number	Generic	Integer	<input checked="" type="checkbox"/>	Admin
Edit   Del   Remove	3	Board Member	Generic	Checkbox	<input checked="" type="checkbox"/>	Admin
Edit   Del   Remove	4	Are you on Facebook?	Personal Question	Checkbox	<input type="checkbox"/>	Admin

Using links in the section header you can:

- Create a new Question.
- Select from existing questions in a popup window and them (or a copy of them) to the survey.
- Reorder questions in your survey.
- Preview survey questions as they'll appear to the user taking the survey. Please note that questions are grouped by categories and ordered by the selected order number.

Tip: You can attach questions from the library directly or create a copy (clone) of each question. In the latter case if you modify the selected question it will not affect the version in the Library.

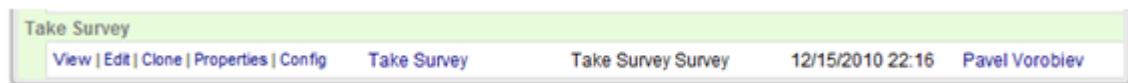
From the "Actions" column in the list of questions you can:

- Edit a survey question.
- Permanently delete a question from all surveys.
- Remove a question from the current survey.

## Survey Pages and Links

When you create an object with the "Survey" attribute Rollbase creates a special category of pages for that object called "Take Survey". This page can be used to answer questions attached to a particular survey and can be edited and cloned just like any other Rollbase page.

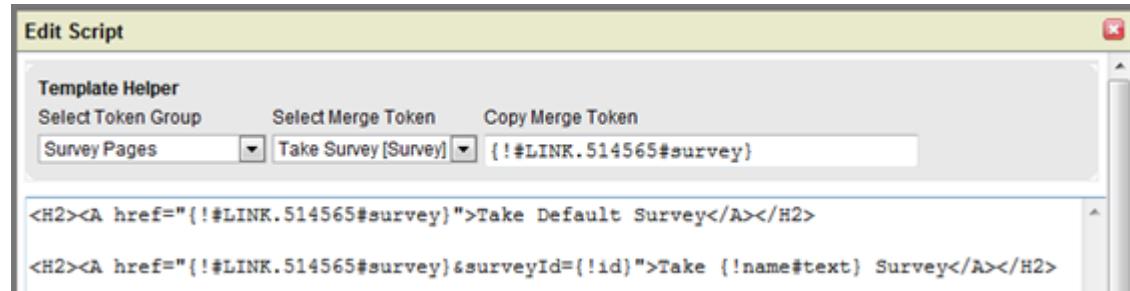
The URL to a Take Survey page may include the ID of a particular survey: &surveyId=123456 If that ID is not included the default Survey record (selected in the Config page) will be used. If this URL parameter is not included - the system will use last survey record in scope (being viewed or accessed).



You can access pages of this type from an object definition's page. Click on the "Config" link to configure page-level settings:

- Number of columns to display questions in (1, 2, or 3)
- Default Survey record used for this page

Now you can add a URL to your "Take Survey" page to any template on an application or portal page using the "Surveys" section in the Template helper. The "Survey Pages" group in the Template helper provides you with a merge field token that will be resolved into the URL of the Survey page. You can use this URL in links or in an HTML button control.



The "Survey Answers" component lists all surveys taken. For each survey it displays a list of questions and answers provided. If ranking criteria was set for a particular question, the system also calculates and displays an assigned score along with maximum and minimum possible scores.

Tip: Minimum score is only used when negative ranking was specified. Otherwise the minimum score is 0.

The object fields "Survey Score" and "Rank" (i.e. score as a percentage of maximum score) can be used to set up triggers (i.e. business logic and automation) related to surveys.

## Taking a Survey

Using the right link (as described above), an application user or portal user can navigate to "Take Survey" page. There he/she will be presented a list of selected questions grouped in sections by category.

Survey: Sample Survey

Generic

Last Job Title  Error: Last Job Title must be specified

Last Salary

Red = Required Information

Personal

Are you on Facebook?

Submit Cancel

Answers to survey questions are subject to validation rules similar to those for Object Fields: required questions must be answered, integer questions must be parsed as integers, etc.

Finally the results of a survey are recorded and can be viewed along with the record of each Survey Taker. On the record View page you will find a "Survey Answers" component (use the Page Editor to add it if it is not automatically added to your View page).

Survey Answers					
Order No	Question	Answer	Max	Min	Score
1	Last Job Title	QA Engineer	1700	700	
2	Are you on Facebook?	<input checked="" type="checkbox"/>	200	200	
3	Last Salary	\$100,000.00			
					Total: 900

## Using Surveys on Portals

Surveys in portals can be a useful tool to, for example, collect customer feedback. To add surveys to your portal:

- Make sure that your "Portal User" object also has the "Survey Taker" attribute.
- Create a Survey object, survey questions and survey record (see above).
- Use the Page Editor to add the "Survey Answers" component to the New and Edit pages of your Portal User/Survey Taker object.
- Use the "Config" link on that Survey Taker page to select the Survey record to be used by that page.

Now when a visitor is created or updated, the portal page will display a list of survey questions. Answers to these questions will be associated with the visitor record and available on the View page.

Tip: You can include a desired survey ID directly into the link to a survey page by adding the following additional URL parameter: &surveyId=12345



---

# 6

## Customizing the User Experience

---

You can customize the user experience by controlling what users see on app pages and how they navigate through the application. Additional files required for customization, such as images and cascading style sheets, can be uploaded as Hosted Files. Portals allow you to create a user interface and host it yourself.

For details, see the following topics:

- [Pages, the Page Editor, and Grid Controls](#)
- [Working with Views](#)
- [Programmatic Client-side Customization](#)
- [Rollbase Portals](#)
- [Hosted Files](#)

## Pages, the Page Editor, and Grid Controls

This section explains the concepts behind and the tools available for working with the Progress Rollbase user interface (UI) components that combine to form Rollbase application pages. It explains how pages, sections, tabs, cells, and fields work together to form a functioning UI layer. It also guides you through creating and modifying pages using the **Page Editor** and discusses the advanced grid component, which allows you to create and edit data in groups of related records while working with a parent record, such as line items or child records.

## Editing Pages

The top right of corner of the main Rollbase UI screen contains the following buttons: **Design This Page**, **Print**, **PDF** and a display of the amount of time it took the page to render.



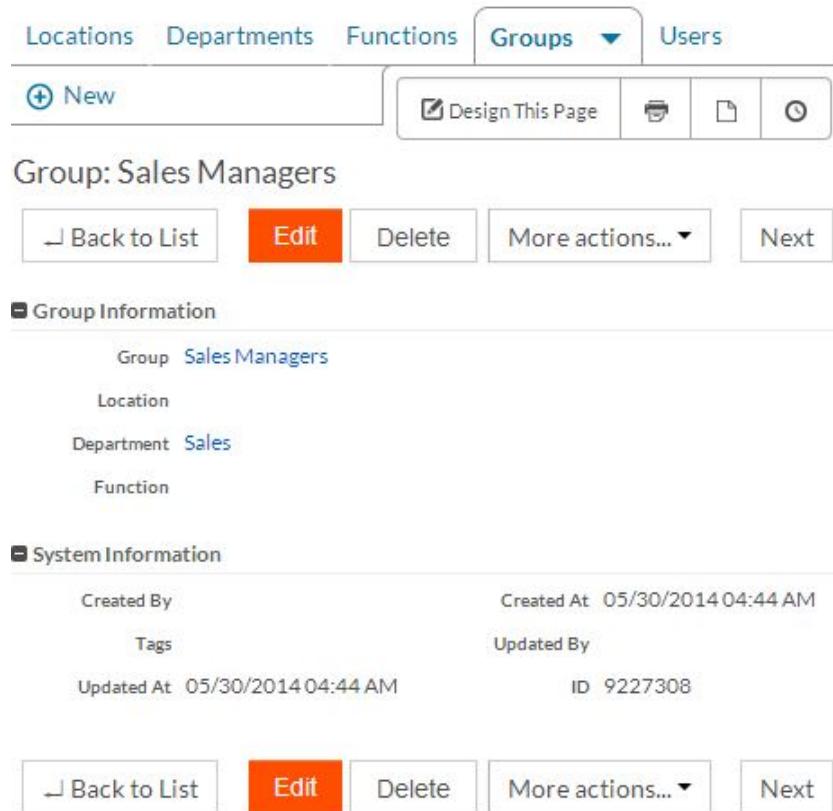
The **Design This Page** button allows Administrators to make changes to the current page using the real-time WYSIWYG (what you see is what you get) **Page Editor**. The **Print** icon displays a printable version of the current page without sidebars, headers, footers, or active links. The **PDF** button creates a PDF document of the printable version of the current page.

When you click on an Application link in the Sidebar, the main Rollbase UI screen displays a view Page with that Application's most recent records, by default. Pages can contain a number of components and Administrators can control which components each Page contains, as well as the layout of components within each Page and the specific configuration of each component.

For example, the screenshot below shows a page containing a View component for navigating records in the **Groups** tab:

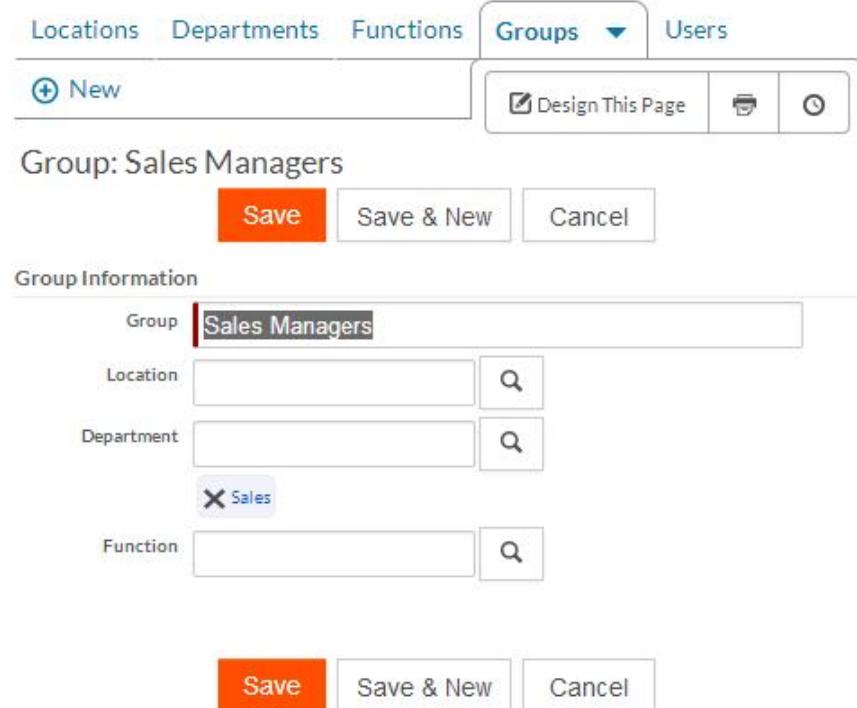
Action	Group	Location	Department	Function	Updated At	Updated By
<a href="#">Edit</a>   <a href="#">Del</a>	Sales Managers	Sales			05/30/2014 04:44 AM	
<a href="#">Edit</a>   <a href="#">Del</a>	US Managers	United States			05/30/2014 04:44 AM	
<a href="#">Edit</a>   <a href="#">Del</a>	Full Access Group				05/30/2014 04:44 AM	

Administrators can create and edit Views for any object as needed. In this example, clicking on a **Groups** record takes you to a view page where you can see field data associated with that record, along with related views showing information about any records related to that record:



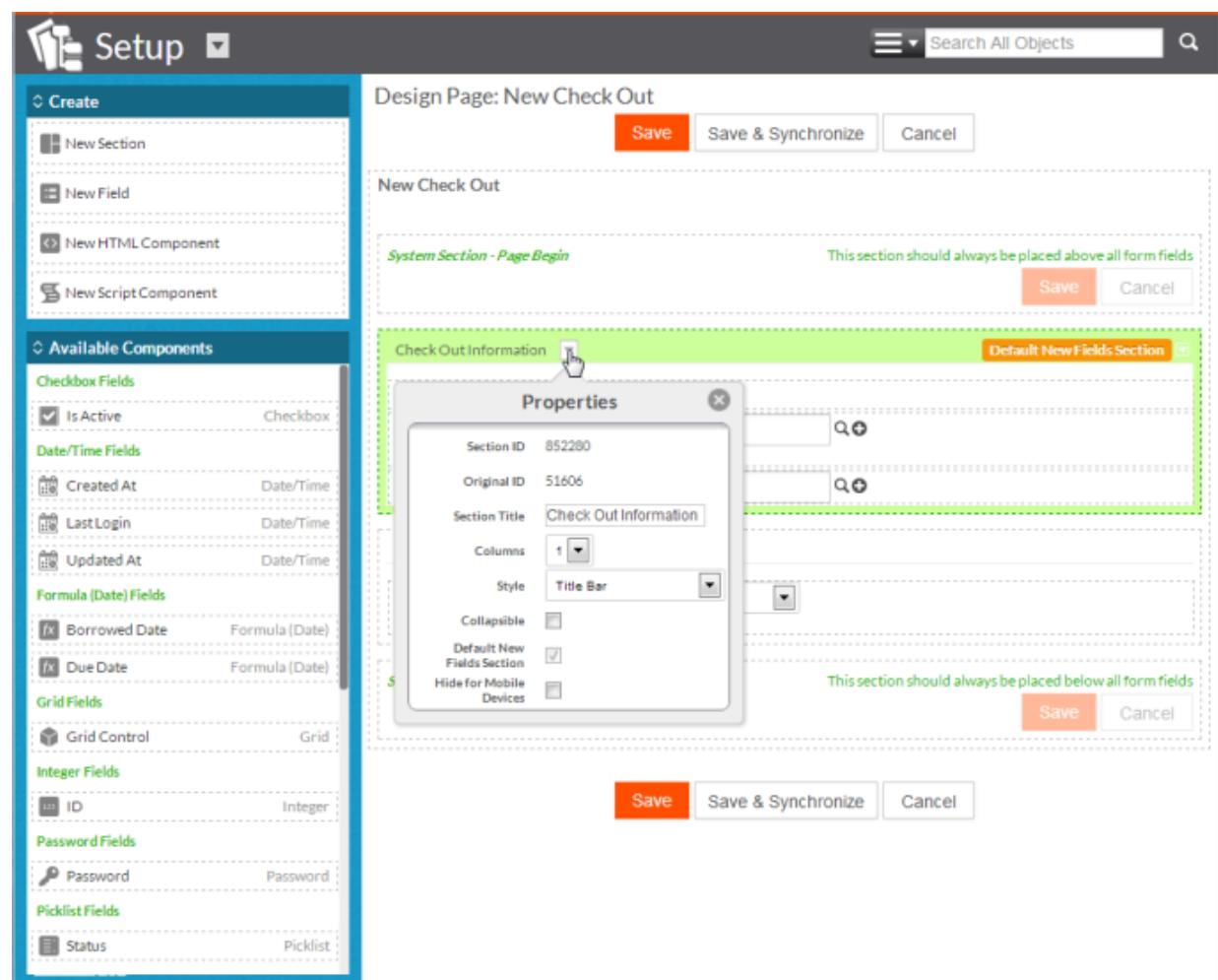
The screenshot shows a 'Groups' page with a navigation bar at the top. The 'Groups' tab is selected. Below the navigation bar, there is a 'New' button and a toolbar with 'Design This Page', 'Print', 'Copy', and 'Delete' icons. The main content area displays a group record for 'Sales Managers'. The record includes sections for 'Group Information' (Group: Sales Managers, Location: Sales, Department: Sales, Function: ), 'System Information' (Created By, Created At: 05/30/2014 04:44 AM, Tags, Updated By, Updated At: 05/30/2014 04:44 AM, ID: 9227308), and a toolbar with 'Back to List', 'Edit', 'Delete', 'More actions...', and 'Next' buttons.

To edit a record, click **Edit**. The following page appears:



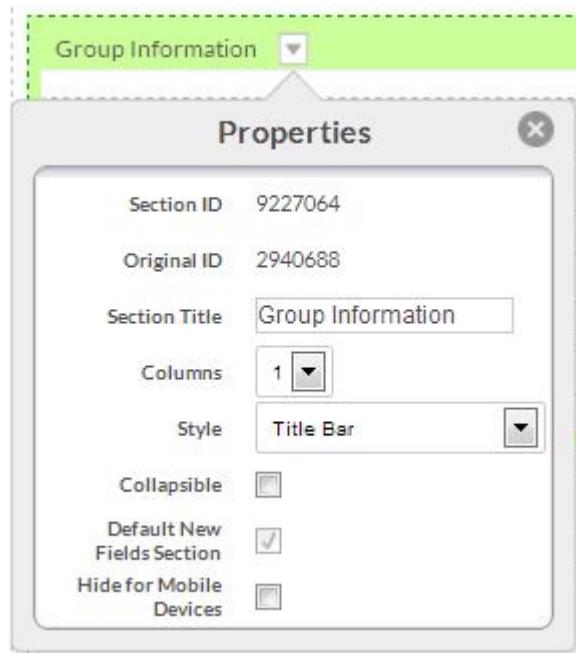
The screenshot shows the 'Edit' page for the 'Sales Managers' group record. The top navigation bar and toolbar are identical to the previous screenshot. The main content area shows the 'Group Information' section with the 'Group' field set to 'Sales Managers'. Below this are fields for 'Location' (with a search icon), 'Department' (with a search icon), and 'Function' (with a search icon). The 'Function' field has a value of 'Sales' with a delete icon. At the bottom of the page is a toolbar with 'Save', 'Save & New', and 'Cancel' buttons.

Administrators can customize the layout and content of any page by clicking **Design This Page** in the navigation bar to invoke the **Page Editor**. The following example shows a new record page. You can drag and drop components to move them around the page, edit component properties and create new components on the fly.



## Adding Columns and Changing Alignment

Sections on the page are typically aligned from top to bottom in one column. However, generic and list pages have an option to align sections into one or two columns. This option can be enabled in the **Page Editor** by selecting the page title and editing the **Columns** property.



If the page has more than one column, its sections will include an **Alignment** option, allowing you to select **Left** (the default), **Right**, or **Center**. Using these options, you can create richer page layouts, for example, to place charts side-by-side on a dashboard page.

## Buttons

You can add Buttons to View, New Record, Edit, or Status Change pages. These buttons will appear in row next to "Edit" or "Save" button on top and bottom of the page. When a Button is clicked it can perform one of the following actions:

1. Run JavaScript
2. Open URL in new browser's window

To manage Buttons use the Buttons section on Object View page. To create new Button specify the following info:

- Display Label
- Button's name (used as HTML tag "name")
- Action to perform
- JavaScript template or URL template. Both templates may include record's tokens to be replaced with real values at runtime.
- Check pages to which you want to add new Button.

To add/remove Button from pages use the Page Properties link.

## Using Grid Controls to Manage Multiple Records

Grid controls allow users to create, update and delete a group of related records while editing a parent record. Grids enforce user permissions. For example, a user with only view permission would not be able to create or update records.

You can add a grid to the **New**, **Edit**, or **Status** pages of:

- A parent object that has a one-to-many relationship with a child object, such as a purchase order might have to line items. The grid will display the related records on the "many" side.
- Both objects in many-to-many relationships.

The following screen shot illustrates a grid in a room reservation system. Fields that are not read-only can be edited in place.

Quantity	Furniture	Description	Size
24	Bistro Chairs	Plastic molded	
2	Glass Buffet	Modern design, countertop height	50" X 24"
6	Glass Tables	Seats 4	
1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square

You can add grid components when creating a relationship, or you can add them to existing pages with the **Page Editor**. A grid and a lookup field for the same related records cannot exist on the same page. To add a grid to a page with a lookup field, you will need to remove the lookup field.

## Adding a Grid Control with the Page Editor

To add a grid to existing object pages, follow these steps:

1. Open the page in the **Page Editor**.
  - a) For example, to add a grid to the **New** page, navigate to the object tab and click **New**.
  - b) Click **Design This Page**:



The **Page Editor** opens the definition of the **New** object page.

2. If the page already contains a lookup field for this relationship, remove it:
  - a) Right-click the lookup field.
  - b) Select **Remove**.
  - c) Remove the empty section.

3. From the **Create** pane, drag a **New Section** and drop it in the location where you want the grid to display.
4. From the **Available Components Grid Fields** section, select **Grid Control**, and drag and drop it in the new section as shown in the example below:

Page Editor: New Room

The screenshot shows the Page Editor interface for a 'New Room' form. At the top, there are three buttons: 'Save' (orange), 'Save & Synchronize' (light blue), and 'Cancel' (light blue). The main content area is titled 'New Room'. It contains several sections: 'System Section - Page Begin' (with a note: 'This section should always be placed above all form fields') and 'System Section - Page End' (with a note: 'This section should always be placed below all form fields'). The 'Room Information' section includes fields for '(Record Name) Room' and '(Lookup (Reservation)) Reservation'. A 'Default New Fields Section' is shown below. A 'New Section' is inserted, containing a '(Grid) Grid Control' component, which is highlighted with a green background. The 'Location and Address Information' section follows, containing fields for 'Address 1', 'Address 2', 'City', 'State/Province', 'ZIP/Postal Code', and 'Country'. The 'Grid Control' component is a rectangular area with a green header and a dashed border, indicating it is a selected or active component.

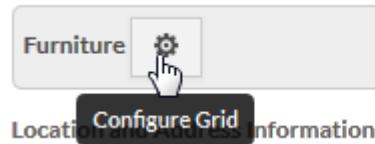
5. Optionally, set a name for the section by selecting it in the right pane and editing the **Section Title** in the **Properties** pane at the top left.
6. Click **Save** or **Save and Synchronize**:
  - **Save** applies your changes to the current page.
  - **Save and Synchronize** gives you the option to apply the changes you just made to other pages for this object.

Next, follow the steps described in [Configuring a Grid Component](#) on page 229

## Configuring a Grid Component

To configure a grid component, follow these steps:

1. Navigate to the page containing the grid and click **Configure Grid**:



2. From the **Relationship** drop-down, select the relationship for the records you want the grid to display.
3. Click **Next**.
4. From the list of available columns and formula fields, select the fields you want to appear as grid columns and use the arrows to move them to the **Selected Columns** list.

Although you can use most field types in a grid component, there are some limitations. For example, you cannot use filtered lookup fields.

5. Optionally, reorder the columns using the up and down arrows.
6. Click **Next**.

The last page of the wizard displays:

Room: Configure Grid of Related Furnishings

Fields Properties				
Field Label	Data Type	Integration Name	Required In Grid	Read Only
Quantity	Integer	quantity	<input type="checkbox"/>	<input type="checkbox"/>
Furniture	Record Name	name	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Description	Text Area	description	<input type="checkbox"/>	<input type="checkbox"/>
Size	Text	size	<input type="checkbox"/>	<input type="checkbox"/>

**Grid Totals**

Select the fields to total this grid by.

Total By	-None-	<input type="button" value="▼"/>	<input type="button" value="▼"/>
And By	-None-	<input type="button" value="▼"/>	<input type="button" value="▼"/>
And By	-None-	<input type="button" value="▼"/>	<input type="button" value="▼"/>

**Initial Rows Sorting**

Select sorting criteria for existing records. This does not imply dynamic sorting for updated or newly created records.

Sort By	Furniture	<input type="button" value="▼"/>	Ascending	<input type="button" value="▼"/>
---------	-----------	----------------------------------	-----------	----------------------------------

**JavaScript Event Handlers**

Tip: Using this feature you can assign JavaScript event handling code to be executed when rows of this grid are created, updated, or deleted. Use `##` symbol as a placeholder for row's number and `##` symbol as a placeholder for Grid number parameter user by API. [Learn more about JavaScript event handlers...](#)

Grid Number: 0
onCreate = <input type="text"/>
onUpdate = <input type="text"/>
onDelete = <input type="text"/>

7. On the last screen, choose from the following options:

- Check **Required in Grid** for fields you want users to enter a value.
- Check **Read Only** for fields that users should not update. This has no effect on required columns.

- Optionally, select the columns to total. This option is only available for numeric fields.
- Optionally, specify sorting for columns.
- Optionally, specify custom JavaScript code to be executed when a grid row is added, updated, or deleted. For more information, see [Grid Control Examples and API](#) on page 647.

8. Click **Save**.

## Using a Grid Component

As shown in the following screen, grid components provide controls for adding, deleting and editing records.

Quantity	Furniture	Description	Size
24	Bistro Chairs	Plastic molded	
2	Glass Buffet	Modern design, <del>countertop height</del>	50" X 24"
6	Glass Tables	Seats 4	
1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square

The following controls for creating, deleting and editing are available to users with the appropriate permissions:

- To add a new related record:
  1. Click **Add *ObjectName***
  2. Fill in some of the empty fields; Rollbase will ignore an empty line.
- To delete a record, click **Delete**.
- To modify a field, enter the new data.

Rollbase saves changes to each related record in the grid when you save the page. If an error occurs in one of the rows, Rollbase displays one or more error messages below the grid component, as shown below:

Quantity	Furniture	Description	Size	
24	Bistro Chairs	Plastic molded		<a href="#">Delete</a>
2	Glass Buffet	Modern design, countertop height	50" X 24"	<a href="#">Delete</a>
6	Glass Tables	Seats 4		<a href="#">Delete</a>
1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square	<a href="#">Delete</a>
				<a href="#">Delete</a>

Error: Row 5 Furniture must be specified

After a successful save, a message in the Rollbase header describes how many line items were created, updated and deleted:

Room "Atrium Social Corner" has been updated.

Furnishings: created: 1, updated: 0, deleted: 0

This topic described basic grid functionality. You can add logic as described in [Grid Control Examples and API](#) on page 647.

## Embedded Quick Create

This component allows a user to create a single related record while creating a core record. Unlike grid components, this embedded panel uses the full quick create page and renders all fields from that page as a part of new record page. This way, fields from core and related objects are essentially mixed on one page.

To use embedded quick create:

- Use the **Page Editor** on a new record page and add the **Embedded Quick Create** component to a new section or in an existing section.
- Click **Configure now** to select options for the component:
  - **Record to be Created** - The related object for which to create the record.
  - **Using Page** - The **Quick Create** page to use. You can create a special copy of the page.
  - **Component Name** - A unique name for the component. In the page's HTML, this name will be appended to all field names used by this component.

New Employee: Configure "Embedded Quick Create" Component

Configure "Embedded Quick Create" Component

Red = Required Information

Record to be Created: Group [empl\_group1]

Using Page: Quick Create

Component Name: GROUP

Save Cancel

- On the new record page, you'll see all of the sections and fields from the selected **Quick Create** page rendered as part of the main page:

New Group

Group Information

Group Name: Eng 1

Group Code: ENG\_1

Create New Employee

Last Name: Bill

First Name: Smith

Salary:

Hire Date: 08/01/2012

The embedded quick create component performs the same validation operations as a regular **Quick Create** page. The resulting record will have a relationship with the newly created core record.

Important: Due to system limitations only fields available for grid components can be rendered by an embedded quick create component.

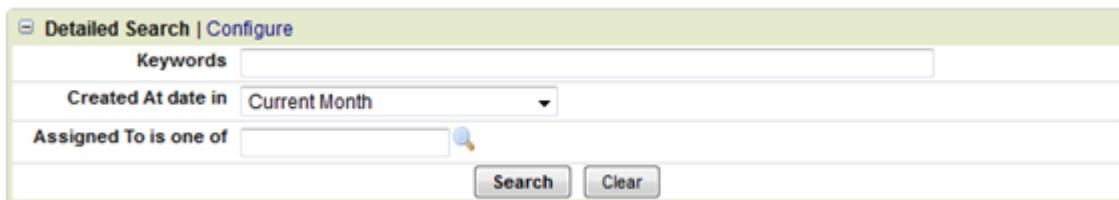
## Detailed Search

The detailed search component allows the filtering of records by a set of preconfigured fields and operands (equals, greater than, etc.).

Use the **Page Editor** to add a **Detailed Search** component to a generic, list, or search results page. Then, configure the detailed search component by selecting:

- Which search results page to use to display results (if you have several pages)
- How many columns use to arrange filters
- Whether or not to hide the keyword search text box
- Whether to filter results by date intervals
- Whether to use AND (all fields) or OR (any field) logic
- Fields and available operands to appear in the detailed search component
- For text boxes: the size in columns (Defaults to 30)
- For text multi-select picklists: the size in rows (Defaults to 4)

The configured component allows users to quickly filter records by selected values:



## Customizing the Header and Footer

You can customize the look of application pages by adding a custom header and/or footer, as shown in the following screen.

The screenshot shows a web application interface for a library system. At the top, there is a navigation bar with links: Titles, Users, Reports, Comments, Library Members, Check Outs, and Edit this Page. Below the navigation is a blue header bar with the text 'Great Reads Check Out System' and a graphic of books. The main content area displays a table of check-out records. The table has columns for Action, Transaction#, Member, Borrowed Date, Due Date, and Titles. Two rows of data are shown, each with a 'View | Edit | Del' link. The footer of the page contains the text 'May your days always have happy endings!'.

If you want to use images or refer to a CSS, you should upload those as hosted files first. Follow these steps to modify the header and footer:

1. Navigate to the application definition. For example,
  - a) From the drop-down menu of applications, select **Setup Home**.
  - b) In the **Application Setup** section, click **Applications**.
  - c) Click the name of the application you want to modify.
2. From the **More Actions** menu, select **Header And Footer**.
3. Enter your custom style information in **HTML Header** and **HTML Footer** boxes.
4. Click **Save**.

For example, the custom header and footer shown above uses a hosted graphic and locally declared styles. The **HTML Header** defines `my-header` and `my-footer` classes as follows:

```

<style>
.my-header {
background: #eee;
margin-bottom: 10px;
padding: 10px;
margin-bottom: 25px;
font-size: 25px;

```

```

font-weight: bold;
padding: 35px;
border-radius: 5px;
background: #00b7ea;
background: -moz-linear-gradient(top, #00b7ea 0%, #009ec3 100%);
background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#00b7ea), color-stop(100%,#009ec3));
background: -webkit-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
background: -o-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
background: -ms-linear-gradient(top, #00b7ea 0%,#009ec3 100%);
background: linear-gradient(to bottom, #00b7ea 0%,#009ec3 100%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#00b7ea',
endColorstr='#009ec3',GradientType=0 );
color: #fff;
}
.my-footer {
background: #eee;
margin-bottom: 10px;
padding: 10px;
margin-bottom: 25px;
font-size: 12px;
font-weight: bold;
text-align: center;
padding: 15px;
border-radius: 5px;
color: #fff;
background: #45484d; /* Old browsers */
background: -moz-linear-gradient(top, #45484d 0%, #000000 100%); /* FF3.6+ */
*/
background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#45484d), color-stop(100%,#000000)); /* Chrome,Safari4+ */
background: -webkit-linear-gradient(top, #45484d 0%,#000000 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top, #45484d 0%,#000000 100%); /* Opera 11.10+ */
*/
background: -ms-linear-gradient(top, #45484d 0%,#000000 100%); /* IE10+ */
background: linear-gradient(to bottom, #45484d 0%,#000000 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#45484d',
endColorstr='#000000',GradientType=0 ); /* IE6-9 */
}
</style>
<div class="my-header">
<img src='!#HOSTED_FILE.53382#url}' border='0' align='absleft' />
Great Reads Check Out System
</div>

```

The following, in the **HTML Footer** section, defines the footer:

```

<div class="my-footer">
May your days always have happy endings!
</div>

```

## Working with Views

Views are result sets. They are key components of an object definition because they determine how a user can find and navigate through records. This section discusses how to create and configure views, and define grouping, sorting, totaling and filtering behaviors (including dynamic filtering, expressions and exporting).

To troubleshoot views, a super-admin can enable logging of the SQL queries and results, see [Enabling Logging for Charts and Views](#) on page 474

## Creating and Editing Views

Administrators can create and edit views to return specific records. Follow these general steps to create a view:

1. Navigate to the object definition for the type of records you want to display in the view. For example, click the drop-down menu on the object tab and select **Object Definition**:

2. In the ribbon listing object components click **Views**.
3. In the **Views** section, click **New View**.

The **New View** screen displays. The **View Name** section contains the following:

**View Name**

Provide a name for this view.

View Name

Hide this View from View Selector

Hide count of total records for this View

Private  (Private Views are only available to their creator)

4. Enter a name, and choose from the following options:
  - **Hide this View from View Selector** If the selector is disabled, a view can be used as a default view.
  - **Hide count of total records for this View** Users will see better performance if this information is not displayed for complex views with many records.
  - **Private** Private views cannot be published as part of an application or be used in other components such as triggers and templates.
5. In the **Columns** section, use the arrows to select the fields that will display as columns:

Columns

Select the fields to display in each column of this view.

Available Columns

- City
- Country
- Created At
- Created By
- Devices
- ID
- Phone Number
- State/Province
- Street Address 1
- Street Address 2
- Updated At
- Updated By
- ZIP/Postal Code
- Furnishings

Selected Columns

- Room
- Reservation

▲ ▼ ▲ ▼

Show "Actions" column in this view

6. In the **Sorting and Grouping** section, optionally select whether to group and/or sort records in the view:

Sorting and Grouping

Select the fields to sort and group this view by.

Group By	--None--	<input type="button" value="▼"/>		<input type="button" value="▼"/>
Sort By	--None--	<input type="button" value="▼"/>		<input type="button" value="▼"/>
...then by	--None--	<input type="button" value="▼"/>		<input type="button" value="▼"/>
...then by	--None--	<input type="button" value="▼"/>		<input type="button" value="▼"/>

Disable dynamic records sorting by clicking on the links atop of columns

7. For objects with numeric fields, optionally use the **Totals and Subtotals** section to define columns for which you want totals calculated:

<b>Totals and Subtotals</b>			
Select numeric fields to page total, subtotal (by groups - if selected), grand total (for all pages) this view by. Select any other field to count number of records. Important limitation: grand total cannot include formula fields.			
Total By	--None--	<input type="button" value="▼"/>	<input type="button" value="▼"/>
And By	--None--	<input type="button" value="▼"/>	<input type="button" value="▼"/>
And By	--None--	<input type="button" value="▼"/>	<input type="button" value="▼"/>
<input type="checkbox"/> Hide group subtotals for this view. <input type="checkbox"/> Hide page totals for this view. <input type="checkbox"/> Hide grand totals (on all pages) for this view.			

8. In the **View Filters** section, choose the fields and operation for filtering. For example, the following filter causes the view to display only rooms that are not reserved:

View Filters | Learn more

Date Field: Reservation is not empty

Filter Conditions: All (AND)

9. In the **Permissions** section, specify which roles and users can see this view.
10. Click **Save**.

## Adding Columns

The next step is to define which fields should appear as columns in your view. Use the checkbox labeled **Show Actions column in this View** to specify whether or not you want users who can access this view to see the standard **Actions** column that provides links to Edit and Delete records. For example, you may want to hide this column in Views exposed in external Portals.

Columns

Select the fields to display in each column of this view.

Available Columns

- Account
- Created At
- Created By
- Description
- ID
- Lead Source
- Opportunity Name
- Priority
- Private
- Probability
- Tags
- Type

Selected Columns

- Opportunity
- Amount
- Expected Revenue
- Close Date
- Opportunity Owner

Show "Actions" column in this view

When you create a new View, Record Name Field is added by default to Selected Columns.

Note: If you do not include the Record Name Field in the View, Rollbase will display a warning message asking if you are sure you want to not include it. The Record Name Field provides a link to drill down to the record's detail Page that is why it is typically not a good idea to exclude this from your Views.

Warning: You can include templates and formula fields into your View. However please be aware that this may result into decreasing performance due to high volume of server-side calculations.

## Grouping and Sorting

Once you have chosen columns, you can specify grouping and sorting behavior. Second sorting criteria will apply when two records have the same column selected for first sorting. You can sort by up to three columns. You can also select direction of grouping and sorting: ascending or descending.

Optionally you can disable sorting in a view by clicking on the link atop of columns.

Group By	Company	Ascending
Sort By	Lead	Descending
...then by	--None--	
...then by	--None--	

Disable dynamic records sorting by clicking on the links atop of columns

## Totalling Columns

You can total on up to three columns. For numeric columns you can select options:

- Total: sum of all non-null values
- Average: total divided by number of records with non-null values
- Count: number of records
- Min: minimum of all non-null values
- Max: maximum of all non-null values
- Variance: statistical variance of all non-null values
- Standard Deviation: Square root of Variance

For all non-numeric columns the only available option is Count.

When totaling is enabled on one or more columns, two extra rows become available at the bottom of the View displaying the following Fields:

- Calculated sub-total for the set of records shown on the current group (if grouping is enabled).
- Calculated amount for the set of records shown on the current Page.
- Sum of amounts for all records in this View on all Pages (grand total). This sum is not available for formula fields.

Note: If current group spans across more than one page, sub-total is calculated for all pages.

You can turn off any of these three ways of totaling for the View you are editing.

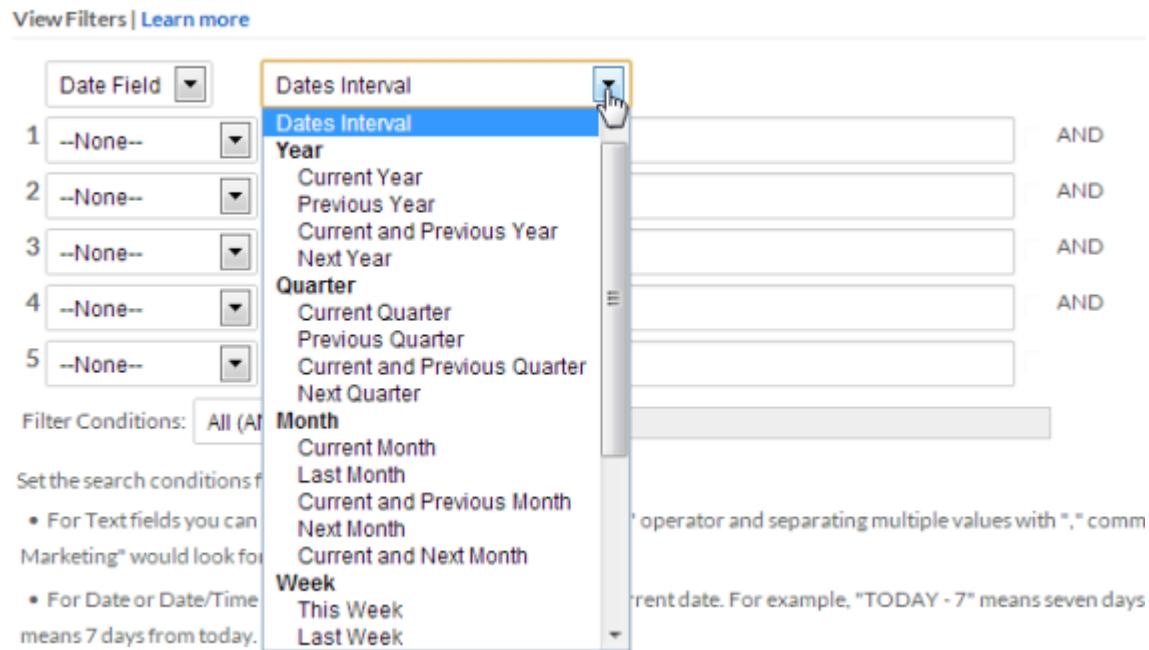
Total By	Num1	Total
And By	--None--	
And By	Num3	Average

Hide group subtotals for this view.  
 Hide page totals for this view.  
 Hide grand totals (on all pages) for this view.

Important: Due to performance considerations, a grand total value will not be calculated for formula fields. Totaling by formula fields is only available for subtotals and Page totals.

## Filtering by Dates

You can easily filter records used in Views, Reports, and Charts by interval of dates. From drop-down list select Date or Date/Time field which exists in your object. From second drop-down select interval relative to current date: current year, next quarter, current week etc. That will create filter for View etc. output which will be applied before any other filter you may create.



## Filter Criteria

When defining Views, you can design detailed filter criteria that provides a way to narrow in on specific data you are interested in seeing in that View. This filtering mechanism is the same as is used by Object-specific search and Reports to determine which records are shown in the output.

The filter mechanism allows you to design up to five filters by selecting a field, an operator and a value.

For example, in the filter shown below, the first filter is specifies records where an account's annual revenue exceeds ten million dollars. The second filter specifies accounts for which an opportunity has been identified.

View Filters | Learn more

The screenshot shows a 'View Filters' interface with the following configuration:

- Date Field:** Date Field (dropdown)
- Dates Interval:** Dates Interval (dropdown)
- Conditions:**
  - 1 Annual Revenue greater than 10,000,000.00 AND
  - 2 Opportunities is not empty AND
  - 3 --None-- AND
  - 4 --None-- AND
  - 5 --None-- AND
- Filter Conditions:** All (AND) (dropdown)

You can use several different operators, depending on the selected field type. They are: equals, not equal to, less than, greater than, less or equal, greater or equal, starts with, contains, does not contain, is one of, is not one of, is empty and is not empty.

To filter by date columns you can use special tokens derived from current date.

Token	Description	Example
TODAY	12:00 AM of current date	09/28/2011
WEEK	1st day (Sunday) of current week	09/25/2011
MONTH	1st day of current month	09/01/2011
QUARTER	1st day of current quarter	07/01/2011
YEAR	1st day of current year	01/01/2011

You can add/subtract integer number from token to shift on number of periods. Examples:

- To filter by current week (from Sunday to Saturday) use conditions: date>=WEEK AND date<WEEK+1
- To filter by previous month use conditions: date>=MONTH-1 AND date<MONTH
- To filter by previous and current year use conditions: date>=YEAR-1 AND date<YEAR+1

In addition, some fields, such as Dates and User lookups, have special tokens that can be used in filters:

- For Text fields, you can create "or" filters by using the "is one of" operator and separating multiple values with "," (commas). For example Sales, Marketing would look for either Sales or Marketing.
- For Date or Date/Time fields, use TODAY to represent the current date. For example, TODAY - 7 means seven days ago and TODAY + 7 means 7 days from today.
- For Date or Date/Time fields, use tokens YEAR, QUARTER, MONTH and WEEK to refer to the current year, quarter, month and week respectively.
- For User Lookup fields, use CURRENT\_USER to represent the current user.

Warning: Do not use quotation marks around keywords or phrases.

Note: The precise meaning of the TODAY token is 12PM+1ms of the current day in the time zone of the current user for Date/Time fields; for Date fields, either the beginning (for Greater Than operation) or end (for Less Than operation) of the current day.

The filter conditions parameter allows you to control how each of the filters are used to determine the output of the View:

- All (AND): Selecting this condition means that all filter conditions must be met in order for a record to display in this View.
- Any (OR): Selecting this condition means that at least one of the filter conditions must be met in order for a record to display in this View, but it doesn't matter which one.
- Expression: If the above conditions are not flexible enough, you can select Expression to define your own conditions.
- Formula: A flexible way to filter records using any combination of data columns and special functions. This option is not available for OpenEdge Services.
- Search criteria: To filter using OpenEdge ABL Syntax. This option is available only for the OpenEdge services. For information on filtering by search criteria, see [Filtering by Search criteria](#) on page 243.

Filter expressions allow you to specify how filters should be evaluated together to determine whether or not a record should be shown in the View. To define an expression, use the numerical value of the filters (1 through 5) along with AND, OR, and NOT keywords and parenthesis "(" ")" to group sub-expressions together. Always make sure that you have an equal number of open "(" and closed parenthesis ")". Examples:

- (1 OR 2) AND 3 - Only show records that match filter 3 and at least one of 1 or 2
- (1 AND 2) OR NOT 3 - Only show records that either match both 1 and 2, or do not match 3.
- (1 AND 2) OR (3 AND 4) - Only show records that match both 1 and 2, or both 3 and 4.
- (1 AND (2 OR 3)) OR NOT 4 - Only show records that do not match 4 or those that match 1 and either 2 or 3.

## Inline Editing

Records in View can be edited inline just like fields on View Record pages. Click icon while mouse is hovering over particular column in view. Then enter new value and click "Save" button.



## Filtering by Formula

Though filtering Views based on expressions offers a lot of flexibility it does not cover all possible filtering scenarios. For this reason Rollbase offers filtering via a Formula which may include any stored fields (i.e. non-dynamically computed fields), SQL operands (including LIKE, IN, AND, OR, NOT), helpers (such as current User id, customer id, portal user id, etc) and the following special functions:

- `#YEAR(date)` returns date's year as integer
- `#MONTH(date)` returns date's month as integer in range 1 - 12
- `#DAY(date)` returns date's day of month as integer
- `#IF(expr, val1, val2)` returns val1 if expr is evaluated to true, val2 otherwise.

A View filter Formula may include tokens derived from the current date as shown above. For instance, to filter by a `d_o_b` date field where month equals the current month we would use a formula like:

```
#MONTH(d_o_b) = #MONTH(TODAY)
```



## Filtering by Search criteria

For OpenEdge Service objects, Rollbase offers filtering by search criteria to enable you to construct a complex filtering criteria in OpenEdge ABL syntax.

Note that filtering by Search criteria is available only to the OpenEdge Service objects that implement the `orderby` parameter in a JSON filter string. This must be done as part of developing your JSDO catalog. For more information on the recommend parameters to be implemented in your JSDO catalog file, see [Creating an Application from OpenEdge Data](#) on page 326.

The Search criteria supports the following search conditions:

- String
- Numeric
- Logical
- Date

The following table provides search examples:

**Table 1: Search criteria examples**

Condition	Valid Operators	Search criteria or Filter clause examples
String	ABL operators for ABL CHAR and LONGCHAR data types	<ul style="list-style-type: none"> <li>• Name='Girish'</li> <li>• Name&lt;&gt;'Girish'</li> <li>• Name BEGINS 'Gi'</li> <li>• Name MATCHES '*ri*'</li> <li>• NOT(Name MATCHES '*ri*')</li> <li>• (Name='John') AND (Last-name='Lennon')</li> <li>• (Name &lt;&gt; 'Girish' AND Name &lt;&gt; 'John' AND Name &lt;&gt; 'Lennon')</li> </ul> <p><b>Note:</b> Name of Field in your Search criteria is the field name in the JSDO catalog. That is, if the JSDO Catalog has Last-name but the Rollbase field name is Lastname, you must use the field name, Last-name in your search criteria.</p>
Numeric	ABL operators for ABL INTEGER, INT64, and DECIMAL data types	<ul style="list-style-type: none"> <li>• Age=20</li> <li>• Age&lt;&gt;20</li> <li>• Age&gt;=20</li> <li>• Age=?</li> <li>• Age&lt;&gt;?</li> <li>• Age=20</li> <li>• Age=20</li> </ul>
Logical	ABL operators for LOGICAL data types	<ul style="list-style-type: none"> <li>• SENDNEWSLETTER=Bool</li> <li>• SENDNEWSLETTER=NOT Bool</li> </ul>

Condition	Valid Operators	Search criteria or Filter clause examples
Date	ABL functions for DATE, DATETIME, and DATETIME-TZ data types	<ul style="list-style-type: none"> <li>• DOB=DATE (04, 13, 1987)</li> <li>• UPDATEDON&lt;=TODAY</li> <li>• ORDEREDON=DATE (OrderDate)</li> <li>• UPDATEDON=MONTH (TODAY)</li> <li>• DOB&lt;&gt;YEAR (DATE (04, 13, 1987))</li> <li>• ORDERDATE=WEEKDAY (TODAY)</li> </ul>

For information on ABL operations and functions, refer to the OpenEdge ABL Reference guide in the OpenEdge documentation set.

## Dynamic Filtering

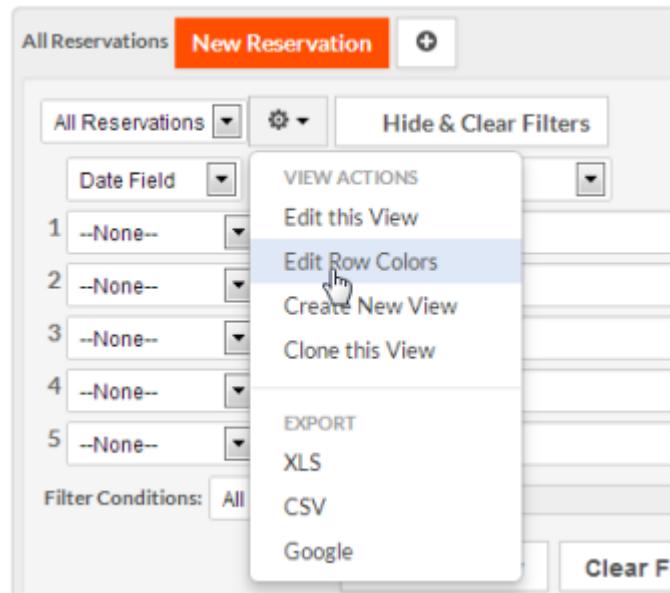
In the View header, clicking the **Filter** link opens up the View filter conditions options within the View header itself. This allows you to temporarily modify the filter conditions of your selected View without changing the saved version. Use the **Refresh View** button to reload the View once you have made changes and the **Clear Filters** button to go back to the default filter conditions of the saved View.

The screenshot shows a dynamic filtering interface. At the top, there is a dropdown for 'All Departments' and a 'Hide & Clear Filters' button. Below these are dropdowns for 'Date Field' and 'Dates Interval'. There are five filter conditions, each consisting of two dropdowns for '1' and '2'. The conditions are grouped by 'AND'. At the bottom, there is a 'Filter Conditions' dropdown set to 'All (AND)' and buttons for 'Refresh View' and 'Clear Filters'.

Click the **Hide & Clear Filters** link in the View header to turn off dynamic filtering and go back to the saved View.

## Filter Results Color Code

You can define colors to highlight records that meet certain criteria. On a page where the filter button is available, click **Filter** and select **Edit Row Colors** from the **View Actions** drop-down menu:



The **Color Codes** filter displays. You can define one or more filters and select the color to be used for each. If the record meets more than one criterion, the first applicable color will be used. The following example shows two filters, one where the **Start Time** was before the current day and the second where it is in the future:

Reservation: Color Code View: All Reservations

Color Codes					
1	Start Time	less than	TODAY	CFD5FF	
2	Start Time	greater than	TODAY	19FF75	
3	--None--	--None--		FFFFFF	
4	--None--	--None--		FFFFFF	
5	--None--	--None--		FFFFFF	

**Save** **Cancel**

The example filter renders the row colors as shown, with future reservations in green and past reservations in gray:

Action	Reservation	Room	Start Time	End Time	Last Change
<a href="#">Edit   Del</a>	Amit Sankar	Executive Conference	10/08/2014 02:00 PM	10/03/2014 03:00 PM	10/03/2014 11:36 AM
<a href="#">Edit   Del</a>	Stacy Walters	Fish Bowl Conference	10/06/2014 10:30 AM	10/06/2014 12:30 AM	10/03/2014 11:32 AM
<a href="#">Edit   Del</a>	Amit Waladi	Classroom 4	06/20/2014 08:00 AM	06/18/2014 05:30 PM	06/18/2014 01:22 PM
<a href="#">Edit   Del</a>	Sandy Beauchamp	Atrium Social Corner	06/25/2014 11:00 AM	06/18/2014 01:30 PM	06/18/2014 01:21 PM

## Programmatic Client-side Customization

Topics in this section explain how HTML event handlers and the Rollbase AJAX API allow you to create a customized user experience and provides a set of examples. Progress recommends that you use some sort of browser-side debugging as you develop. The JavaScript `alert()` method provides a simple way to debug your client-side JavaScript as you write it. Be sure to regularly monitor your browser's JavaScript errors. For Firefox, the FireBug plug-in can be useful for this purpose. In Internet Explorer (IE), we recommend enabling the **Display every JavaScript error** option. In Firefox, if you do not have FireBug, you can use the **Error Console** window.

Making too many AJAX calls from a UI page (say, making them periodically with small time interval) can seriously affect the system's performance. To prevent that, the total number of AJAX API calls is limited per user's login session. The limit is 1000 calls by default, which depending on your license, can be changed in Private Cloud installations.

## HTML Event Handlers

HTML events occur in a client's web browser when a user moves, clicks and drags the mouse, enters text and performs other interactions. HTML components, such as text boxes or selects (picklists), can expose JavaScript handlers that can be invoked when these events occur. You can read more about HTML events at this recommended web site:

[http://www.w3schools.com/jsref/jsref\\_events.asp](http://www.w3schools.com/jsref/jsref_events.asp)

Using event handlers, you can customize the behavior of your HTML components. In the example below, the pair of handlers named `onfocus` and `onblur` for Text Areas will expand a component to eight rows when it receives focus (`onfocus`) and then compress it to two rows when it loses focus (`onblur`):

```
onfocus = "this.rows=8"
onblur = "this.rows=2"
```

As shown, the `this` reference can be used inside event handlers to access or modify the properties of the relevant input field.

Tips for writing event handling scripts:

- You cannot use `form` in a Page's onload script. Instead, you can rely on the fact that the HTML form used to edit or create an Object record is always named `theForm`.
- A good practice is to verify that the JavaScript function you are calling exists on the current Page.
- Use prefixes to avoid naming conflicts. Note that Rollbase JavaScript system functions have the prefix `rbf_` and system variables have the prefix `rbv_`.
- Please keep in mind that disabled HTML controls do not send values upon HTML form submit.
- It is important to use the integration code of the selected picklist option rather than the field itself in formulas. That field is replaced (a system-generated id) will be different from installation to installation, so you cannot rely on it in formulas when publishing your Application.

## Defining Event Handlers

Progress Rollbase provides a way to specify JavaScript handlers for HTML events. To access this feature, navigate to an Object's definition page (**Application Setup > Objects**, and click the object, do not click **Edit**) Scroll down to the **Fields** section and click the **Events** link in the **Action** column for a particular field.

Alternatively, you can navigate to the **Field View** Page and select **Event Handlers** from the **More Actions** drop down box.

Different field types expose different event handlers. For instance, a text input box exposes the following: `onchange`, `onclick`, `onfocus`, `onblur`, `onkeypress`, `onselect`, `onmouseover`, `onmouseout`. A picklist exposes the following: `onchange`, `onclick`, `onfocus`, `onblur`, `onmouseover`, `onmouseout`. Read-only and hidden fields do not expose any handlers.

The following screen shows an example screen to define event handling code for an email address:

### Email Address: JavaScript Event Handlers

Save
Cancel

#### JavaScript Event Handlers

You can use integration names of other Client fields to reference them in event handling code for your **Email Address** field. Note that these other fields may or may not be present on a particular page.

Event Handlers Helper

onchange = "	<input type="text"/>
onclick = "	<input type="text"/>
onfocus = "	<input type="text"/>
onblur = "	<input type="text"/>
onkeypress = "	<input type="text"/>
onkeydown = "	<input type="text"/>
onkeyup = "	<input type="text"/>
onselect = "	<input type="text"/>
onmousedown = "	<input type="text"/>
onmouseup = "	<input type="text"/>
onmouseover = "	<input type="text"/>
onmouseout = "	<input type="text"/>

You can [debug your event handlers](#) by viewing the full HTML of the input field as it will be rendered.

Save
Cancel

The **Event Handlers Helper** allows you to select names of other Object fields and values (or codes) for picklists and lookups. You select from the drop-down menu and optionally, picklist. The values to use in JavaScript display in the center field. Note that, unlike similar helper components for templates, these helper tokens contain just the fields' integration names, rather than template tokens. These names can be used to refer to HTML elements in client-side JavaScript.

To view how your event handling code appears in generated HTML at runtime, click the "Debug" link to bring up the debug window. This window displays the actual field's HTML and renders the component below.

**Tip:** Turn on notifications about JavaScript errors in your browser. Please note that generated HTML event-handling code will be enclosed in double quotes, as shown in the onfocus method above. If you need to use a double-quote " inside your code, precede it with a backslash instead, such as \" , or consider using a single quote '.

**Tip:** Try to keep your event handler code relatively short. For longer handlers you can call JavaScript functions inside custom script components in your page(s).

**Tip:** you can use this in the body of event handlers to refer to the field and its properties. You can also pass this as parameter to any custom JavaScript functions that need to work with this field.

**Tip:** Event handling code is stored per field. Once specified, event handling code will be used for all Pages of type New, Edit and Status Change.

## Copying a Field's Value to Other Fields

In some cases you may need to use a value entered into one field as the default value for another field. For example, an email address is likely to be re-used as login name. To achieve this you can implement the onblur handler for the email field. Enter the following JavaScript code:

Email Address	<input type="text" value="admin@foo.com"/>
Login Name	<input type="text" value="admin@foo.com"/>

```
if (form.loginName && form.loginName.value=='')
form.loginName.value=this.value;
```

This code first verifies that the loginName field exists in the current form and has no value specified by the user. This results in the following:

You can use the prefix form. in front of any field integration name to reference that field in the DOM. For example, in the above code we reference the loginName field as form.loginName.

## Disabling Fields

Often, behavior of one field should depend on a selection made in another field. Consider the example below, in which there are three fields:

- Club Member (checkbox)
- Membership Fee (currency)
- Member Since (date)

The last two fields should only be available for the user to enter data if "Club Member" is checked, as shown below:

Club Membership

Club Member	<input checked="" type="checkbox"/>
Membership Fee	<input type="text"/>
Member Since	<input type="text"/>

Club Membership

Club Member	<input type="checkbox"/>
Membership Fee	<input type="text"/>
Member Since	<input type="text"/>

The steps below describe how to accomplish this.

1. Create a script component on the Page with the following code:

```
<script>
function my_showClubControls(form) {
    var isMember = form.club_member.checked;
    form.membership_fee.disabled = !isMember;
    form.member_since.disabled = !isMember;
}
</script>
```

2. Add event handling code to the onclick event of the "Club Member" checkbox (You can pass form as a parameter to JavaScript functions):

```
if (typeof showClubControls == 'function') my_showClubControls(form);
```

3. For consistency, add the same code as above to the Page's onload handler:

```
if (typeof showClubControls == 'function')
my_showClubControls(document.theForm);
After this, "Membership Fee" and "Member Since" fields will
be enabled or disabled, depending whether or not the checkbox "Club Member"
is checked or unchecked.
```

## Setting Default Values

In some cases selection or input of a certain value in your form should determine the default value for another field. Consider an extension of the example described in [Disabling Fields](#) on page 250, in which a selection in a picklist called "Membership Type" determines the default value for another field called "Membership Fee".

Follow these steps to create the logic that sets the value of a field based on the user's selection in another field:

1. Create a picklist called "Membership Type" with three values. Each of the values should have an integration code as shown below (important):

Field Type	Picklist
Field Label	Membership Type
below. Each value should appear on its own line.	
Values	Admiral A Captain C Sailor S

2. Add a script component to the Page:

```
<script>
function cust_membFee() {
    with (document.theForm) {
        if (membership_fee.value=='') {
            var code = membership_type.options[
                membership_type.selectedIndex].getAttribute('code');
            if (code=='A')
                membership_fee.value=1000;
            else if (code=='C')
                membership_fee.value=500;
            else
                membership_fee.value=200;
        }
    }
}
</script>
```

```
        }
    }
}
</script>
```

3. Add the following code to the "onchange" event handling code for the "Membership Type" picklist:

```
if (typeof cust_membFee == 'function') cust_membFee();
```

Now when the "Membership Type" option is selected and the "Membership Fee" box is empty, this box will be pre-populated with a new value depending on the selection.

## Using System Settings to Define Values that Might Change

The example shown in [Setting Default Values](#) on page 251 embeds values directly into JavaScript code. This makes it difficult to maintain these values that may be changed over time. It is often the case that a better solution can be achieved by using System-Wide Settings.

System-Wide Settings are available from the Administration Setup area. You can create fields for the "Settings" Objects as you do for any other object definition. The difference is that only one Settings record exists per customer. Values from that record can be used in various templates, formulas, etc.

To use system settings, let's make some changes to the shown in [Setting Default Values](#) on page 251:

1. Go to the "Settings" object and create three Currency fields for the Settings Object: "Admiral Fee," "Captain Fee," and "Sailor Fee."
2. On the Settings object's Edit screen, enter values for these fields. Your customers can easily change these values later.
3. Modify the above script to use values from the Settings record, rather than hard-coded values, as shown here:

```
<script>
function cust_membFee() {
    with (document.theForm) {
        if (membership_fee.value=='') {
            var code = membership_type.options[
                membership_type.selectedIndex].getAttribute('code');
            if (code=='A')
                membership_fee.value='{!!SETTINGS.admiral_fee}';
            else if (code=='C')
                membership_fee.value='{!!SETTINGS.captain_fee}';
            else
                membership_fee.value='{!!SETTINGS.sailor_fee}';
        }
    }
}
</script>
```

## Rollbase AJAX APIs

HTML event handlers are very useful to assist in the creation or editing of Rollbase records. However, to fully utilize the power of HTML event handling, you may need to retrieve data from other Rollbase records, or update other records, from within custom client-side code. This can be achieved by using the Rollbase AJAX API. Any JavaScript on a Rollbase Page can call these APIs as JavaScript functions.

The Rollbase AJAX API performs authentication using a server-side session and a client-side cookie. Access Control rules used in the Rollbase user interface and the Web API apply to the AJAX API as well: a currently logged-in user must have valid permissions to view/edit the requested object record.

All currently available API functions are listed below. We are continually extending the list of available APIs.

**Important:** Using AJAX API to make frequent calls to the production server may seriously slow the system, especially if many users are online simultaneously. To prevent it from happening Rollbase limits total number of AJAX calls per user login session to 1000. Private Cloud customers may alter that limit.

## Examples

The examples in this section demonstrate use of Rollbase API functions.

### Showing or Hiding a Page Section

You can show and hide sections on an Edit page based on a user's selection. This example was contributed by Tim Colson.

The example use case:

- Object has picklist with values "Not a member" (code "NO") and "Club member" (code "YES")
- Page's section titled "Membership Info" should be shown if "Club member" option is selected and hidden otherwise.

To accomplish this, follow these steps:

1. Add the following script to the Edit page:

```
function customizeEdit() {  
    var sectionID = rbf_getSectionIdByTitle("Membership Info");  
    var code = rbf_getPicklistCode("membership");  
    rbf_showOrHideSection(sectionID, code=="YES");  
}
```

2. Add `onchange` event handler to picklist: `customizeEdit();`
3. For consistency, add `onload` event handler to page: `customizeEdit();`

When the section is hidden, it looks like this:



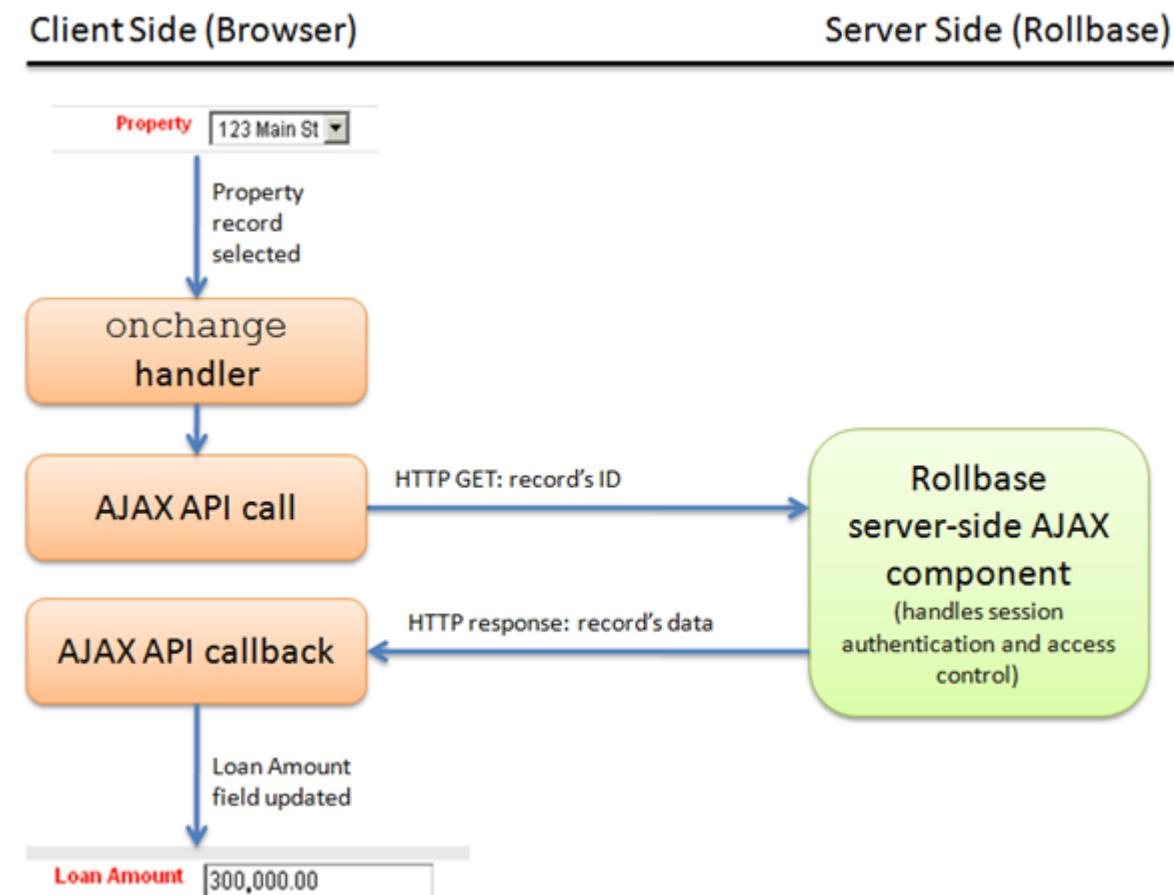
```
Membership -- Please select --
```

When the section is shown, it looks like this:

Membership	Club member
Membership Info	
Monthly Fee	<input type="text"/>
Start Date	<input type="text"/> 

## A Simple Lookup

Consider the following example: A Rollbase Page includes a lookup field for a property and an amount for a proposed mortgage. Upon selection of the property, we want to use the property's value as the default for the mortgage amount. This means a server-side trip using the AJAX API, as illustrated below.



To implement this example, follow these steps:

1. Create Object's representing Property and Mortgage, with a one-many relationship between Property and Mortgage (one property may have more than one mortgage)
2. For the Property lookup field in the Mortgage object definition, implement the following onchange event handler:

```
if (typeof cust.populateAmount=='function')
  cust.populateAmount();
```

3. Add the following script component to the "New Mortgage" Page using the Page Editor:

```
<script>
  function cust.populateAmount() {
    var propId = document.theForm.R4388623.value;
    rbf_getFields("property1", propId, "appraisal_amount",
    cust.callbackAmount);
  }

  function cust.callbackAmount(objName, id, dataValues) {
    if (document.theForm.amount.value == '')
      document.theForm.amount.value =
      dataValues.appraisal_amount;
  }
</script>
```

Now, when the Property lookup field is updated, it triggers an AJAX call to Rollbase and assuming that the requested record exists and the user has appropriate access rights, the value of the appraisal\_amount field will be set as the default to the amount field.

## A Financial Calculation

While formulas provide a powerful way to display calculations on View Pages, they are not so useful on Edit Pages, since formulas cannot dynamically utilize changes made by a user while inputting data into a form. However, formulas can be used on Edit and New Pages as placeholders for dynamically calculated values, even though the actual calculations now have to be performed on the client side using HTML events and the JavaScript API.

Consider the example of an Object that has three fields:

- Currency field with integration name: amount
- Percent field with integration name: rate
- Formula field (return type: currency) with integration name: interest, formula body:  ${!amount} * {!rate} / 100$

The Record View Page looks like this:

Amount	\$40,000.00
Rate	12.00 %
Interest	\$4,800.00

You can add these same fields to the Record Edit page:

Rate	<input type="text" value="12.00"/> %
Amount	<input type="text" value="\$40,000.00"/>
Interest	\$4800.00

However, when the user changes the content of the two editable fields (Rate and Amount), the content of the formula field does not change. In order to have the field change, you need to add a financial calculation similar to the following:

1. Add a Script component to Record Edit and New Page:

```
function cust_interest() {
    var m = rbf_getFloat(rbf_getFieldValue("money"));
    var r = rbf_getFloat(rbf_getFieldValue("rate"));
    var interest = m*r/100;
    rbf_setFieldContent("interest",
        rbf_formatCurrency(interest));
}
```

2. Add onchange handlers to the amount and rate fields:

```
if (typeof cust_interest == 'function') cust_interest();
```

3. Add the same code to the onload script on Record Edit Page for consistency.

Now the content of the Interest field will be dynamically updated on the Record Edit and New Pages in a consistent manner with the Record View Page.

## Access Control

Since the Client-Side API gives users (or portal users) access to underlying data, functions that access and modify the data are subjected to Access permissions are checked twice: First - for Current user (Portal User Role for Portals), Second - for System Role "Server API". If neither current user nor Server API role has sufficient permissions API call will fail. Note: Please refer to [Security and Access Control](#) on page 375 for more information.

Function	Access Required
rbf_selectQuery()	View all records of given object type
rbf_selectValue()	View all records of given object type
rbf_selectNumber()	View all records of given object type
rbf_getFields()	View current record
rbf_getRelatedIds()	View current record
rbf_getRelatedFields()	View current record
rbf_setBinaryFieldValue()	Edit current record
rbf_setTextFieldValue()	Edit current record
rbf_updateRecord()	Edit current record
rbf_runTrigger()	Edit current record
rbf_createRecord()	Create record of the specified object type
rbf_deleteRecord()	Delete current record

# Rollbase Portals

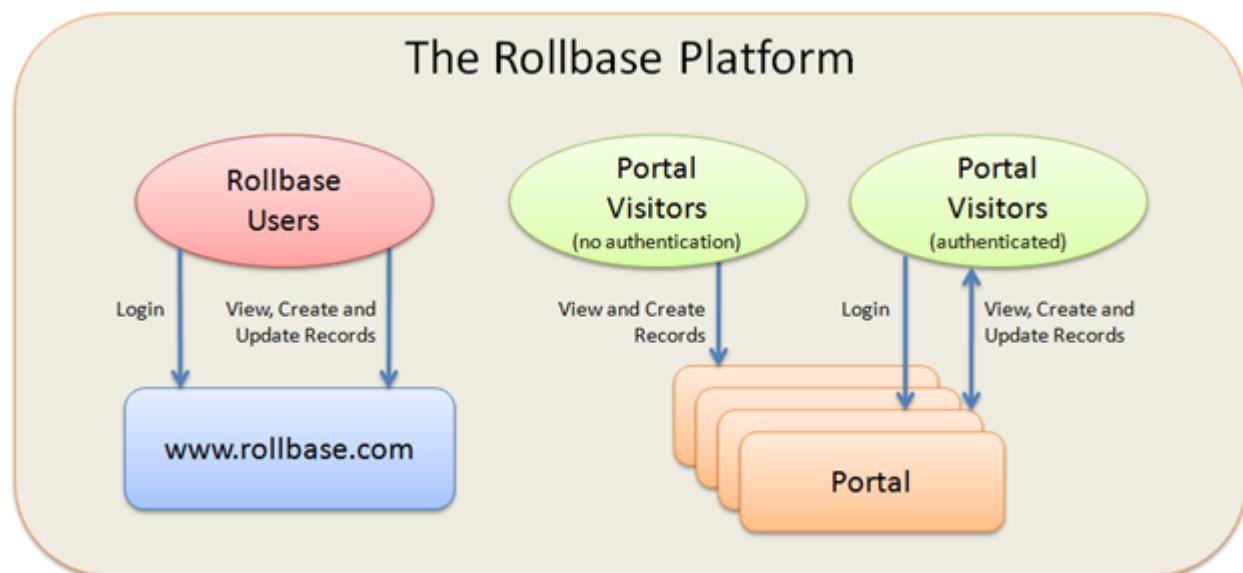
Portals are essentially external-facing web applications that are built using the same tools and components as standard Rollbase applications. Portals typically run as a part of a corporate website or intranet. They can be used to display dynamic lists of records and view record data, allow creation of records through custom forms and invoke triggers and custom business logic without requiring website visitors to have named Rollbase user accounts.

You can include any number of Portals with an Application and publish them as a part of an Application's XML. For information on publishing applications, see [Publishing and Distributing Applications](#) on page 407).

Portals are different from normal Rollbase applications in three significant ways:

- Portals can be accessed by anyone with internet access, though they can be password protected with named accounts
- Portals do not run within the normal Rollbase user interface. Rather, they consist of a set of pages that are linked together to form a cohesive website
- Portals can adopt the look and feel of any existing website where they can be embedded (in some cases portals can be used to power a domain's entire website; for an example of this see <http://www.geo30.com>)

The following diagram illustrates the differences between Rollbase application and portal usage:



## Planning and Building Portals

In addition to Portal-level settings, a Portal consists of a set of Portal Pages interconnected by links in any order you desire. This gives a developer greater flexibility than is available with normal Rollbase application Pages. However, it also carries a greater responsibility, so planning your Portal strategy in advance is important.

Warning: Building a sophisticated and easy to use Portal often requires significant knowledge of web design, HTML, CSS and JavaScript.

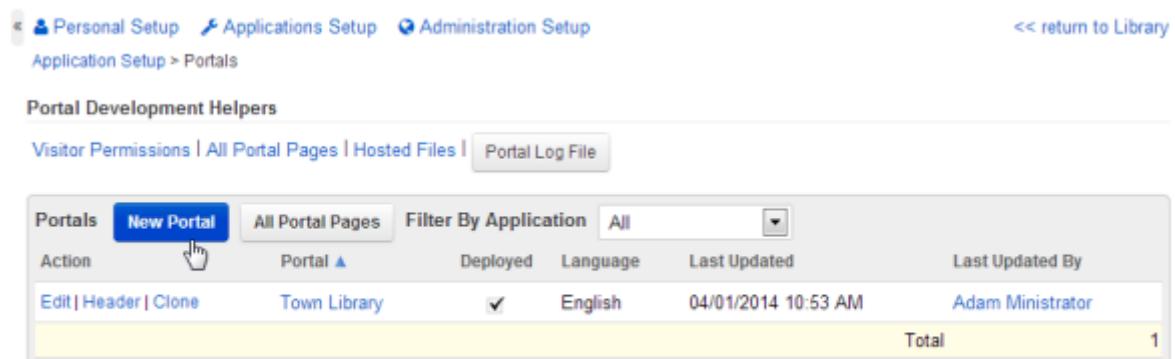
It is often a good idea to start with a diagram that shows how visitors can navigate through the Portal Pages you intend to create.

## Creating a Portal

Because each portal user cannot select their own language, date format and time zone the way regular users do, portal-level settings will be applied for all portal users.

Create a portal using the following steps:

1. Navigate to the portal setup page (**Application Setup > Portals**)



The screenshot shows the 'Application Setup > Portals' page. At the top, there are navigation links: 'Personal Setup', 'Applications Setup', 'Administration Setup', and a 'return to Library' link. Below the links, the page title is 'Portal Development Helpers'. There are four tabs: 'Visitor Permissions', 'All Portal Pages', 'Hosted Files', and 'Portal Log File', with 'Portal Log File' being the active tab. A table lists existing portals. The columns are: Action, Portals, Filter By Application (set to All), Action, Deployed, Language, Last Updated, and Last Updated By. One row is visible, showing 'Edit | Header | Clone' for 'Town Library', checked for Deployed, English for Language, 04/01/2014 10:53 AM for Last Updated, and Adam Ministrator for Last Updated By. A 'Total' row at the bottom shows the number 1.

Action	Portals	Filter By Application	Action	Deployed	Language	Last Updated	Last Updated By
Edit   Header   Clone	Town Library	All		✓	English	04/01/2014 10:53 AM	Adam Ministrator
Total 1							

2. Click **New Portal**.

The **New Portal** page displays:

The screenshot shows the 'Portal: New Portal' configuration page. At the top, there are navigation links: Personal Setup, Applications Setup, Administration Setup, and a 'Save' button. Below the navigation, the page title is 'Portal: New Portal'.

**Define Portal**

- Portal Name:** A text input field.
- Is Deployed:** A checked checkbox.
- Field-Level Help:** A checked checkbox with a note: "If checked, Field-Level Help will be available when hovering over the ? icon."
- "Powered By" Logo:** A checked checkbox.
- AJAX Calls:** A checked checkbox with a note: "for non-authenticated portal visitors."
- Language:** A dropdown menu set to "English".
- Date Format:** A dropdown menu set to "04/02/2014 04:59 PM".
- Time Zone:** A dropdown menu set to "(GMT-05:00) Eastern Standard Time (EST) America/New\_York".
- Description:** A text input field.

**Add to Applications**

Checkboxes for selecting applications:

- Select All
- Progress Rollbase
- MyTestApp
- Pacific Sports
- Room Reservation
- Library

**Save** and **Cancel** buttons are at the bottom.

3. Enter the following information:

- Portal Name:** Give the portal a memorable name.
- Is Deployed:** Exposes the portal to external access; keep this box unchecked until the portal is ready for use.
- Field-Level Help:** If checked, a "?" icon displays next to fields and contains any field-level help that you have defined.
- "Powered by" Logo:** Check this box to include a Powered by Rollbase logo at the bottom of portal pages.
- Ajax Calls:** If this box is checked, AJAX calls will be allowed to this portal without authentication. Depending on where and how the portal is deployed, this can be a security risk. Please use this option carefully.
- Language:** Select the language for the portal pages.
- Date format:** Select the date format to use in this portal in display and input fields.
- Time zone:** Select the time zone for this portal. All Date/Time field values will be adjusted to this time zone.
- Description:** Provide a description for this portal.
- Main Page:** Select the default Home page for this portal. Rollbase creates a Main Page automatically for new portals.

- **Login URL:** Use this setting only for portals embedded into other web sites that require authentication. This is the URL for logging in and the URL that will display when a user logs out.
  - **Add to Applications:** Select the applications to which this portal will be added.
4. Click **Save**.

If you are planning to use a portal as part of your website--rather than embedding it in one of your website's pages using an HTML iframe--you will need to configure the header and footer to adopt the look and feel of your site. See [Creating a Custom Header and Footer](#) on page 260 for more information.

## Creating a Custom Header and Footer

If you do not already have HTML code to use for your Header and Footer, the best way to get started is by picking a Page on your site to use as a template. View the HTML source of this Page and remove any central content where you would like your Portal content to appear. All HTML code above this content should be used as your Header and all HTML code below should be used as your Footer.

Tip: You can choose to include the default Stylesheet named Portaltheme.css above the Header in each Portal Page.

Follow these general steps to build the Header and Footer of your Portal:

1. If any code in your Portal Header or Footer HTML contains references to external files such as images, JavaScript files, CSS files, or links to other Pages using relative URLs (such as "../images/myimage.gif", "../index.html", etc.), you will need to do one of the following:
  - Switch to fully qualified URLs. In other words, always include the full path in your URLs such as "http://www.mycompany.com/index.html."
  - Include the following tag in the `<head>` element of your header: `<base href="www.mycompany.com">` ...where mycompany.com is your website's URL (i.e., the full path to the directory where your content can be found if the relative URLs were added to the end of it). This base tag tells any relative URLs to use the specified path as the root and is the only way relative URLs will resolve to the right domain.
2. You can use JavaScript code and CSS styles in the Header and Footer, but make sure any URLs to externally referenced files are fully qualified as specified above. In addition, each JavaScript file and CSS file included must itself use fully qualified URLs if you are not using a base tag in your header.
3. If you have enabled the HTTPS setting for this Portal, you either need to be using a base URL with HTTPS support, or you need to make sure that all of the images and files referenced in your Header and Footer have fully qualified HTTPS URLs.
4. You can change the main page by editing your portal's settings. When you are ready to go live with your portal make sure it is marked as deployed and simply provide access to this URL from appropriate places on your website or intranet.
5. You can use merge fields in the Portal's header and footer, such as `{!#CURR_USER.firstName}`. For information on merging fields, see [Adding Business Logic](#) on page 151.

When you finish creating settings for your Portal, you'll find a URL to the Main Portal Page on the Portal View Page which looks something like this:

<http://www.rollbase.com/router/servlet/Portal?c=391197&p=469206&h=false&g=469208>

## Creating Portal Pages

When you create a new Portal, Rollbase creates a new empty Page and assigns it as the Portal's Main Page. To add more Portal Pages, click the New Page link. For each new Page, complete the following info:

- Display name: Provide a unique name identifying this page
- "Only logged in portal users can access this page": Check this box if you are creating a page that only authenticated Portal Users should be able to access (we will discuss how to set up portal authentication below)
- Page Type: Select the type of page to create from the following options:
  - Generic Page: An empty page in which you can embed list views (list of records of a specific object type) and arbitrary web content (HTML, JavaScript, etc)
  - Search Results Page: A page used to display the results of a search within a portal
  - Object View Page: A page to view a single record of a specific object type
  - Object Create Page: A page to create a single record of a specific object type
  - Object Edit Page: A page to edit a single record of a specific object type
  - Object Selector Page: A page shown in a popup window that is used to select related records when using a lookup field in a Create or Edit page
  - Login Form: A page specifically designed to allow Portal Users (i.e. Records of an object with the 'Portal User' attribute) to login and authenticate to a portal
- Object Type (not shown for Generic Pages): Depending on the type of page you selected above, you will be presented with different object types to select here that determine what type of object record(s) this page will be dealing with:
  - If you selected a Login Form page, you will be presented with a list of objects that have the Portal User attribute; in this way you are creating a login page for a specific type of portal user.
  - If you selected any page type other than Generic Page and Login Form, you will be able to choose from all available objects

Portal Pages can be shared among different Portals in your Customer. When you create a new Page by clicking on the "New Page" link in a Portal's View page, this new Page is automatically assigned to the selected Portal. Later, you can share that Page with other Portals by using the "Assign Pages" link.



## Portal Page Properties

The "Actions" column in the list of portal pages offers the following choices:

- View: Preview the selected portal page in a pop-up window. For View and Edit pages you will be asked to select an existing record first
- Edit: Edit the selected portal page in the Page Editor
- Clone: Clone the selected page and edit the new page in the Page Editor
- Del: Delete the selected page (not available for the main portal page)
- Copy to: Clone the selected page and assign it to a different portal
- Properties: Set the selected Page's properties; available properties vary depending on the page's type (see below)

Page Properties allows you to set the following type-specific page attributes:

Property	Description	Page Type
Page Name	Human-readable page name	All
Only logged in portal users can access this page	Portal user must be authenticated, otherwise he/she will be redirected to this portal's Login page	All except Login
Destination Page	Page to redirect visitor to after form submission	New, Edit, Login
Automatically redirect to destination page when visitor already logged in	Self-explanatory	Login
Onload	JavaScript code to be invoked by the DOM "onload" event	All
Onunload	JavaScript code to be invoked by the DOM "onunload" event	All

## Portal Page URLs

Templates used on portal pages may frequently use URLs to other portal pages. To simplify portal development Rollbase provides UI helpers which for each page may generate:

- Page's URL
- Link to page
- Page's ID



For Generic or List pages these tokens can be used in UI templates directly. However for View or Edit pages you need to add an ID of the record being edited or viewed (generic token does not include this info for obvious reasons). For that append URL "id" parameter to UI token

```
{ !#PORTAL.159007.159009#url}&id={ !id}
```

At run time this will be resolved into full record's URL.

Tokens pointing to portal pages can be used in email templates as well.

## Using an EVAL Block on a Portal Page

The following example uses an #EVAL[ ] block on a Portal page to generate and output an HTML table with a list of records of Object "A" with integration name "a1":

Note: To use this example make sure that Portal User role has sufficient access to view records of Object A.

Use Page Editor and add Script Component with the following body:

```
#EVAL[
  function f1() {
    var arr = rbv_api.selectQuery("SELECT id,name FROM a1", 1000);
    var buff = '<table cellpadding=5><tr><th>ID</th><th>A</th></tr>\n';
    for (var k=0; k<arr.length; k++)
      buff += '<tr><td>' + arr[k][0] + '</td><td><a href="portal.jsp?c={!#CUSTIM.id}&p={!#PORTAL.16270.#id}&g={!#PORTAL.16270.16412#id}&id=' + arr[k][0] + '"';
    buff += '</a></td></tr>\n';
    buff += '</table>\n';
    return buff;
  }
  f1();
]
```

When page is rendering, #EVAL[ ] block is executed on server side. This block invokes server-side Query API to retrieve a list of records, then generates and outputs HTML table with retrieved data.

Please note that JavaScript body is wrapped in f1() function. Single call to that function performs actual calculations and makes output.

Second column in HTML table contains links to View page. Please note that template tokens are used to output IDs necessary to build that link.

Resulting output looks like this:

ID	A
16147	<a href="#">A500</a>
16150	<a href="#">A501</a>
16152	<a href="#">A502</a>

## Portal Security

There is something to keep in mind when you consider security aspects of Portals:

- Rollbase portals use HTTPS protocol.
- Access to portal pages can be limited to authenticated visitors only.
- Rules for password authentication can be set when you edit Password field on Portal User object:
  - Minimum length of password
  - Check if password must include both letters and digits
- Access rights for portal users are set for "Portal User" role. These permissions cannot be relationship-based or include LDF filters.

As an additional security measure you can specify whitelist of IP addresses to be checked when portal user logs in. For information on the additional security setup and administration, see [Advanced Setup and Administration](#) on page 421.

Important: Login session for Portal User will expire after certain period of inactivity. Private Cloud customers may configure this time interval (see [shared.properties](#) on page 505).

Tip: Administrators can login into Portal as selected visitor using Portal View page > More Actions dropdown.

Setting permissions for portal user objects requires careful planning to ensure that portal users cannot access information they are not supposed to. For the example from the previous section, you may have the following security requirements:

- Portal Users can create, view and edit their own personal visitor information (i.e. their own record).
- Portal Users cannot access personal records of other visitors.
- Portal Users can create and view (optionally edit and delete) their own comments.
- Portal Users cannot access comments created by other visitors.

To satisfy these requirements you can set the following permissions on the Visitor and Comment objects:

Object	Role	Access Granted
Visitor	Portal Users	Create new record
Vistor	Record Creator	View, Edit
Comment	Portal Users	Create new record
comment	Record Creator	View

In this design any Portal Users can create a new "Visitor" record - this represents self-registration. But the "View" privilege is granted only to the Record Creator. This means that any authenticated visitor can only view her own personal record. If she tries to access data of another visitor she'll be denied access.

Tip: You can assign permissions to the "Record Creator" pseudo-role from the Permissions section of an Object definition's details page.

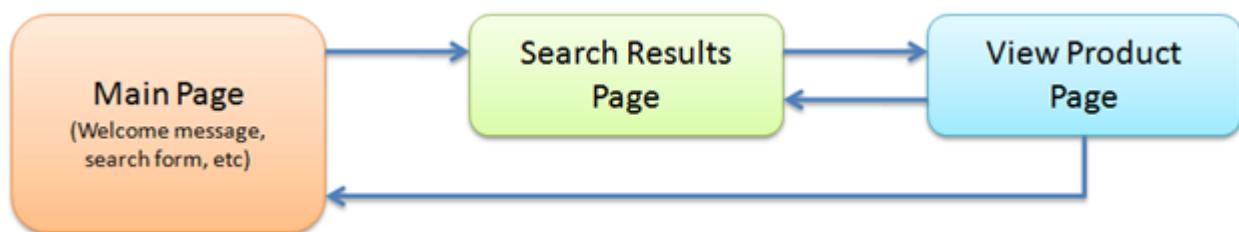
Tip: You can assign permissions through relationship between current user and records the same way as between regular User and records (see [User Roles and Permissions](#) on page 393).

## Creating Portals Without Authentication

Consider the following simple Portal example, where a portal user can:

- Search a list of products
- View a list of search results
- View information on a selected product

The following diagram shows the required Portal Pages and navigation among them:



To create this simple but fully functional Portal, take the following steps:

1. Create a new Portal as described in previous sections that will generate the Main Page.
2. Edit the Main Page and add a Text Search component to it (this will automatically be available in the "Available Components" section of the page editor), along with a template-based welcome message.
3. Create a Search Results Page for the Product object. The system will place a list of Product components onto this Page for you automatically.
4. Create a View Page for the Product Object. Rollbase will place existing Product fields onto this Page by default automatically.
5. Edit the Product View Page and remove or re-arrange Fields as needed. Create an HTML or Script component and, using the merge fields selector in the page editor, add links to the previously created Pages: Main Page and Search Results. Using these links, visitors can navigate through the Portal.
6. Finally, add permissions to view Product Object records to the Portal User Role.

Tip: You can create and edit Portal Links in the Page Editor to add explanatory text or images.

Warning: Unless the proper permissions are assigned to the Portal User role, visitors to your Portal will not be able to view, edit or create records. Make sure to update the Portal User role permissions when creating and deploying portals.

## Creating Portals With Authentication

Consider the following more complex example, where a Portal User can:

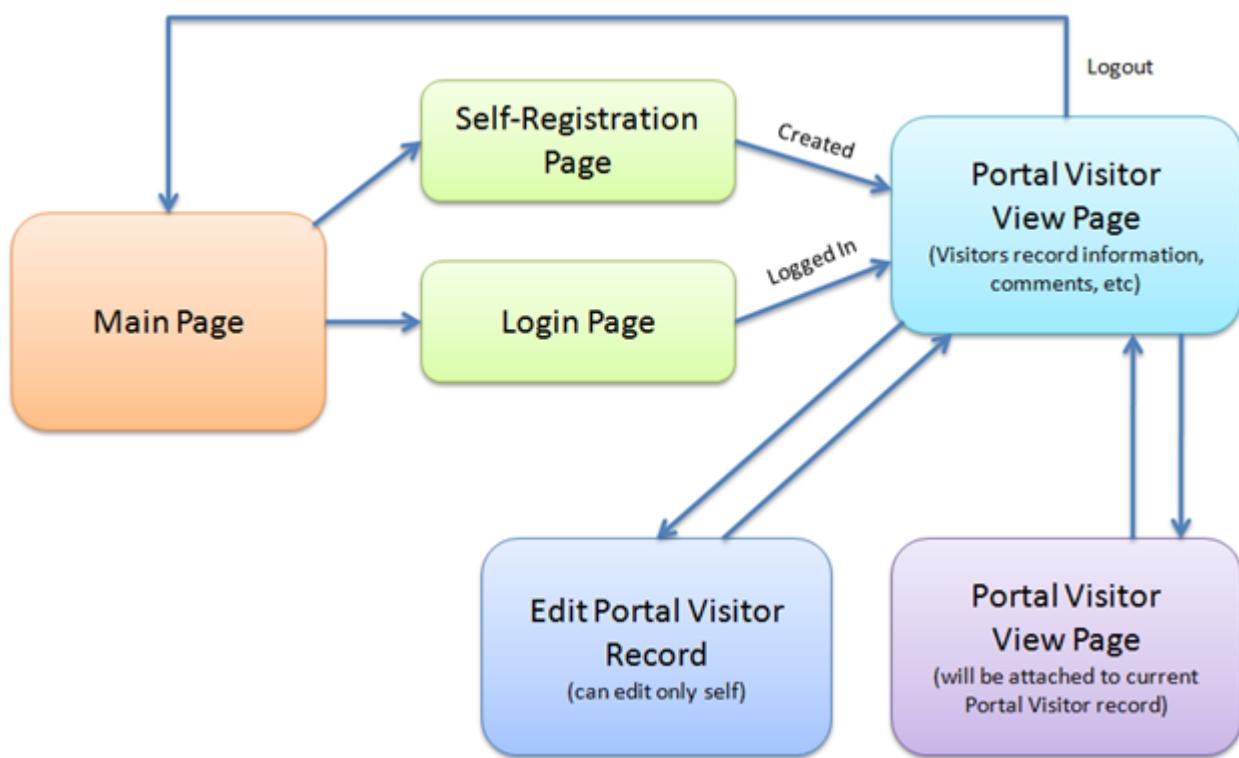
- Register for access to a Portal

- Login to that portal through login Page
- View his/her information
- Create comments
- View list of his/her own comments (but not comments created by other visitors)

To build such a portal you need to first create the following:

- An Object called "Visitor" with the Portal User and Contact attributes.
- An Object called "Comment" with a Text Area field
- A One—Many Relationship between Visitor and Comment.

The next step is to create a Portal and several portal pages. The following diagram illustrates the required Portal Pages and the typical navigation between them:



To create this Portal, create the following Portal Pages:

Page	Object	Page Type	Description	Authentication Required
Main	None	Generic	Has links to Login and Self-Registration Pages	No

Page	Object	Page Type	Description	Authentication Required
Self-Registration	Visitor	New record	Allows new visitor to enter some personal info, including login name and password	No
Login Page	Visitor	Login	Contains Login form	No
Visitor View	Visitor	View	Displays personal info and list of comments created by current visitor	Yes
Edit Visitor	Visitor	Edit	Allows existing visitor to change some personal info, including login name and password	Yes
Create Comment	Comment	New record	Allows creation of a new comment record. Current visitor will be automatically attached to the new comment record (and vice versa).	Yes

## Hosted Files

As with most websites, in portals you often need to make use of and reference files such as images, Flash movies, JavaScript, CSS, etc. These files are typically referenced in web pages through <IMG>, <SCRIPT> and other HTML tags. Rollbase provides a convenient way to host and reference arbitrary files through a mechanism called Hosted Files.

For example, using this feature, third party and custom JavaScript and CSS libraries can be uploaded and used by your portals to create a unique user experience and look and feel. JavaScript Hosted Files can be very useful in development of both client-side and server-side scripts. You can preview these files while working in Formula editor using drop-down list "View JS Hosted Files".

Note: Hosted files can be included as part of published applications just like other application components (e.g. Portals, Objects, etc.)

Note: Hosted files are most often used in Portal pages but can also be used in Rollbase Object pages.

You can prepare CSS Stylesheet with all tags used by Rollbase UI (see [Rollbase CSS Styles](#) on page 824) and upload this CSS as Hosted File. In this case you could use hosted CSS to customize appearance of all pages in your Customer. For information on hosting CSS, see [Advanced Setup and Administration](#) on page 421.

You can reference hosted files using merge fields in various scenarios such as in Template and Formula fields, HTML and Script components in pages, as well as within email and document templates. Using hosted file tokens in email templates provides a convenient way to add email attachments. See [Hosted File Tokens](#) on page 268 and [Using Hosted File Tokens](#) on page 269 for more information.

## Managing Hosted Files

Hosted Files can be associated with one or more applications. You can modify text-based hosted files, such as JavaScript, HTML, or XML, by simply modifying the text on within Rollbase without uploading a new file.

To upload and manage hosted files from setup:

1. Navigate to **Setup Home**.
2. In the **Applications Setup** section, click **Hosted Files**.

Action	File Name	Content Type	Last Updated	Last Updated By
Edit   Del	Books	image/jpeg	08/25/2014 10:44 AM	Adam Ministrator
Total 1				

3. To create a new hosted file, click **New Hosted File** and:
  - Enter a display name
  - Browse to upload the file (the size of a single uploaded file cannot exceed maximum allowed for Rollbase installation (5MB by default))
  - Optionally, enter a description
  - Select the applications to which this file should be attached

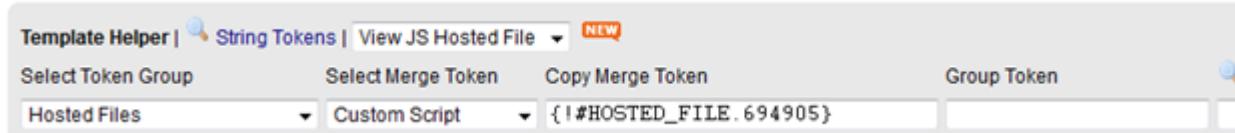
## Hosted File Tokens

The following table summarizes usage of merge field tokens corresponding to different types of hosted files: TOKEN NAME WHERE USED REPLACED WITH Filename Filenam [image] Filenam [text]

Token Name	Where Used	Replaced With
<i>Filename</i>	Anywhere, for example: { !#HOSTED_FILE.7034090 }	URL to the file hosted by Rollbase Note: When used in an email template, the hosted file is added to the email as an attachment instead of as a URL in the email body
<i>Filename [image]</i>	Any Template field or page-level HTML or Script component, for example: <img src='!#HOSTED_FILE.7034090' border='0' align='absmiddle' />	HTML <img> tag with URL to image file. This works very similarly to Shared Image fields.
<i>Filename [text]</i>	Any Template field or page-level HTML or Script component, for example: { !#HOSTED_FILE.13687907#text }	Entire text of the hosted file. This approach has many possible usages such as using custom JavaScript libraries in server-side formulas.

## Using Hosted File Tokens

In the Template Helper UI when editing any of these components, select the Hosted Files group and then select the merge field token corresponding to the desired hosted file and paste it into your template.



### Example Using Tokens in Formulas

The following example illustrates the usage of hosted JavaScript files in server-side formulas (using the *Filename*[text] format described above). Assume you have created and uploaded a JavaScript file that contains:

```
function my_func(x, y) {
  return x+y;
}
```

You can create a server-side formula which uses this file by using the hosted file's *Filename*[text] merge token and then invoking the function, as follows:

```
{ !#HOSTED_FILE.499203#text }

my_func({ !num1 }, { !num2 });
```

If you use the formula debugger to debug this formula, it will be parsed in the following form:

```
function my_func(x, y) {  
    return x+y;  
}  
  
my_func(100, 200);
```

**Important:** When using JavaScript functions in server-side formulas, do not use a stand-alone return statement; doing so will cause an error. The result of the very last JavaScript statement (typically a function call) will be used as the formula result. For information on server-side APIs, see [Server-side API](#) on page 559.

---

# Supporting Mobile Users

---

Rollbase offers the following mobile capabilities:

- **Rollbase Mobile** – Rollbase Mobile provides drag-and-drop tools and generated services that accelerate development of custom mobile apps. A Rollbase Mobile App can be created from scratch or be based on an existing Rollbase Web App. Rollbase Mobile Apps can be deployed as hybrid phone apps or as apps that mobile users access with a browser.
- **Mobile-Web Enabled** – A non-customizable Rollbase-generated mobile app that you can enable per Rollbase Web App. This functionality was formerly known as Mobile Edition. You choose the application objects and the associated views exposed to users through the mobile app, but you cannot change the look and feel of the mobile app itself. Users logging in to their usual application URL with a mobile device will be redirected to the mobile-web application.

You can also use REST calls to integrate Rollbase data into a mobile app that you create outside of Rollbase. See [Rollbase REST Methods](#) on page 732, [Toolkit for Mobile Applications](#), and [Metadata API and XML Reference](#) on page 687 for more information.

For details, see the following topics:

- [Rollbase Mobile](#)
- [Mobile-Web Enabled applications](#)

# Rollbase Mobile

Rollbase Mobile allows you to design and build mobile apps that can access a variety of data sources. For data stored in Rollbase, Rollbase Mobile automatically generates services to access the data. After you create the app in the Rollbase environment, you use the Mobile App Builder visual designer to drag and drop Mobile UI components, map them to data sources, and specify events and how they are handled—often without the need for additional coding.

The Mobile App Builder allows you to take advantage of the latest technologies, including: HTML5, CSS, JavaScript, jQuery, and Apache Cordova PhoneGap. After you design and test a Mobile App, you can deploy it as a hybrid app or as a browser app:

- Hybrid apps run inside a native container using the browser engine (but not the browser) to render the HTML and process the JavaScript locally. Hybrid apps created with Rollbase Mobile can be installed on:
  - Apple iOS devices, such as the iPhone and iPad
  - Android devices, including numerous smart phones and tablets made by a variety of manufacturers
- Browser apps are hosted by Rollbase and users access them from a Web browser on their mobile devices

The high level steps for creating a new Rollbase Mobile App include:

1. Create the Rollbase Mobile App Definition as described in [Creating a Rollbase Mobile App project](#) on page 302.

Rollbase Mobile creates a default User Interface (UI) that includes a functioning login page, one empty content page, and session services to handle user log ins.

2. Design the UI in the Mobile App Builder. Consult the [Mobile App Builder help](#) for information on using the Mobile App Builder.
3. Map the data sources generated for you as JSDO services to the UI Components.
4. Add events to invoke the services, as described in this [example](#)
5. Test, as described in [Testing Rollbase Mobile Apps](#) on page 309.
6. Deploy, as described in [Packaging and Deployment Options](#) on page 310.

During development of a Mobile App, you can switch back and forth between the Rollbase environment and the Mobile App Builder project. Similar to Rollbase Web Apps, Mobile Apps have a **Setup** page. The **Design** button launches the Mobile App Builder. When you are in the Mobile App Builder, click **Save & Close** to return to the Mobile App **Setup** page in the Rollbase environment.

If you learn best by doing, try the procedures described in [Simple Rollbase Mobile App Example](#) on page 281. To understand how Rollbase Mobile works, start with the [Overview](#) on page 273.

## Overview

Rollbase Mobile Apps can be created from scratch or based on an existing Rollbase Web App. If you create a Mobile App based on an existing Rollbase Web App, you select the core objects that you want to use in the Mobile App. At the time of creating a Rollbase Mobile App, you only specify the core objects you want to include in the Mobile App. You can later edit the Mobile App to include all the dependent objects. The Mobile App project includes generated JSDO services to access these objects, and any related objects (see [Mobile run-time architecture](#) on page 277 for more information on JSDOs).

The Mobile App is independent of the Rollbase Web App on which it is based, and you can, attach, import, and delete the Rollbase objects and views the Mobile App accesses with no effect on the Web App. If you create the Mobile App from scratch, you can add Rollbase objects and views for it to access. After initial Mobile App creation, if you make changes to any Rollbase components such as the views or objects used by the Mobile App, you need to synchronize the changes.

The following tools support Rollbase Mobile App development:

- **New Mobile App** — This screen opens when you select **Create Mobile App** in the **Create a New Application** wizard. Here you enter values for properties such as the App name, its type (such as a Phone App), and whether to create it from scratch or based on an existing Rollbase Web App. You can save the settings or click **Save & Design** to launch the Mobile App Builder.
- **Setup Mobile Application page** — Includes the objects the Mobile App can access and roles and permissions specific to the application. The **Setup** page has controls for designing, testing, and hosting a Mobile Web App. Adding objects to the list on the **Setup** page does not automatically add them to the Mobile App. Click **Edit** to specify the objects and views to include in the Mobile App, save and click **Sync Metadata** to add them to the Mobile App Project.
- **Mobile App Builder** — The multi-page, multi-tab Mobile App Builder opens when you click **Save & Design** for a new app, or when you click the **Design** from the **Setup** page for an existing app. The Mobile App Builder design canvas simulates the device you selected as the Mobile App type. For example, for a Phone App, you can drag and drop visual components onto the image of a phone page. This tool also provides built-in coding services (*Session Services* and *JSDO Services*) to more easily instantiate and access JSDOs in the open Mobile App with little or no additional JavaScript coding. After you have defined Rollbase objects and the views that you want the App to access, Rollbase creates an initial set of Session and JSDO Services based on the object views that you selected. You can then use these JSDO Services to instantiate and access JSDOs according to UI and data events that you select and add to the App. When you are ready to test, you can run the Mobile App as a Web App in an emulator that visualizes the App based on the selected Mobile App type. You can launch the test emulator either from the Mobile App Builder or from the Setup Mobile Application page. If you want to test the App as a Native App running on an Apple iOS or Android device, you can export the respective IPA and APK files to your local data storage from the Mobile App Builder. You can also export the HTML5 and JavaScript files if you want to test the Mobile App as a Web App outside of Rollbase. However, the Web App opens without the device emulation provided by the Rollbase test environment.

---

**Note:** If you are working with Rollbase Private Cloud, to develop Mobile Apps using the Mobile App Builder, you must first map your Private Cloud account to a Progress Pacific account (Progress ID) as described in [Enabling Mobile App Development in Rollbase Private Cloud](#) on page 300.

---

The following topics describe the basic development tasks that you can perform using these Mobile development tools:

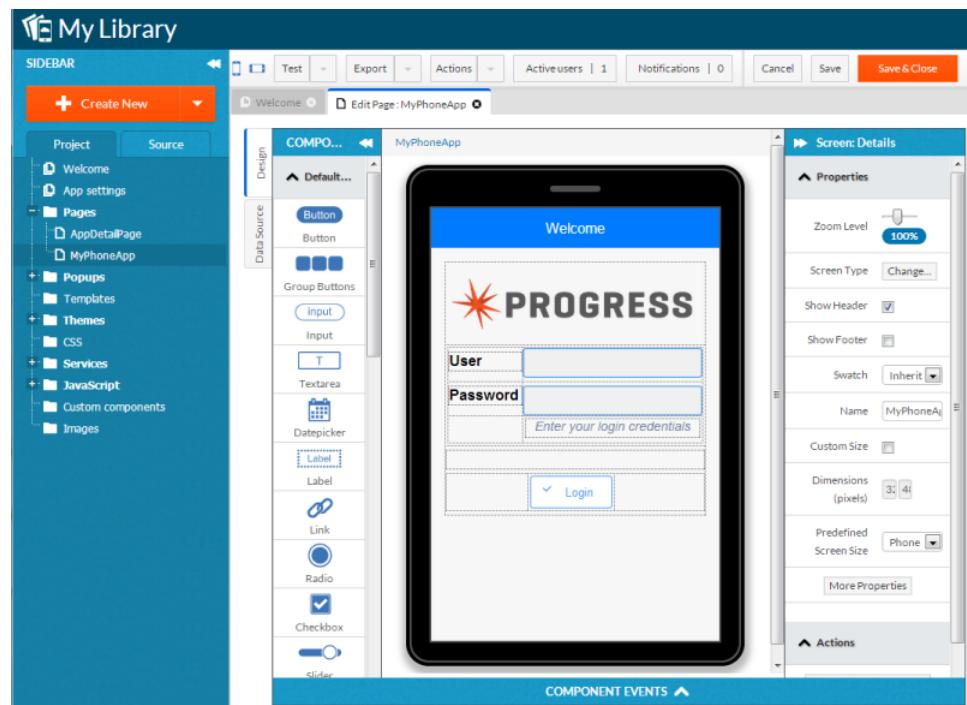
- [Creating a Rollbase Mobile App project](#) on page 302

- [Editing Rollbase Mobile App projects](#) on page 306
- [Adding objects to a Mobile App](#) on page 309
- [Adding views to a Mobile App](#)
- [Testing Rollbase Mobile Apps](#) on page 309

## The Mobile App Builder project

As described in [Overview](#) on page 273, when you create or update a Mobile App in Rollbase, you do the work in a new or existing project that is created in the Mobile App Builder. A *Mobile App Builder project* contains the work to build a single Mobile App, regardless of the ultimate deployment platform—whether it will be deployed as a browser app, a hybrid app, or both. You build the App itself using HTML5 and JavaScript components, which you can test from within the Mobile App Builder. You can also export the device-specific binary files from the project for testing and deployment.

The following figure shows the Mobile App Builder open to a newly created Mobile App named Library Members:



This example shows an App page selected for the new Mobile App, which is named **MyPhoneApp**. Rollbase automatically creates this page from the default Phone App template. Rollbase always create this as the default login page for every new App. Rollbase also creates an **AppDetailPage**, which is the default App content page that displays after a successful login. You can rename these default pages (and any other pages you create) as you want, and the Mobile App Builder updates existing project references to them automatically.

The pane on the far left contains two tree-view tabs, a **Project** tab that lists project elements that you can add to the App, such as new pages, components, and services of various types (including JSDO and Session Services), and the **Source** tab, which lists and allows you to view the contents of source files that are generated and maintained for each of the three supported deployment platforms you can use: iOS, Android, and Web. Above these two tabs is a **Create New** selection list that allows you to select and add one of the project elements that is listed in the **Project** tab.

When you create a new Mobile App from the Rollbase **Mobile Apps** section, it automatically creates the following project elements:

- A set of initial project settings (**App settings**, listed under the **Project > Project** node). This is a set of tabs that specify general settings (such as the Mobile App name and start page, which by default is the login page), external resource information (such as the loaded jQuery Mobile and PhoneGap library versions), various settings for Android and iOS Native App generation, and push notification settings.
- The default login and content pages for the App (**MyPhoneApp** and **AppDetailPage**, listed under the **Project > Pages** node) where you can begin adding UI components and data sources using the **Design** and **Data** tabs, respectively.
- A set of Session Services to manage a single user login session for the App (listed under the **Project > Services > SessionService** node). Each Session Service implements a Session operation, including creation of a Session instance (Instance Session Service, named **Session**), handling user login and Rollbase JSDO initialization (Login Session Service, named **Login**), handling user logout (Logout Session Service, named **Logout**), and settings for the Session and its operations (Settings Session Service, named **Settings**). The Instance Session Service also provides two events, Offline and Online, which you can use to check if your Mobile App is offline or back online. When you first create the Mobile App, Rollbase automatically:
  - Initializes the Settings Session Service with the basic information required to connect the Mobile App to the Rollbase cloud.

---

**Caution:** Do not change these settings, as they are required to allow your Mobile App to connect to the cloud.

---

- Adds the Instance Session Service and Login Session Service as data sources for the default login page (**MyPhoneApp**).
- Maps the user name and password entered by the user to corresponding request parameters of the Login Session Service.
- Adds UI events to the login page and its components to instantiate the Session when the page is loaded and to initialize and invoke the Login Session Service when the page is displayed, the user name and password values are set, and the **Login** button is clicked.
- Adds completion, success, and error events to the login data source to handle the invocation of the Login Session Service and its results, including navigation to the App content page (**AppDetailPage**) when login succeeds.
- Adds the Logout Session Service as a data source for the default App content page.
- Adds UI events to the App content page to display page components and to invoke the Logout Session Service when a **Logout** button is clicked.
- Adds success and error events to the logout data source to handle the invocation of the Logout Session Service and its results, including navigation to the login page when logout succeeds.

- Provides a means to display error messages on the default pages in response to error events, which you can specify as a value passed to a built-in `Appery()` JavaScript message function.
- A separate set of JSDO Services (listed under the **Project > Services** node) created for each Rollbase object associated with the view or views exported to the Mobile App. The exact Service instances created for each set of JSDO Services depend on the object views selected for export in the **New Mobile App** wizard and the relationships among the objects associated with the exported view or views. These Services are named for the object the JSDO accesses, using the object's integration name. For example, the JSDO Services for accessing the Title object are named for its integration name, `title_lb`. Each JSDO Service instance implements a JSDO operation, including each of the standard Mobile CRUD operations (Create, Read, Row, Update, and Delete JSDO Services), a JSDO settings operation (Settings JSDO Service), an operation to create the JSDO instance (Instance JSDO Service, named `JSDO`), and one or more Mobile invocation operations (Invoke JSDO Services, named `get*`, for example `getAll_members`) that read different record result sets from the associated object depending on its exported view, or views, and its relationships. You can use the **Data** tab to manually add any of these JSDO Services as data sources, map their data requests and responses to UI components on the page, and along with the **Design** tab, handle the appropriate UI and data source events to load a page and read, display, and modify Rollbase object data.

If and as you add any additional project elements, they appear under the node for their project element type.

Any App page that you select appears in the design area of the **Design** tab (in this case a phone page), where you can drag-and-drop visual components from the **COMPONENTS** pane on the left to the design area in the middle. To the right of the design area is a **PROPERTIES** pane where you can set component properties (such as the **Text** property value for a label), which apply to whatever visual component you have selected in the App page, including the page itself. As described previously, the **Data** tab allows you to add data sources to the selected App page and map their data to UI components on the page.

Above the displayed design area, the App Builder shows an active navigation path through the UI components you have added to the page, allowing you to more easily find and access the properties for a given component. Above the **Design** and **Data** tabs, the App Builder shows navigation tabs for any App pages and other existing project elements you have visited. You can delete one or more of these navigation tabs when they are no longer needed. Above the navigation tabs is a ribbon for performing various actions, such as changing the App page orientation (vertical or horizontal), exporting the project for Native App deployment, viewing and setting up push notifications, saving your current work, and closing the Mobile App Builder, among others.

To begin building the App, you typically create your UI, first on the default content page, then on any additional pages as you add them. Then, you:

- Add appropriate Session and JSDO Services to each page as data sources, and map the data for each Service by dragging and dropping between that data and the corresponding visual components and their properties (fields).
- Ensure that each Service is invoked appropriately in response to events on pages, components, and related Services.
- Optionally add custom coded JavaScript when responding to events on both the JSDO Services and the UI components on the page.

---

**Caution:** As you make changes, remember to click **Save** frequently to avoid losing changes you don't want to lose.

---

Finally, you can find additional documentation on the features of the Rollbase Mobile App Builder, including the Session and JSDO Services, by clicking the **Help Me** action in the lower left-hand corner of the App Builder page. This opens the [Rollbase Mobile App Builder Documentation Home](#) page in a separate tab of your browser.

For an example working with Session and JSDO Services in the Mobile App Builder, see [Simple Rollbase Mobile App Example](#) on page 281.

## Hosting and deployment options

You can package and deploy the same mobile app in multiple ways. These include:

- [As a browser app](#):
  - Progress hosts the app for Rollbase Hosted customers
  - Private Cloud customers download and host the app
- As a native mobile app, which you can distribute through any app store, such as Google Play and iTunes. Supported Operating systems include:
  - [iOS](#)
  - [Android](#)

## Mobile run-time architecture

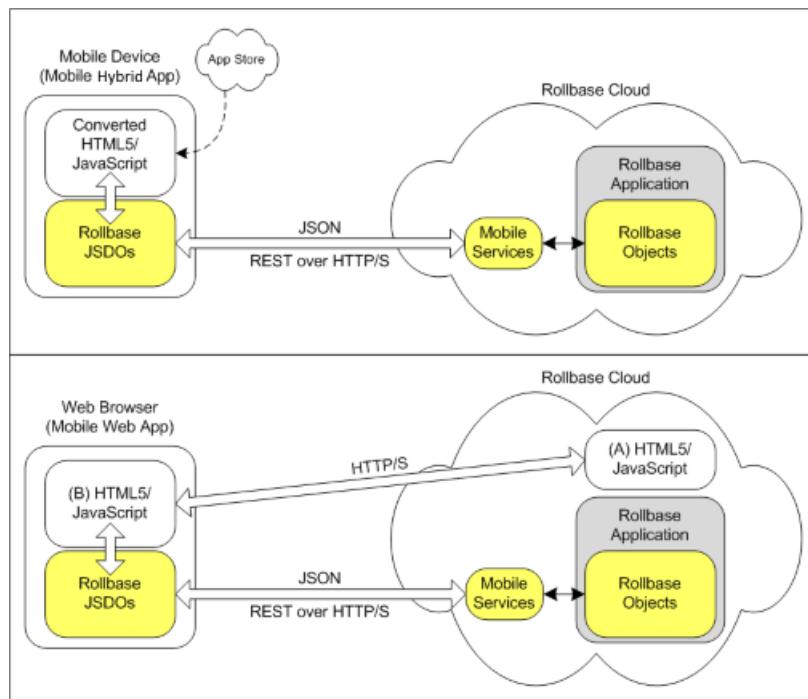
The figures in this section provide an overview of the Rollbase Mobile run-time architecture. These figures describe Mobile applications built with both Rollbase and Rollbase Mobile App Builder components from end to end, including a Rollbase REST transport between all Mobile Apps and Rollbase resources in the cloud.

Note that the Rollbase App Builder, like Rollbase Mobile-Web, generates all calls to the required REST resources to access Rollbase objects. However, Rollbase Mobile relies on a separate set of Mobile services to access these same objects independently of any containing Rollbase Web App (or other Mobile App). The Mobile App Builder also supports access to virtually any REST resource in addition to Rollbase objects, some of which have built-in support within the Mobile App Builder itself, such as OpenEdge Mobile and third-party REST resources. For more information on OpenEdge Mobile, see your [OpenEdge product documentation](#).

## Hybrid apps and browser apps

The following figure shows the basic end-to-end run-time architecture for the two mobile app types supported by Rollbase Mobile—Mobile hybrid apps running in a mobile device container and mobile apps running in the Web browser of any hardware platform.

Figure 1: Rollbase Mobile run-time architecture for hybrid and Web Apps



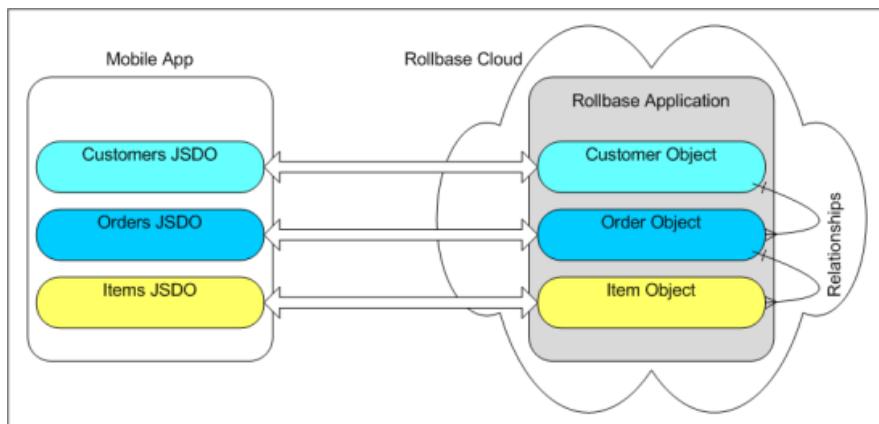
The difference is in how the HTML and JavaScript are deployed and run on the client. For a hybrid app, HTML and JavaScript files are converted to a native device format and deployed to an Apple iOS or Android App store to be downloaded and installed on the mobile device and run in the native device container. For a Mobile Web App, the HTML and JavaScript files are deployed in the Rollbase cloud (A) to load and run in a Web browser (B) like any HTML page. However, the UI is designed to display and operate in the smaller real estate of a Web browser running on a mobile device.

Both app types access Rollbase data through Rollbase JavaScript data objects (JSDOs). A Rollbase *JavaScript data object (JSDO)* is a JavaScript object instance that provides Rollbase Mobile access to the data in one Rollbase object. It does this through record access operations (*Mobile operations*) managed by a dedicated Mobile service in the cloud. This Mobile service provides basic create, read, update, and delete (CRUD) operations, as well as one or more invocation operations that return record results sets (lists), depending on the object view and its relationships. There is a one-to-one relationship between a JSDO, a Mobile service, and its Rollbase object.

Note also that the data and operations specified by these JSDOs are serialized and transported between the Mobile App and the Rollbase cloud as JSON (JavaScript Object Notation) media using REST over HTTP/S.

## Mobile access to Rollbase objects using JSDOs

The following figure provides an overview of how a Rollbase JSDO accesses a Rollbase object, which works the same regardless of the type of Mobile App.

**Figure 2: Accessing Rollbase Objects**

In response to user input, the Mobile App can call standard JavaScript methods on the JSDO to perform Mobile CRUD operations on individual records of its associated Rollbase object. In addition, the Mobile App can call customized Mobile invocation methods on the JSDO to return a list of one or more records from the associated object. The lists of records that these invocation methods return depend on the views defined and implemented for the object and its defined relationships with other Rollbase objects to which it has access. In this way, and according to the particular Mobile App UI that you design, a given Mobile App can read and modify the data in any and all of these related objects.

For example, in the previous figure, assume that objects are accessed according to two selected views of the Customer object. The Rollbase Customer object has a one-to-many relationship with the Order object, which in turn has a one-to-many relationship with the Item object. The Customers JSDO can then read a list of all Customer records or two different subsets, depending on the two Customer views it implements. The Orders JSDO can read a list of all Order records or the Order records related to a specified Customer record. The Items JSDO can similarly read a list of all Item records or the Item records related to a specified Order record. Each of the three JSDOs can perform single-record create, read, update, and delete operations on its respective Customer, Order, or Item object.

A typical Mobile App accessing these objects might, for example, return a list of customers showing the total open balance on shipped orders for each, while another Mobile App might show an aged order delivery status for each customer displaying the number of orders that have been shipped and delivered.

Note that as a developer, to work with these JSDOs in Rollbase, you need to do little or no JavaScript programming. When you create a new Mobile App in Rollbase, Rollbase generates a JSON file (JSDO catalog) for each Rollbase object represented by the object's view or views that you select for export to the Mobile App. Each generated *Rollbase JSDO catalog* contains information about the schema of its associated Rollbase object and the operations provided by the corresponding JSDO to access the object's data. This JSDO catalog effectively tells the JSDO how to communicate with its Mobile service to access the corresponding object hosted by Rollbase or in a Private Cloud.

**Note:** Rollbase JSDO Catalogs are related to, but different from, the OpenEdge JSDO catalogs used to create Rollbase objects from OpenEdge Mobile services. The difference is that Rollbase JSDO catalogs are generated by Rollbase to allow Rollbase Mobile Apps to access existing Rollbase objects, while OpenEdge JSDO catalogs are generated by OpenEdge to provide both Mobile and Rollbase application access to OpenEdge data objects. For more information, see [Creating Rollbase Objects from OpenEdge Services](#) on page 322.

## Mobile login sessions and user access

Each Rollbase Mobile App supports a default Mobile login session to provide access to its selected Rollbase objects. This login session supports Rollbase Form-based Authentication for REST.

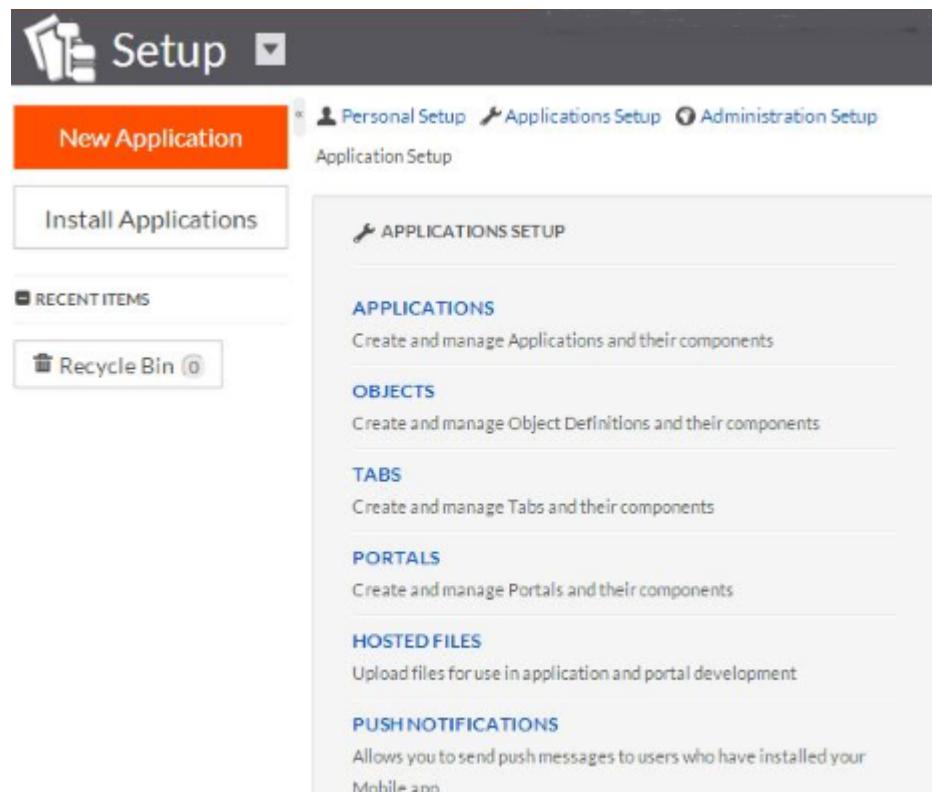
When you create a new Rollbase Mobile App, Rollbase creates a default login page for the App in the Mobile App Builder, including all the necessary setup to implement the login session. On successful login, this login page navigates to a default content page for the App, which in turn navigates back to the login page on successful logout. You can test this login session in the Mobile App Builder immediately after creating the App using the Progress ID you use to log into the Progress Pacific console.

An additional feature of Mobile login sessions is their ability to respond to alternating offline and online conditions that often affect mobile devices and their apps. This feature allows a Rollbase Mobile App to know when it has gone offline, or returned back online, so that it might continue to run and either access or avoid certain application features, depend on their requirements for network and Rollbase server access.

## Working with Push Notifications and other Mobile features

Rollbase Mobile enables you to push messages out to users who have installed your Mobile App, as well as to manage other features available for use in Mobile Apps.

These features are all available using the Mobile Applications Monitoring Dashboard. You can access this Dashboard from **Setup Home > Applications Setup > Push Notifications**:



The **Mobile Applications Monitoring Dashboard** appears and provides a single place to manage these features using the following tabs:

- **Push Notifications** — Lists push-enabled Mobile Apps, which you can open to create, schedule, and send push notifications to some or all mobile devices where the open Mobile App is installed. You can also easily switch between Mobile Apps and view statistics for all push notifications that you have already sent. Note that you can push-enable only Mobile Apps exported and deployed as hybrid iOS and Android Apps.
- **Databases** — Provides access to a database to store push notifications that you create and schedule for each push-enabled Mobile App.

---

**Note:** The Mobile App Builder documentation describes this feature as a general cloud database that you can use to store and access any data you want from within a Rollbase Mobile App. However in this release of Rollbase Mobile, the cloud database feature is **only** supported to set up push notifications.

---

- **Server Code** — Allows you to write custom script using JavaScript that runs on the Rollbase server. Any custom script can be invoked in Mobile Apps using a REST service. Working with server code, you can create and edit scripts, test them, watch tracing and statistics information, create libraries, and define dependencies.
- **Templates** — The Mobile App Builder allows you to create custom templates from existing Mobile Apps that you can use when creating new Mobile Apps. Using this tab you can find a list of your custom templates, view usage statistics for these templates, and open them for editing.
- **Integrations** — Allows you to create Webhooks. A Webhook is a method of augmenting the behavior of a service with custom callbacks. Using this tab, you can create Webhooks and specify a URL where POST requests are sent. You can then attach the Webhooks you have created to specified Mobile Apps. When a selected event occurs, an HTTP POST request is invoked to the Webhook URL.

---

**Note:** On Rollbase Private Cloud, to use the Push Notification API with a Mobile App, use <https://api.appdesigner.rollbase.com> as the base address instead of <http://api.mobile.rollbase.com> as is documented at:

[http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push\\_notifications\\_API](http://docs.mobile.rollbase.com/documentation/backendservices/push/#Push_notifications_API).

---

For more information on using these features of the Mobile Applications Monitoring Dashboard, see the Mobile App Builder documentation, which you can access from the [Rollbase Mobile App Builder Documentation Home](#) page.

## Simple Rollbase Mobile App Example

The topics in this section step you through the creation of a Rollbase Mobile App that will access data from an existing Rollbase Web App. This example uses a simple Library Web app that tracks titles checked out by library members. The example Mobile App will allow library members to use their phones to see a list of all available titles (those not currently checked out).

Before attempting to follow the steps, you must first install the Library Web app:

1. Download the Library Web app from the following location:  
[http://documentation.progress.com/output/ua/resources/Library\\_App.zip](http://documentation.progress.com/output/ua/resources/Library_App.zip).
2. Extract the Library XML file and import it into your Rollbase account. For information about importing application XML, see [Installing and Updating Applications from XML](#) on page 114.

Follow the procedures detailed in the following topics:

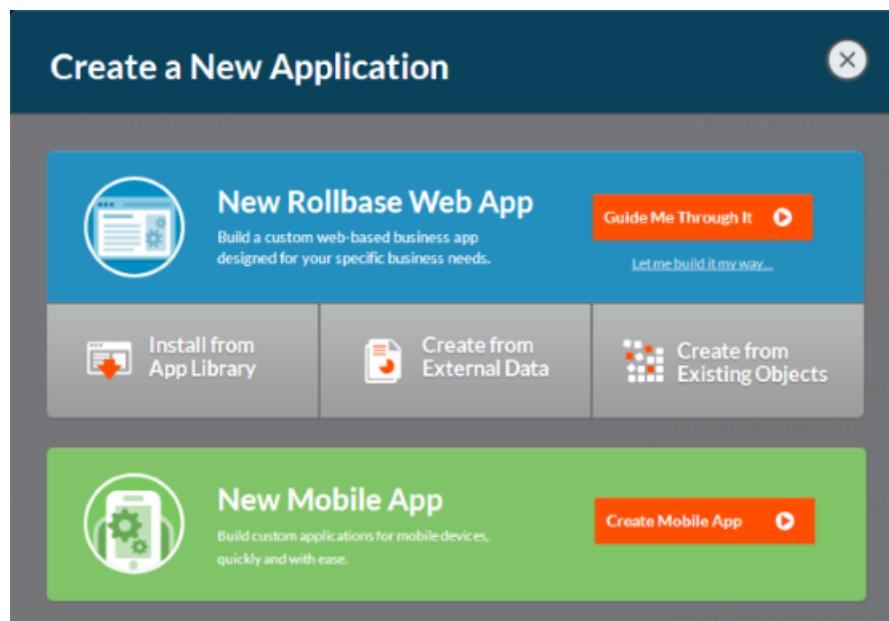
1. [Creating the Mobile App Definition](#) on page 282
2. [Editing the default pages](#) on page 286
3. [Adding components to display records](#) on page 289
4. [Adding data sources to retrieve records](#) on page 293
5. [Adding Events to invoke the Services](#) on page 297
6. [Testing the Example Mobile App](#) on page 299

## Creating the Mobile App Definition

After you have installed the Library Web app as described in the [introduction](#), follow these steps to create a Mobile App definition and launch the Mobile App Builder:

1. In the left sidebar, click **New Application**.

The **Create a New Application** dialog appears:



2. In the **New Mobile App** box, click **Create Mobile App**.

The **New Mobile App** screen displays:

3. For the fields on the **New Mobile App** screen do the following:

- **Name:** Enter **My Library**.
- **Description:** Enter **Displays a list of available titles**.
- **Template:** Leave the default choice, **Phone App**.
- **Create Option:** Choose **Based on Existing Web App** and select **Library** from the list of apps.
- In the **Core Objects** area, all the core objects of the Web App are automatically selected to be included in the Mobile App. For this sample, proceed with the defaults.

**Note:** Only the objects under **Selected Objects** are included in your Mobile App.

Your screen should look similar to the following:

New Mobile App
**Save**
**Save & Design**
**Cancel**

Mobile App Name

Description

Template  Phone App  
 Tablet App  
 Other

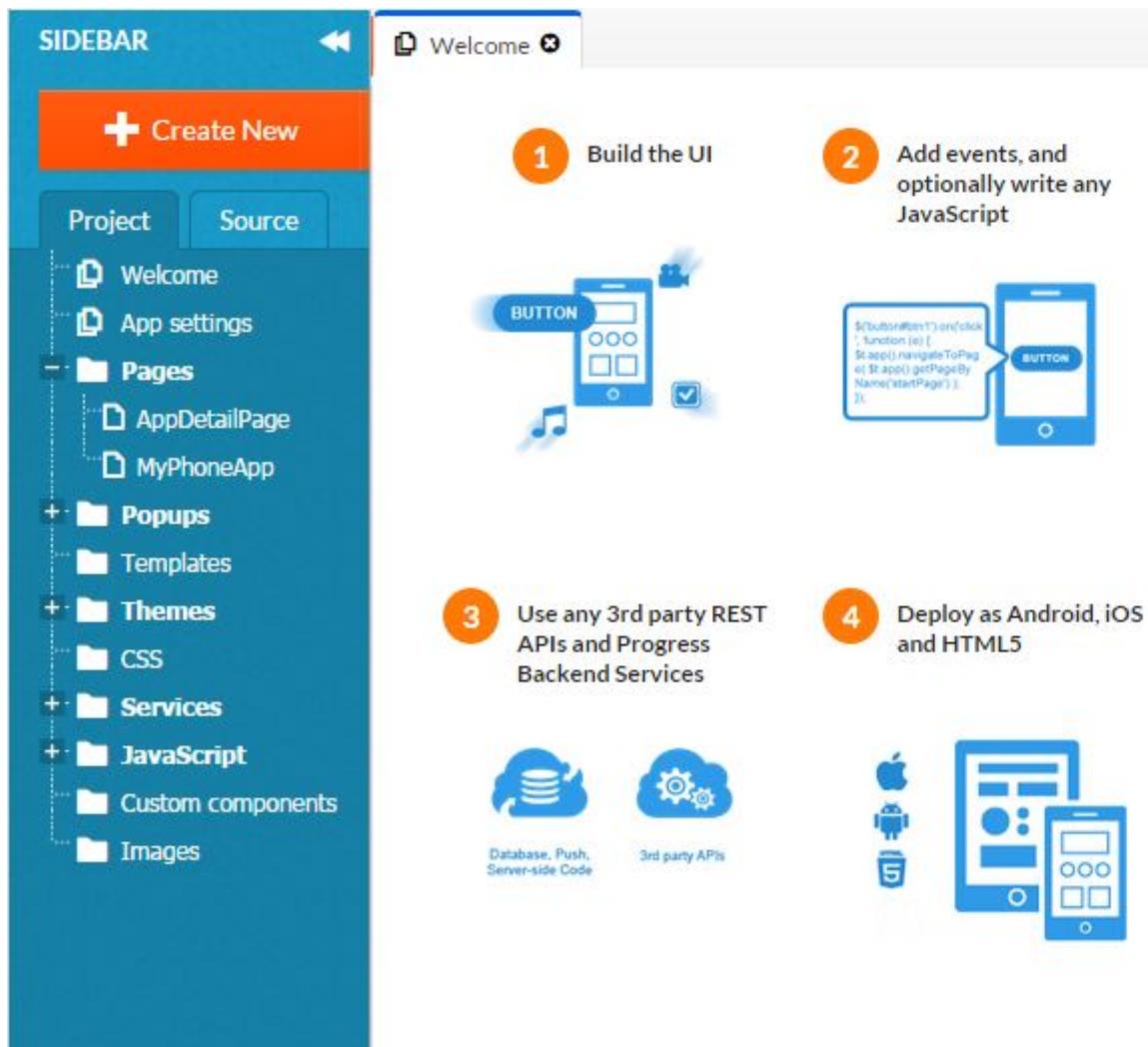
Create Option  From Scratch  
 Based on Existing Web App

**Core Objects**

Available Objects	Selected Objects
	Check Out Comment Member Title User

4. Click **Save & Design**.

After creating a project and generating services based on the selected view, the Mobile App Builder launches and displays the new **My Library** Mobile App project:



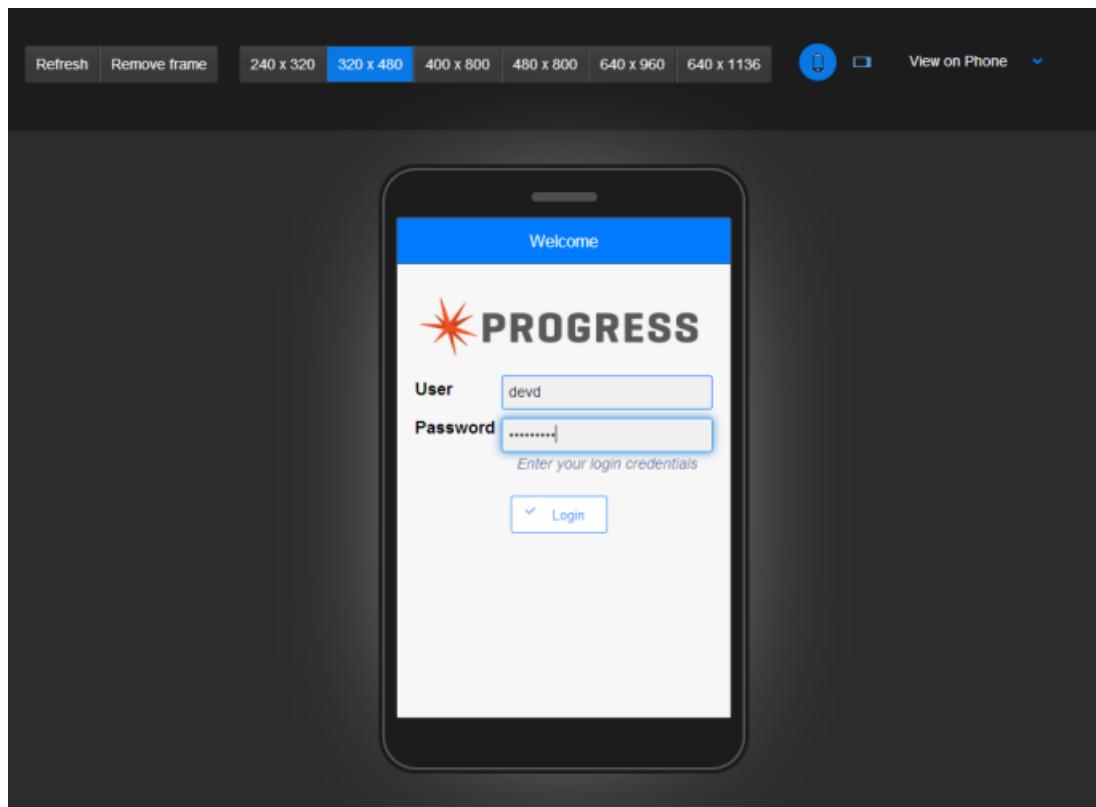
Rollbase Mobile generated two interface pages for you, a login page and a default app page. You can see how this default UI will appear to end-users by following the steps described in [Viewing the Default UI](#) on page 284. Or, you can proceed to the next task: [Editing the default pages](#) on page 286.

## Viewing the Default UI

Follow these steps to see the default pages that Rollbase Mobile generated for you:

1. In the control bar above the **Welcome** tab, click **Test**.

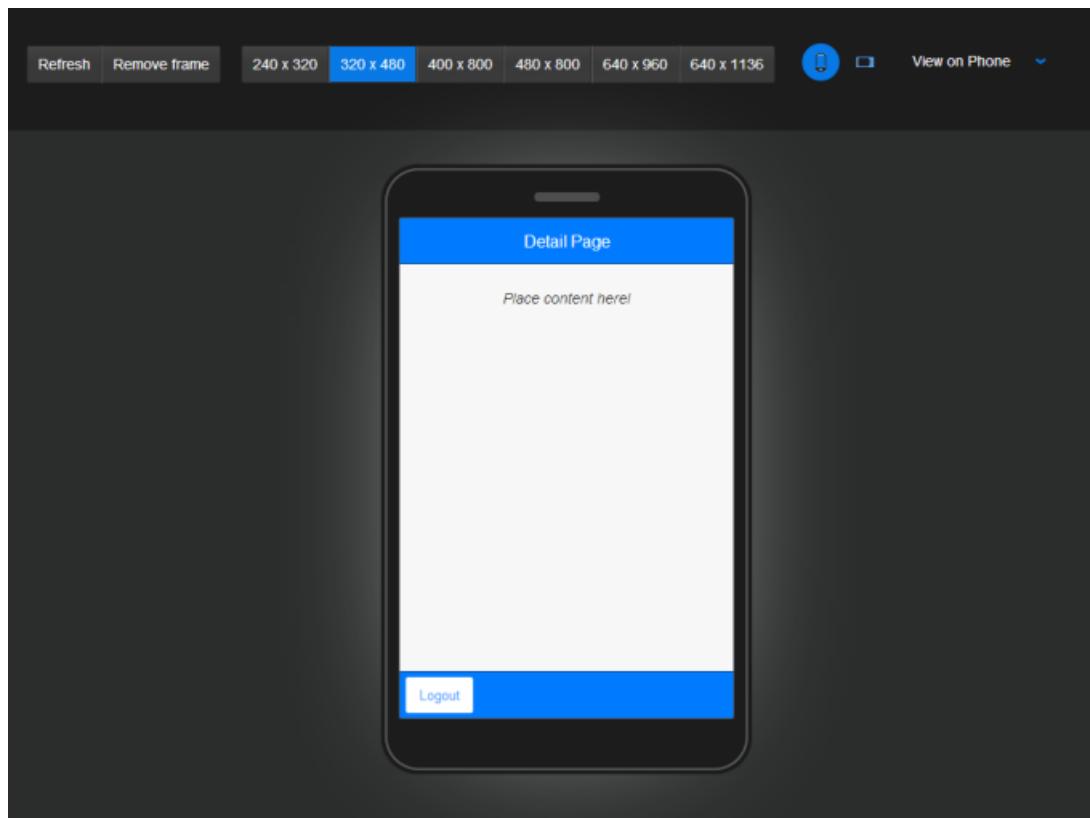
A new tab opens in your browser with a **Mobile Preview**. The default login page displays. You can customize the look and feel of this page and provide your own branding.



2. Enter your Rollbase user name and password to test the login page. Click **Login**.

If the log in is unsuccessful, by default, returned HTTP status codes and messages appear on the login page above the **Login** button.

On successful log in, the preview displays the default content page:



This page is where you will add the UI components and Services to display the list of available library titles.

The controls across the top of the Mobile App Builder screen provide the following options:

- Removing the device frame so the UI fills the browser page
  - Changing the frame size
  - Changing the frame orientation (vertical or horizontal)
  - Viewing the current page on a device by scanning a displayed QR code
3. You can click **Logout** to navigate back to the login page.
  4. Close the **Mobile Review** tab.

Next, follow the instructions for [Editing the default pages](#) on page 286.

## Editing the default pages

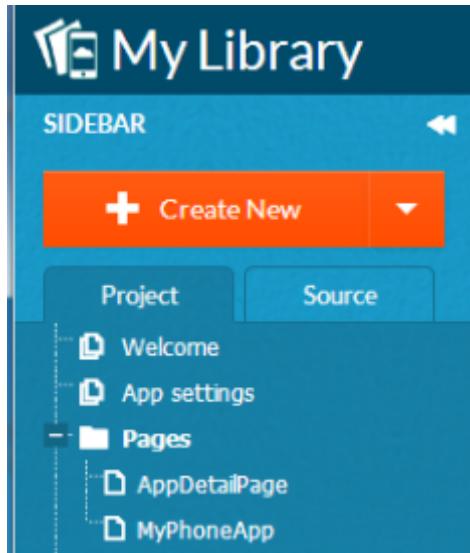
In the Mobile App Builder, the sidebar **Project** tab **Pages** node lists app screens. Select a page to open it in an **Edit Page** tab. Select screen components to display their settings in the **Properties** pane.

By editing component properties, you could, for example, replace the Progress logo with another image or change the error messages to be displayed in the **Label** component above the **Login** button. If you select that component, you will see that it is mapped to `userErrorMsg`. JavaScript associated with the existing data source events contains references to empty user messages that you can modify.

To access and edit My Library pages for this example, follow these steps:

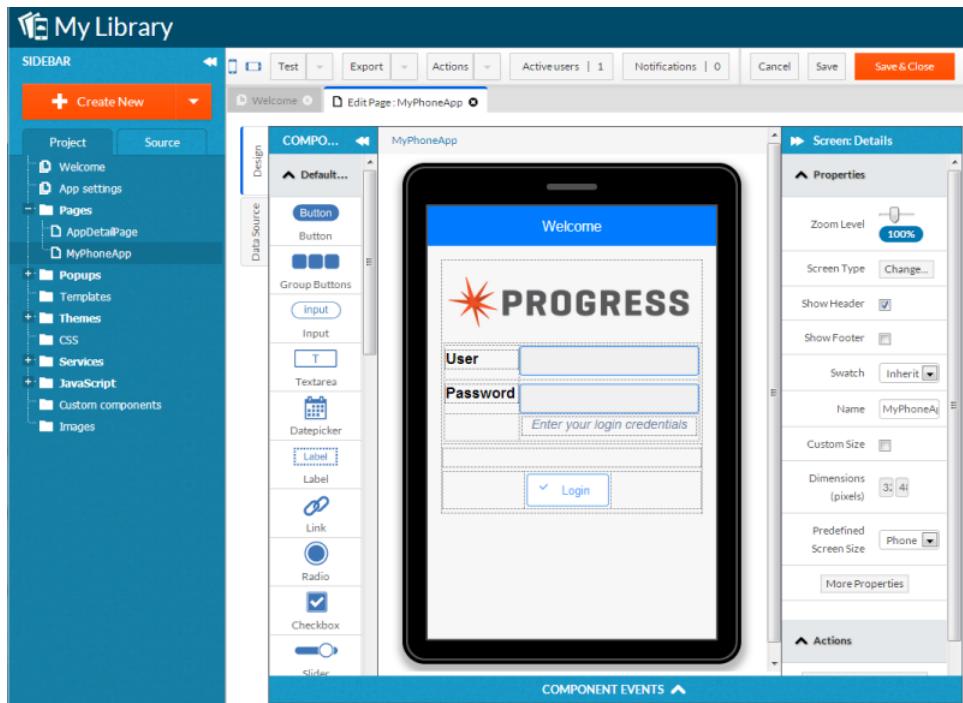
1. In the Mobile App Builder sidebar, expand the **Pages** node on the **Project** tab.

This node contains the default content page, **AppDetailPage**, and the login page, **MyPhoneApp**:



2. Change the page label on the login page:

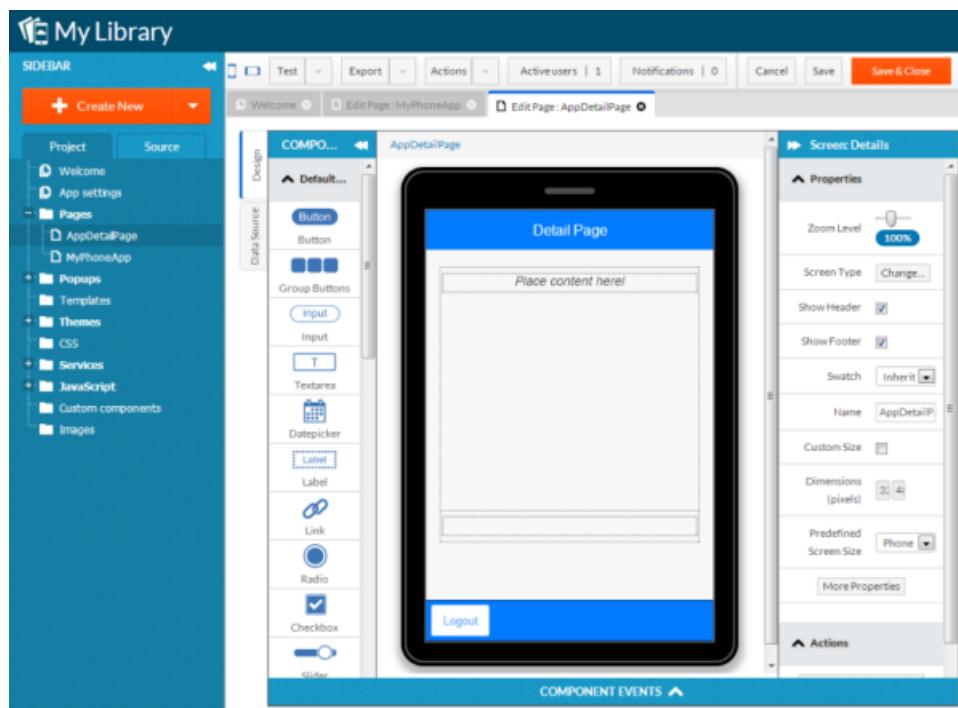
- a) Select **MyPhoneApp** to open the default login page in the **Edit Page** tab:



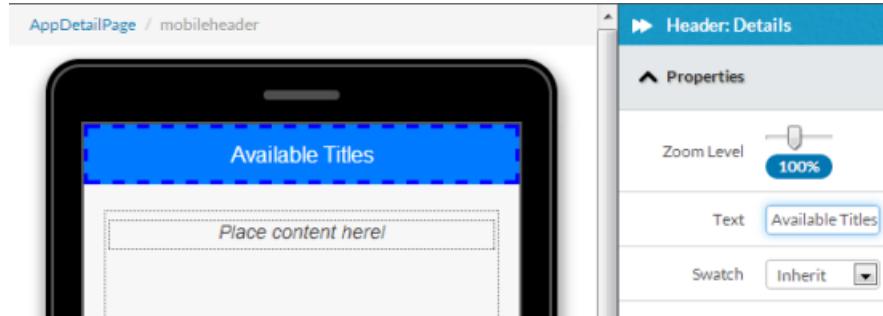
- b) Select the **Welcome** banner at the top of the simulated screen.
- c) In the **Properties** pane, in the **Text** field, change the banner title from **Welcome** to **My Library**:



3. Similarly, change the title on the empty **Detail Page**:
  - a) On the **Project** tab, click **AppDetailPage** to open the default content page in an **Edit Page** tab:



- b) Select the **Detail Page** banner in the simulated screen.
- c) In the **Properties** pane, change the value of the **Text** property to **Available Titles**, as shown:



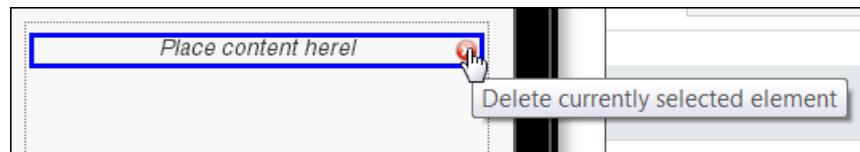
Next, you will [add the components necessary to display the list of available titles](#).

## Adding components to display records

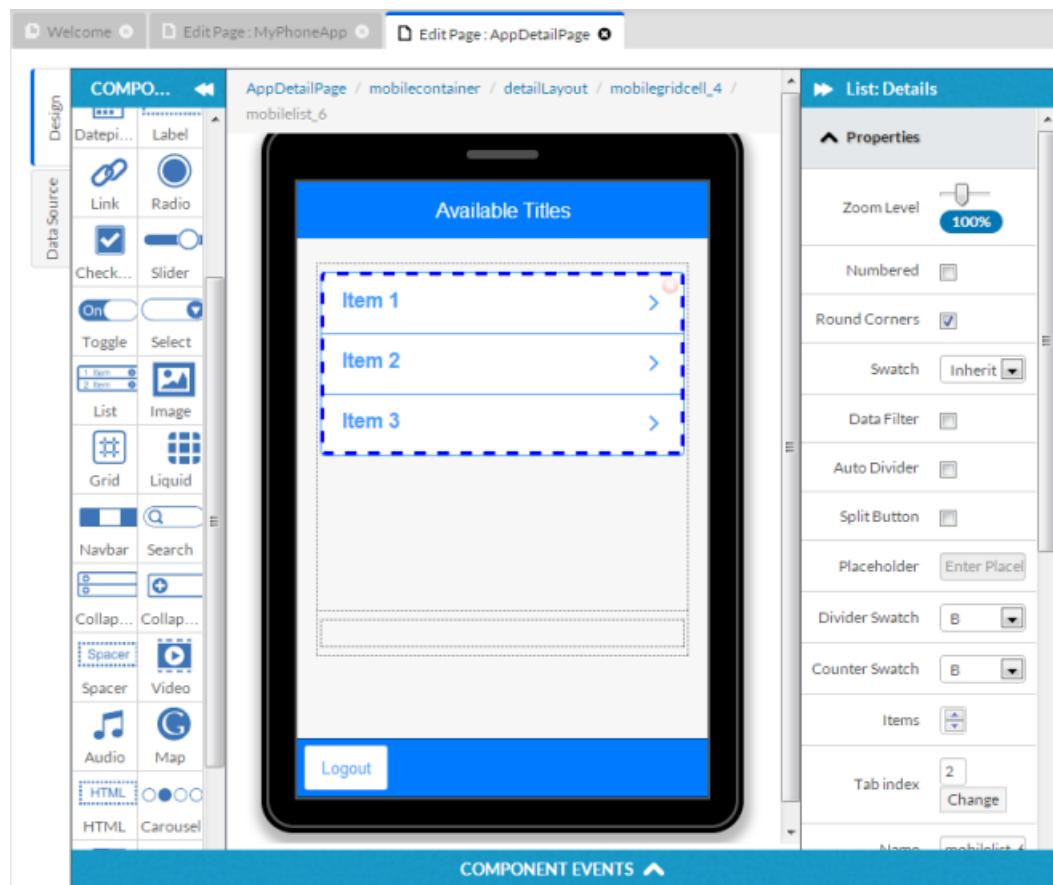
The example app displays a list of available titles to end-users. The Mobile App Builder offers several types of components that could display a result set. You will use a simple list component item and map it to the view of available titles.

Add the list component to the default page as follows:

1. If it is not already open, select the **AppDetailPage** page from the sidebar **Pages** node to open it in an **Edit Page** tab.
2. Remove the placeholder label by clicking the delete button:

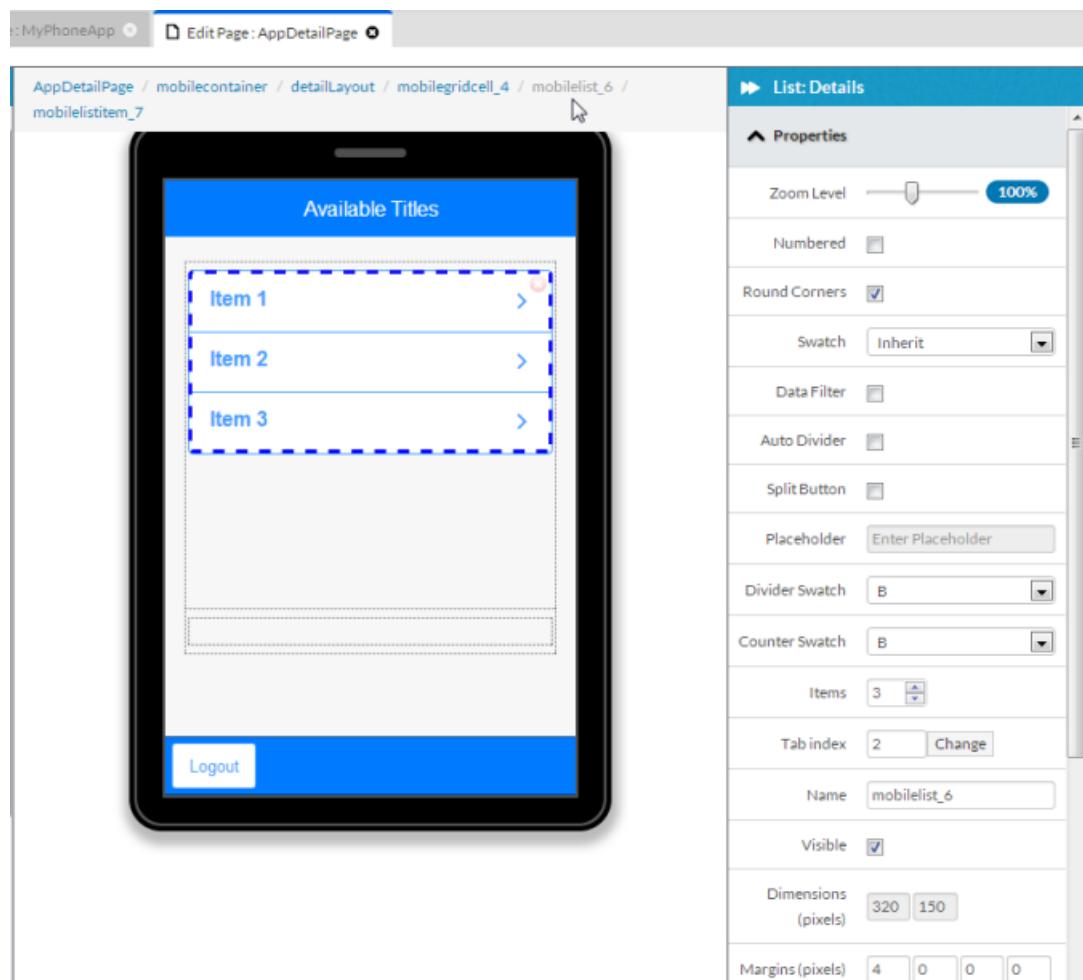


3. Drag a **List** component from the **Components** pane on the left to the simulated phone screen:

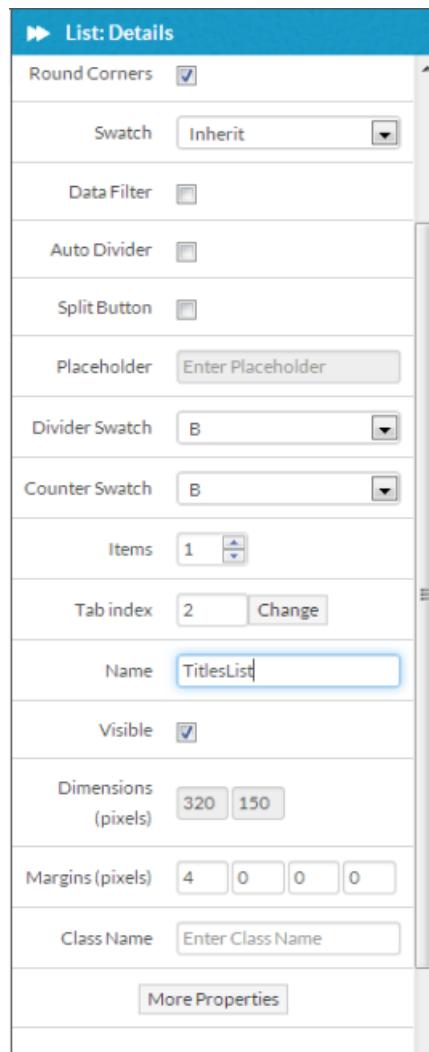


The list component is composed of item components contained within a grid. Each item could be mapped to different types of records. Since this example will only display titles, you can remove all but one item.

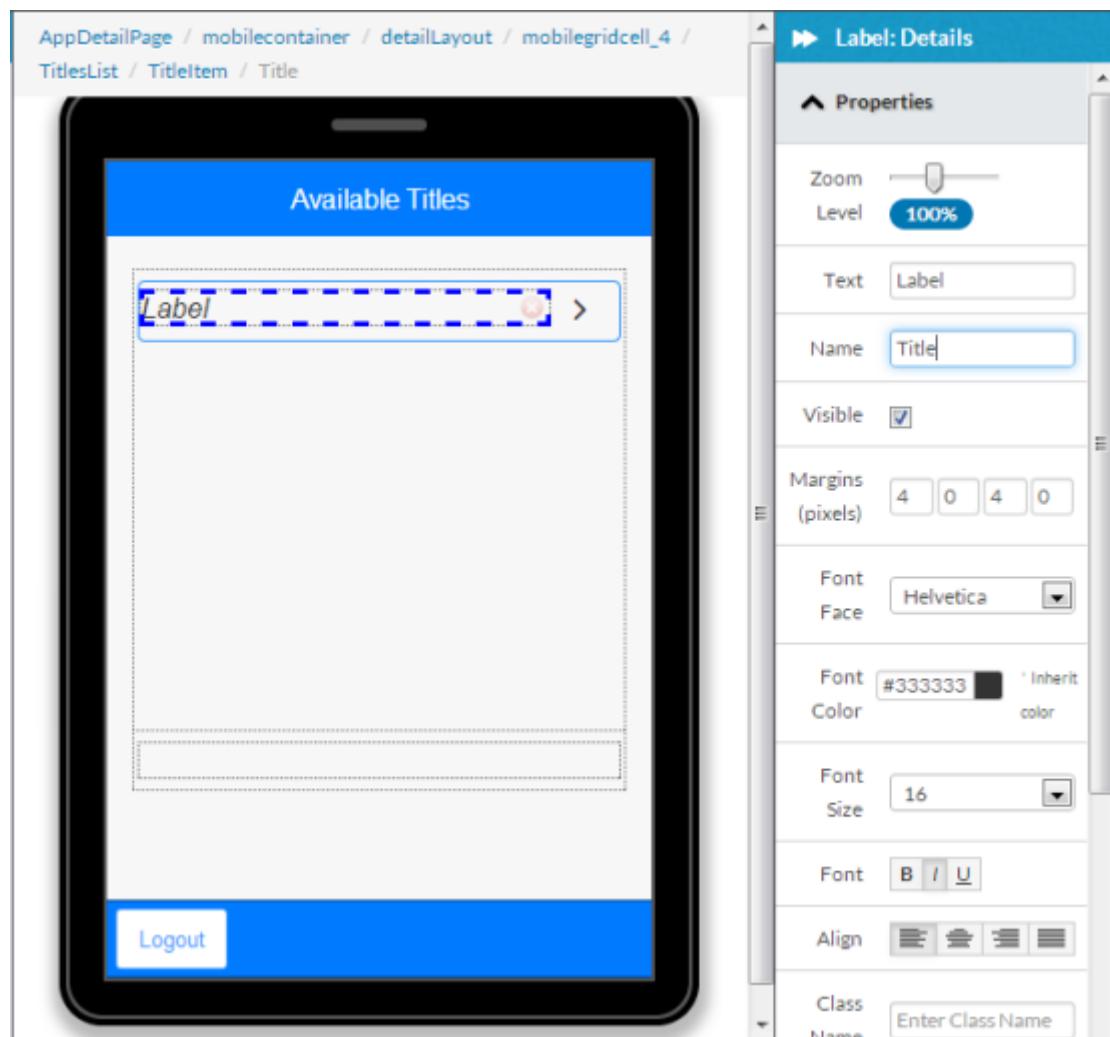
4. To remove the extra items and name the item component so it will be easier to identify, follow these steps:
- The breadcrumbs above the phone image show the hierarchy of components: page, container, layout, grid, list, item. Click the list link to view its details in the **Properties** pane:



- b) In the **List Details** pane, click the **Items** counter to reduce the number of items to 1 and click **Apply**.
- c) Change the value in the **Name** field to **TitlesList**:



5. Drag and drop a **Label** component on top of the **TitlesList** item box and change its **Name** property value to **Title**, as shown:

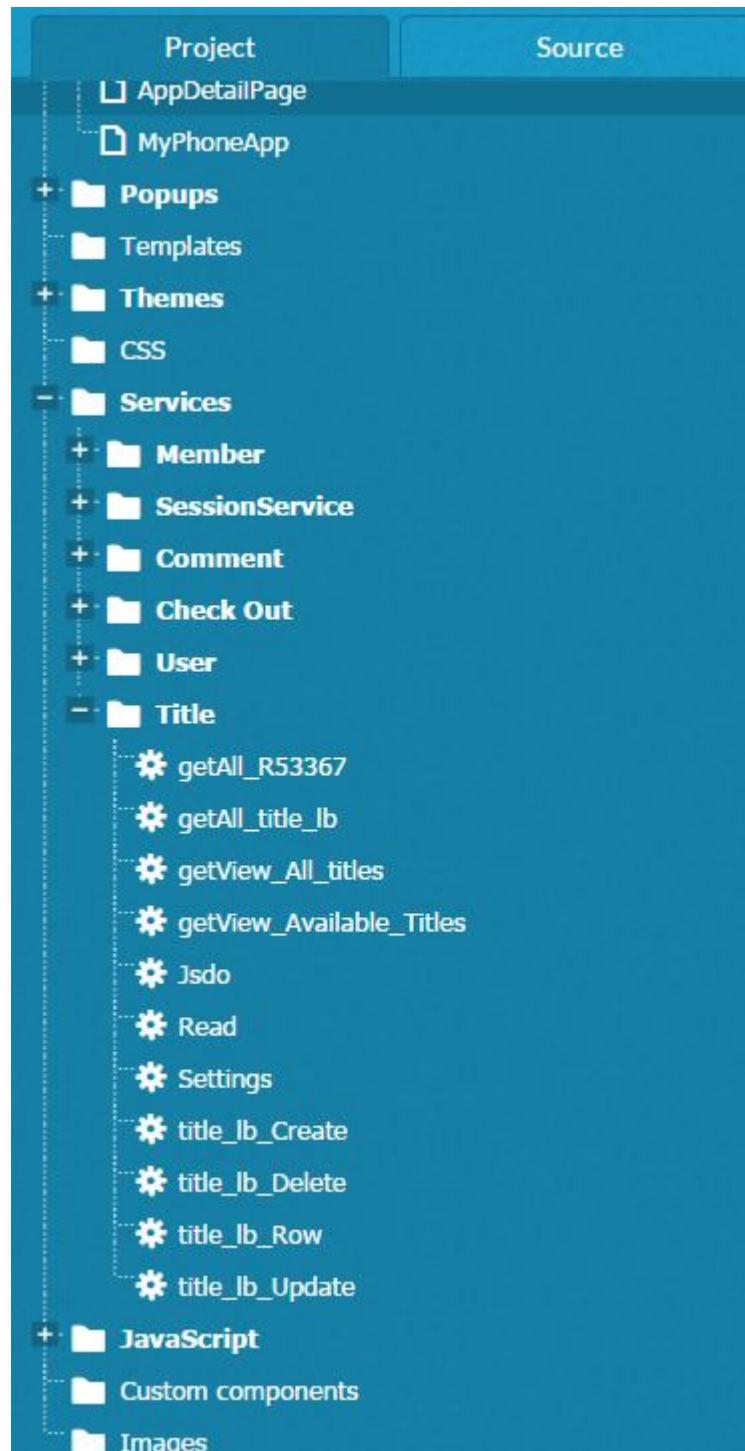


This **Label** component will display the titles. You can edit additional text formatting properties, such as margins, font size, and color for **Label** components.

Next, [add the services required to retrieve the Title records from Rollbase](#).

## Adding data sources to retrieve records

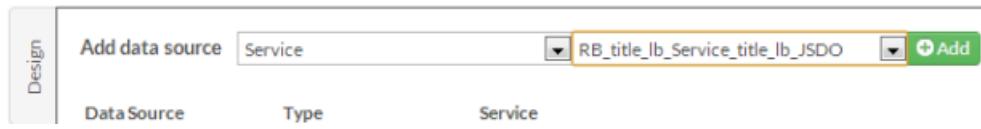
With the UI modified, you need to add data sources. Rollbase Mobile generated the JSDO services that are listed in the **Services** node of the **Project** tab in the sidebar. The following screen shows the variety of services available to access titles, which include those for performing create, read, and update operations:



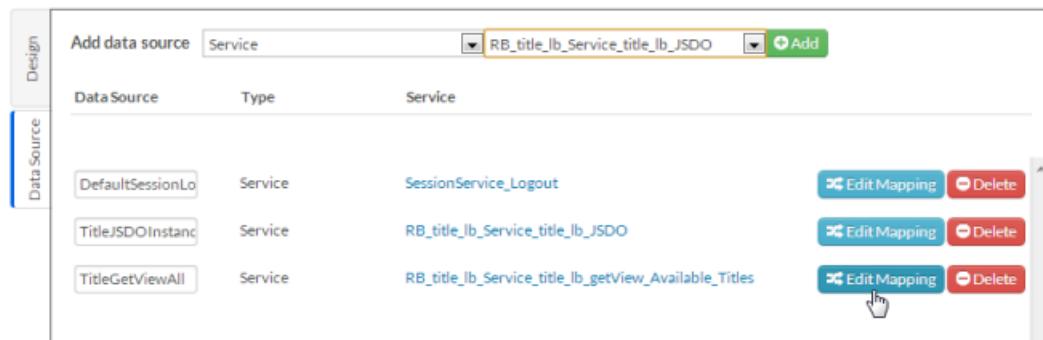
The service with the name ending in **JSDO** instantiates the service to access to title records. The **getView\_Available\_Titles** service returns the result set for the view of available titles.

To add these services as data sources for the Mobile App, follow these steps:

1. With the **AppDetailPage** open in a **Page Editor** tab, select the **Data Source** tab on the left side.
2. For **Add data source**, select **Service** as the type and **JSDO** as the JSDO Service:



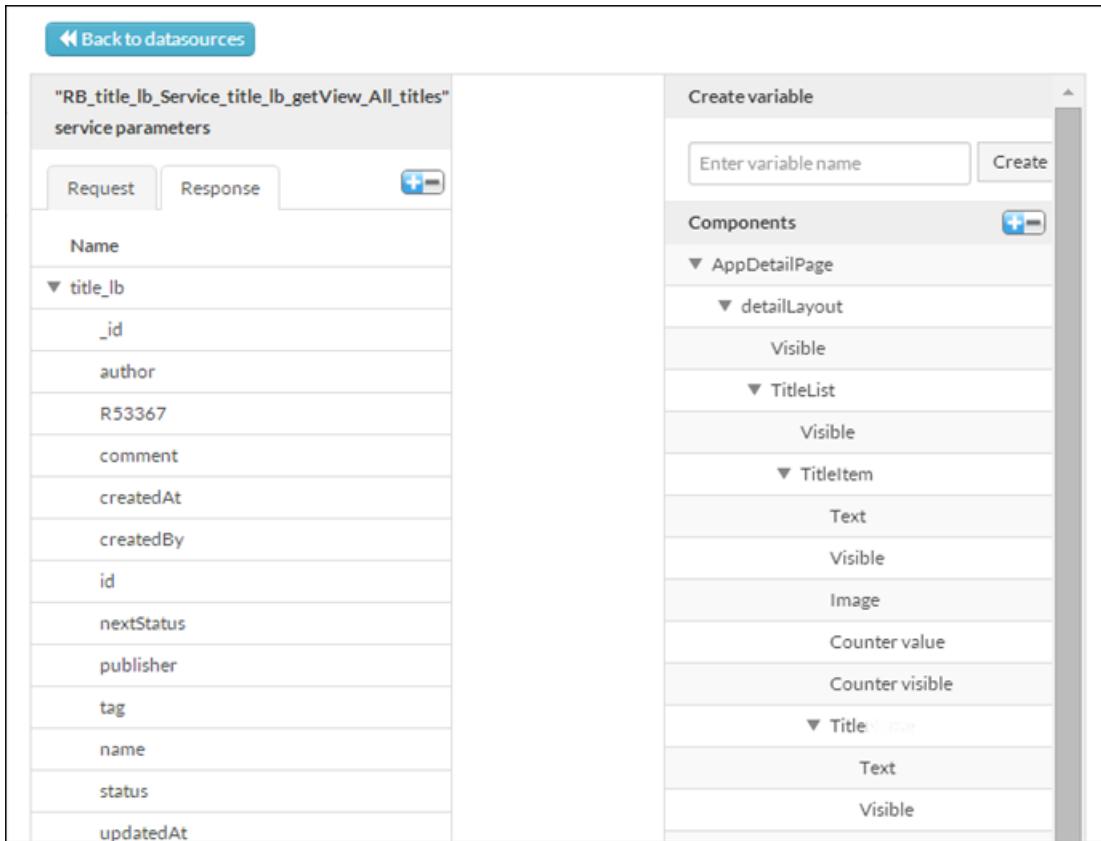
3. Click **Add** to add the data source.
4. Click the **Data Source** name to edit it. Change the name to `TitleJSDOInstance`. This will make it easier to identify the data source later when you create an event to execute it.
5. Repeat the previous procedure and add the `getView_All_titles` JSDO Service as a **Data Source**. Rename this data source to `TitleGetViewAll`.
6. For the `TitleJSDOgetViewAll` data source, click **Edit Mapping**:



The data mapping pane opens for the JSDO Service.

The left pane contains **Request** and **Response** tabs that identify parameters for the service operation, including options for specifying or modifying the values in JavaScript. The right tab allows you to map the parameters to UI components on the page or to JavaScript variables. In this example, you will map response parameters directly to the UI components you modified previously.

7. Select the **Response** tab and click  to expand all the fields of **title\_lb**. In the **Components** list on the right, click  to expand all the UI components under **AppDetailPage**:

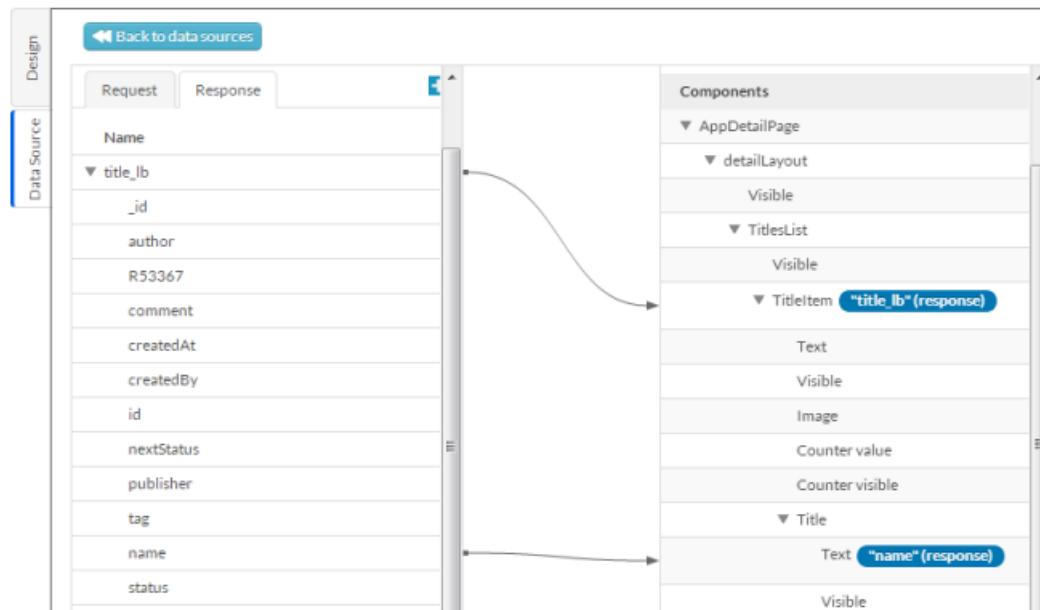


The screenshot shows the RDBMS interface with the following details:

- Left Panel (Response Tab):** Displays the response parameters for the service call "RB\_title\_lb\_Service\_title\_lb\_getView\_All\_titles". The parameters listed are: name, title\_lb, \_id, author, R53367, comment, createdAt, createdBy, id, nextStatus, publisher, tag, name, status, and updatedAt.
- Right Panel (Components List):** Shows the components defined under the "AppDetailPage" component. The components are:
  - detailLayout:** Contains the **TitleList** component, which is set to **Visible**.
  - TitleList:** Contains the **TitleItem** component, which is set to **Visible**. The **TitleItem** component has a **Text** field.
  - Title:** Contains the **Text** field, which is set to **Visible**.

8. Map response parameters to the components on the page. In this case, the page should only display the titles, not the other fields in a Title record, such as publisher. The title is stored in the `name` field of the response. Remember that you added the page components **TitleItem** and **Title** to display the data. Therefore, map the `title_lb` object to the **TitleItem** page component and the `name` response field to the **Title** page component:
- In the **Response** tab, click and drag `title_lb` and drop it on **TitleItem** (the list item).
  - Click and drag the `name` field and drop it on the **Text** field of the **Title** component.

The completed mappings appear as shown:



The arrows point in the direction of data flow, in this case, from the JSDO Service response to UI components on the page.

#### 9. Click **Back to datasources**.

Next, follow the procedure for [Adding Events to invoke the Services](#) on page 297.

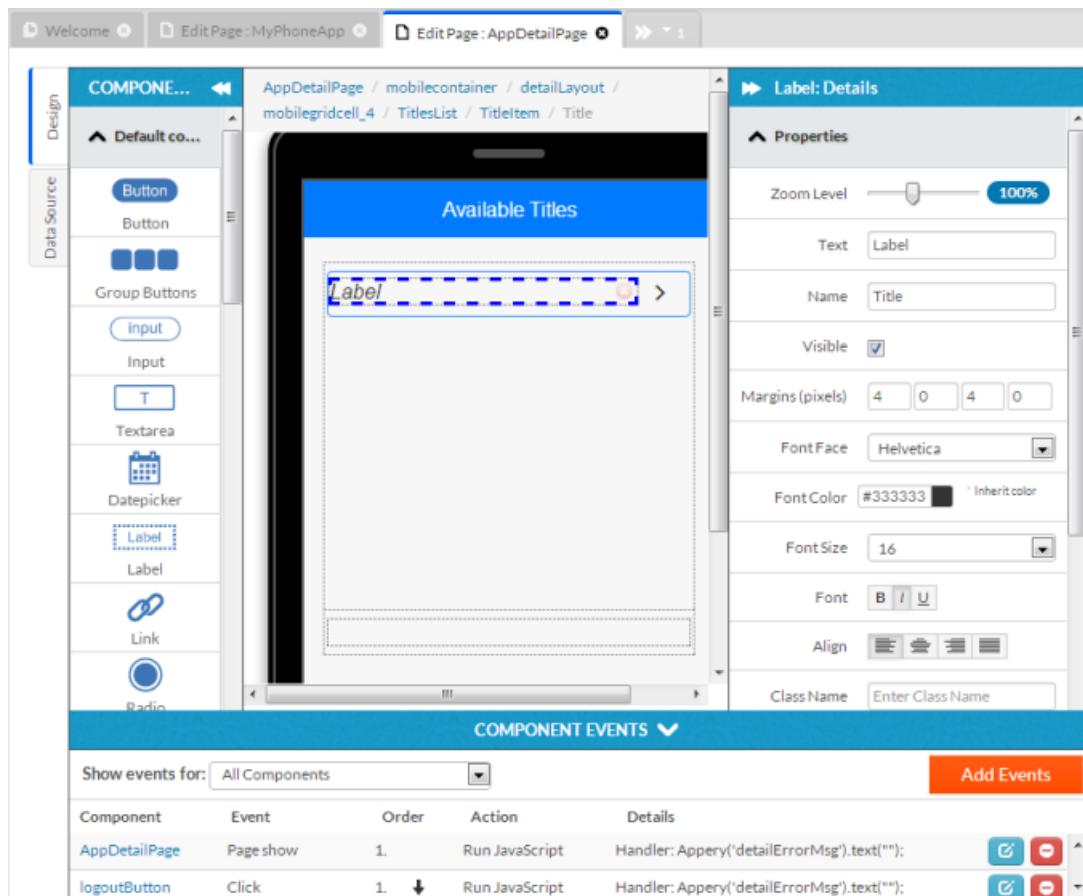
## Adding Events to invoke the Services

With the generated JSDO services mapped to the UI page components, you need to provide a way of invoking the services at runtime. Typical events include when a user taps or swipes the screen.

For this example, the list of titles should display immediately when the end-user logs in and the page displays. The **Load** event is appropriate for this use case. It will execute the data source you just defined, **TitleJSDOInstance** to instantiate the instance. On successful load, a second event will execute the **TitleGetViewAll** data source to retrieve the list of available titles.

Add the two events as follows:

1. Click the **Design** tab on the left side of the **Components** pane.
2. Click the **Component Events** tab at the bottom of the page to expand the controls for viewing and adding events:



3. Create the UI event to execute the JSDO service on page load as follows:

- In the **Show events for** list, select **AppDetailPage**.
- Click **Add event**.
- In the **Event** list, select **Load**.
- In the **Action** list, select **Invoke service**.
- In the **Data source** list that appears under **Details**, select **TitleJSDOInstance**.
- Click **OK**.

The new **Load** event definition appears in the list, as shown:

Component	Event	Order	Action	Details
AppDetailPage	Page show	1.	Run JavaScript	Handler: Appery('detailErrorMsg').text("");
AppDetailPage	Load	1.	Invoke service	Datasource: TitleJSDOInstance

4. Click the **Data Source** tab.

The data source pane appears, with the pane for creating and editing data source events at the bottom.

- In the **Show events for** list, select **TitleJSDOInstance**.
- Click **Add Events**.
- In the **Event** list, select **Success**.
- In the **Action** list, select **Invoke service**.

- e) In the **Datasource** list that appears under **Details**, select **TitleGetViewAll**.
- f) Click **OK**.

The new data source event definition appears in the list:

DATA SOURCE EVENTS				
Show events for: TitleJSDOInstance				
Data Source	Event	Order	Action	Details
TitleJSDOInstance	Success	1.	Invoke service	Datasource:TitleGetViewAll

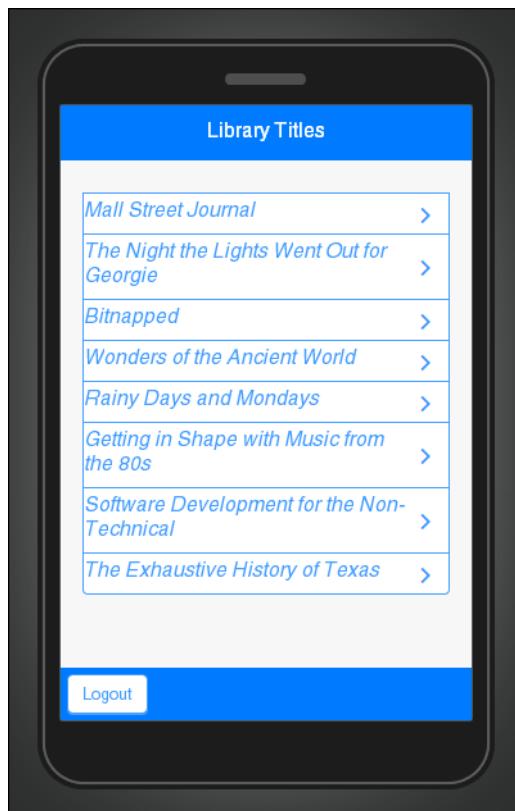
Congratulations! You have created a simple Mobile App based on an existing Web App. You can test it, as described in [Testing the Example Mobile App](#) on page 299.

## Testing the Example Mobile App

Follow these steps to test your Mobile App:

1. In the control bar above the **My Library** banner, click **Test**.
2. Enter your Rollbase user name and password and click **Login**.

On successful log in, the preview displays the default content page listing the available titles from the Library application:



3. You can click **Logout** to navigate back to the login page and close the **Mobile Review** tab.

In addition to the JSDO Services used in this example, you could add other functionality to this Mobile App, such as exposing other fields of Title records. Rollbase generates a set of JSDO services for all views you select and any objects that are related to the objects returned by those views. If you added other views, such as that showing current Check Outs, you could display which titles are checked out and when they will be returned. For authorized users, such as library employees, you could even allow creation and editing of Check Out records.

For operations other than viewing, you could use other generated services, such as:

- **Settings** — Specifies certain initialization options for the JSDO. Typically, you do not need to work with this Service, as Rollbase initializes it for your project.
- **JSDO** — The Instance JSDO Service instantiates the JSDO according to the JSDO catalog generated for the associated Rollbase object in the Rollbase cloud, making it available for use with the other JSDO Services.
- **get\*** — One or more Invoke JSDO Services, with an operation name such as `getView_All_object-name`. These Services return a record result set from the associated object in the Rollbase cloud according to a specified object view or relationship and stores it in the JSDO.
- **Read** — Reads a single record from the associated object and stores it in the JSDO.
- **Row** — Allows you to read a given record from the JSDO that you map into the UI.
- **Create, Update, and Delete** — Services that by default send a request to the associated Rollbase Mobile service to add, update, and delete a record identified in the JSDO. You can optionally set these Services to only operate on JSDO local memory (and not have the request go to the Mobile service). In this case, you need to handle sending the requests to the Mobile service yourself by adding custom JavaScript code to invoke the JSDO `saveChanges()` method.

For more information on using these JSDO Services in a Mobile App, see the sections on JSDO Services in the [Rollbase Mobile App Builder Documentation Home](#). For information on deployment options, see [Packaging and Deployment Options](#) on page 310.

## Creating and updating Rollbase Mobile Apps

The topics in this section describe how to perform common Mobile App development tasks.

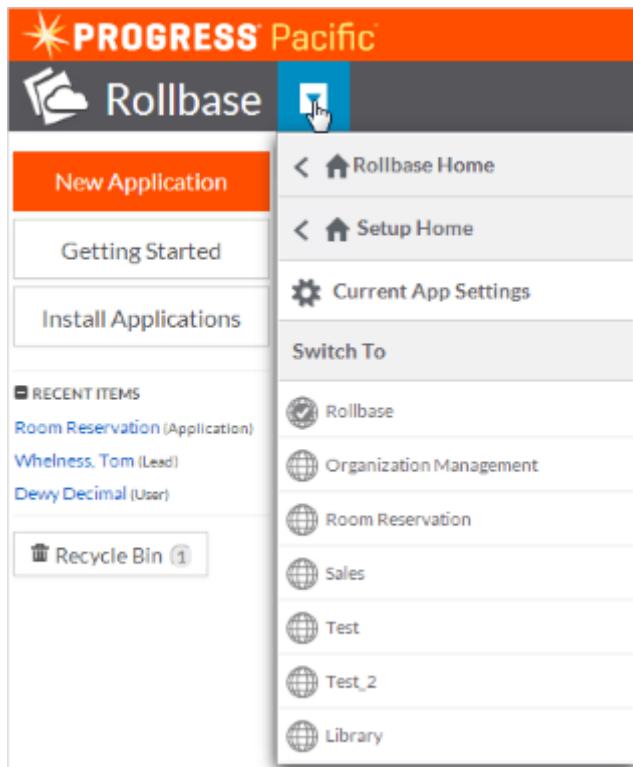
### Enabling Mobile App Development in Rollbase Private Cloud

The Mobile App Builder does not run on your infrastructure like other Private Cloud services. Instead, the Mobile App Builder is hosted by Progress Software. Therefore Private Cloud users must have a Progress Pacific account (a Progress ID) to use the Rollbase Mobile App Builder. A **Mobile Development Account** maps your Private Cloud user account to your Progress ID, thereby enabling you to create mobile apps with the Mobile App Builder.

You can set up your mobile development account by configuring it in your personal settings or Rollbase will prompt you to enter them when you attempt to access the Mobile App Builder.

Do the following to create a mobile development account and enable Mobile App Development:

1. From the header menu, select **Setup Home**:



The **Setup** home page appears:

A screenshot of the 'PERSONAL SETUP' page. At the top, there are three navigation links: 'Personal Setup' (with a person icon), 'Applications Setup' (with a wrench icon), and 'Administration Setup' (with a gear icon). Below these is a large grey box containing a sub-navigation section with a person icon and the text 'PERSONAL SETUP'. It says 'Manage your account settings and password' and provides links to 'My Settings', 'Change My Password', and 'Mobile Development'.

2. From the **Personal Setup** area, select **Mobile Development**.

The **Mobile Development** page appears.

3. Click **Edit** and do one of the following:

- If you have a Progress Pacific account **User Name** and **Password**, proceed to Step 4 on page 302.
- If you do not have a Progress Pacific account, select **Click here** to navigate to the Progress Pacific account creation page and then do the following:
  1. Fill in the appropriate details and select **Register** to create a Progress Pacific account. A success message appears specifying successful account creation and account activation details.
  2. From your email account, open the welcome email from Progress Software and select **Activate Account and Log in Today** to activate your account. You will be redirected to the **Mobile Development** page in Rollbase.

- If you want to manage your Progress Pacific account, select **Click here** to navigate to the Progress Pacific account edit page and then do the following:
  1. Review or update your personal information or Progress Pacific account password. And then, click **Save and Close** to save and close the edit profile page.
  4. Enter your Progress Pacific **User Name** and **Password**, and click **Save**.

A Progress Pacific account can be used in several customer tenants. However, it can only be used once per customer tenant.

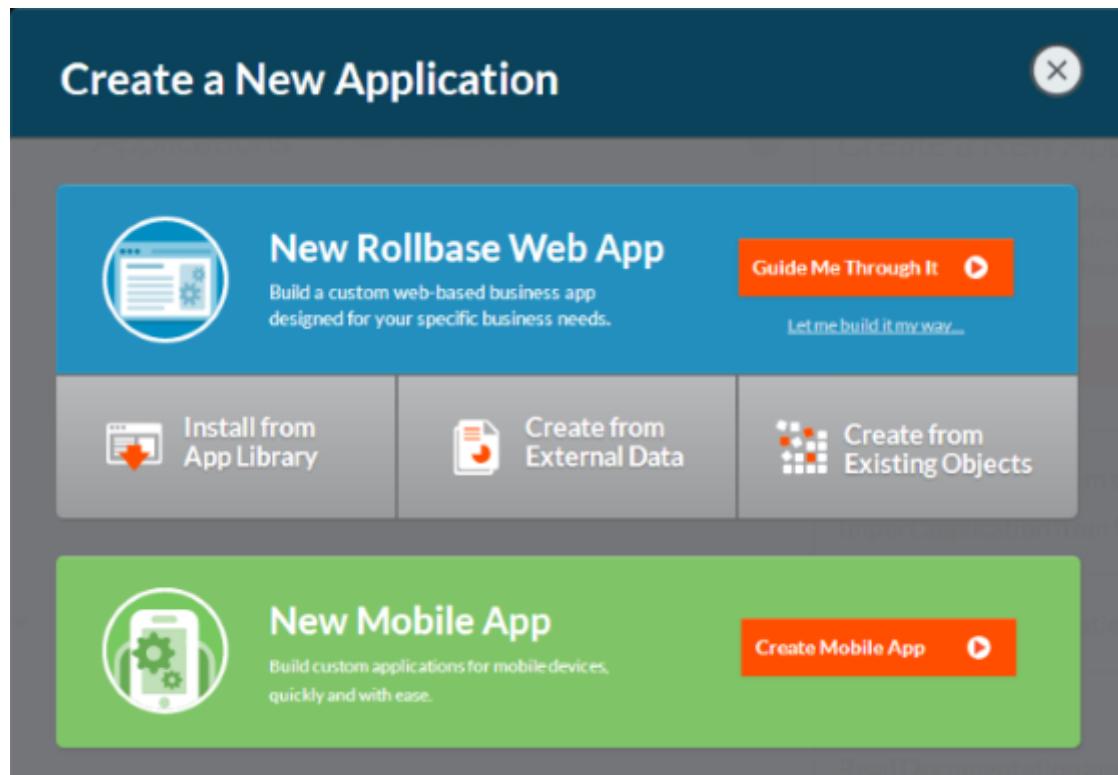
**Note:** To deactivate Mobile Development, you can **Delete** your Progress Pacific Account.

## Creating a Rollbase Mobile App project

If you are working in Rollbase Private Cloud, you must enable Mobile App development before you can create a Rollbase Mobile App project. For information about enabling Mobile App development in a Private Cloud, see [Enabling Mobile App Development in Rollbase Private Cloud](#) on page 300.

Follow these steps to create a Rollbase Mobile App project:

1. From the sidebar, select **New Application**. The **Create a New Application** screen displays:



2. Click **Create Mobile App**.

The **New Mobile App** screen opens:

3. Specify the following:

- **Mobile App Name** — A string that uniquely identifies the App.
- **Description** — A free-form description of the Mobile App.
- **Template** — A template that specifies the form factor (typically associated with a Mobile device) for which you are building the App. The options include templates that provide the basic design canvas on which you add visual components, such as lists and fields. The default is the **Phone App** template. You can also select a basic **Tablet App** template or select from a user-defined template that you have previously defined.
- **Create Option** — The options available to you to create the Mobile App. Select one of the following options:
  - **From Scratch** — To create an empty Rollbase Mobile App, which you can later edit to create new Rollbase objects or add existing Rollbase objects and views to export for access in the Mobile App Builder.
  - **Based on Existing Web App** — To create a Rollbase Mobile App using the core Rollbase objects of an existing Rollbase Web app. You can also later edit this App to create new Rollbase objects or add other existing objects.

4. Select one of the following from the top-right corner of the **New Mobile App** page:

- **Save** to save the Rollbase Mobile App and navigate to the Setup Mobile Application page.
- **Save and Design** to save the Rollbase Mobile App and launch the Mobile App Builder to design the Mobile App.
- **Cancel** to undo the Mobile App creation and navigate to the Rollbase home page (or the page where you selected **New Application**).

## Importing Mobile App projects created with OpenEdge Mobile

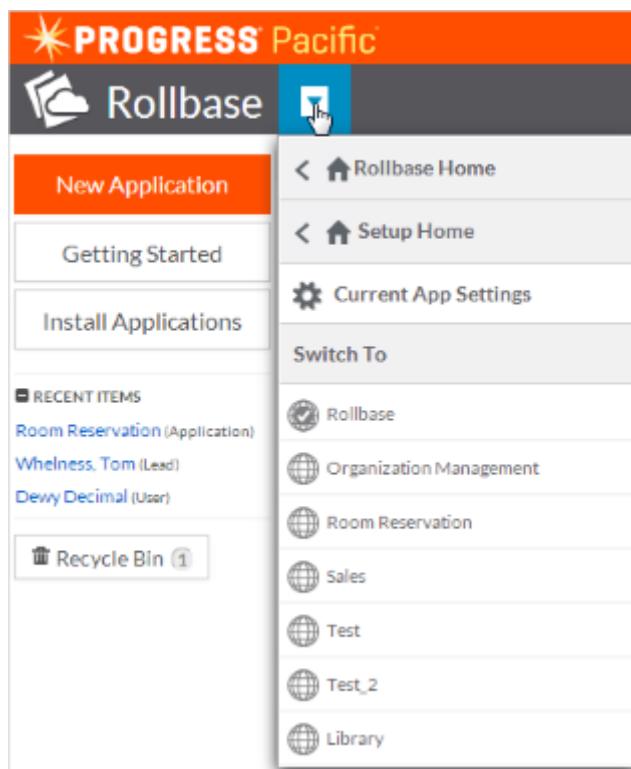
If you have a Mobile project backup (.zip) from the Mobile App Builder, especially for a Mobile App **not** created in Rollbase, you can import it and create a new Mobile App in Rollbase.

Typically, you use this option when you want to import an OpenEdge Mobile App project in Rollbase as a new Mobile App. You first make a backup of the OpenEdge Mobile App project (.zip) from the Mobile App Builder, then import it in Rollbase as a new Mobile App.

You can also backup a Rollbase Mobile App project from Mobile App Builder, but importing the backup file (.zip) in Rollbase as a new Mobile app does not import the Rollbase objects associated with the Rollbase Mobile App. Therefore, Progress Software recommends that you **do not** create a new Rollbase Mobile App from a Mobile App Builder backup of a Rollbase Mobile App. Instead, generate XML for the Mobile App in Rollbase (see [Generate XML](#) on page 109) and use the generated XML to create a new Rollbase Mobile App (see [Installing and Updating Applications from XML](#) on page 114), because this process saves and imports all the Rollbase components associated with the Rollbase Mobile App.

To create a new Rollbase Mobile App from a Mobile project backup:

1. From the header menu, select **Setup Home**.



2. In the **APPLICATIONS SETUP** section, click **Applications**:

**PERSONAL SETUP**  
Manage your account settings and password  
[My Settings](#) | [Change My Password](#)

**APPLICATIONS SETUP**  
Create, install, update, manage, customize and integrate applications  
[Applications](#) | [Objects](#) | [Tabs](#) | [Portals](#) | [Hosted Files](#) | [Application Directory](#)  
| [SOAP API](#) | [REST API](#) | [Code Generator](#)

**ADMINISTRATION SETUP**  
Create and manage users, roles, authentication, backups, batch jobs and more  
[Users](#) | [Roles](#) | [Transfer Owners](#) | [User Access Log](#) | [Support Access](#) | [Settings](#)  
| [Account Settings](#) | [Billing And Support Settings](#) | [Currency Codes](#) | [Exchange Rates](#)  
[Whitelist](#) | [Backup](#) | [Batch Jobs](#) | [Global Text Search](#) | [System Jobs](#) | [Portal Visitors](#)  
[Online](#) | [System Events Log](#) | [System Errors Log](#)

A list of existing applications displays:

[Personal Setup](#) [Applications Setup](#) [Administration Setup](#) [return to Library](#)

[Application Setup > Applications](#)

 Tip: [Learn more](#) about Applications and installation process.

Applications		Reorder Applications		Import From		View Installation Log		
Action	Order No	Application	Version	Deployed	Published	Updates Available	Installation Date	Publisher
Edit	1	Rollbase	5	✓			06/13/2014	Progress Rollbase
Edit	2	Organization Management	6	✓			06/13/2014	Progress Rollbase
Edit   Publish	3	Room Reservation	3	✓			06/23/2014	
Edit   Publish	4	Library	13	✓			07/11/2014	
Total 4								

### 3. Click **Import From**.

The **Import From** screen displays:

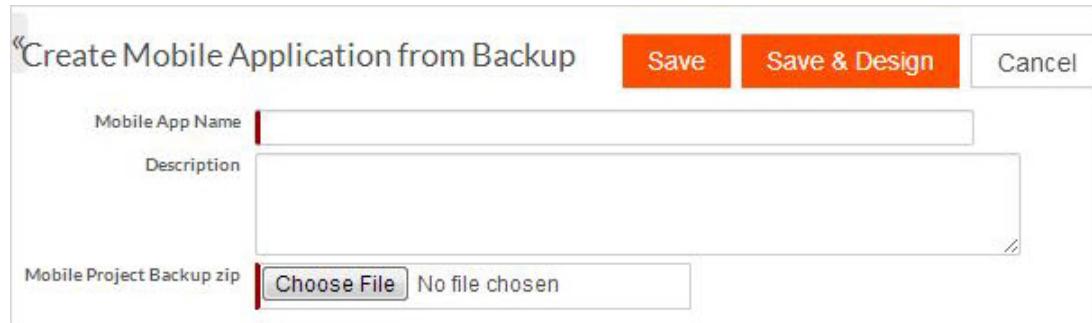
[Import From](#)

Type	Description
<input type="radio"/> Progress Rollbase XML File	This option will install application developed in another Customer and serialized as XML file.
<input type="radio"/> Salesforce.com (Force.com)	This option will import your currently active Salesforce.com application including all of its objects, fields, page layouts and data.
<input type="radio"/> Convert MS Access Database	This option allows you to upload a Microsoft Access Database (mdb or accdb) and automatically convert it into a working web application (recommended when you want to start from an existing MS Access application).
<input type="radio"/> OpenEdge Service	Import Objects and Relationships from uploaded OpenEdge JSOO catalog file
<input type="radio"/> Mobile Project Backup	This option will create mobile application from a project backup archive (zip artifact)

[Next >](#) [Cancel](#)

### 4. Select **Mobile Project Backup** and click **Next**.

The **Create Mobile Application from Backup** page appears:



The dialog box has the following fields and buttons:

- Mobile App Name:** A text input field.
- Description:** A text area for a free-form description.
- Mobile Project Backup zip:** A file input field with a "Choose File" button and a message "No file chosen".
- Buttons:** "Save", "Save & Design", and "Cancel".

5. Specify the following details:

- **Mobile App Name:** A single string (no embedded spaces) that uniquely identifies the Rollbase Mobile App.
- **Description:** A free-form description of the Rollbase Mobile App.
- **Mobile Project Backup zip:** Click **Choose File** and select the Mobile Project backup file (.ZIP).

6. Select one of the following from the top-right corner of the **New Mobile App** page:

- **Save** to save the Rollbase Mobile App and navigate to the Setup Mobile Application page.
- **Save and Design** to save the Rollbase Mobile App and launch the Mobile App Builder page to design the Mobile App.
- **Cancel** to undo the Rollbase Mobile App creation and navigate to the Rollbase home page (or the page where you selected **Setup Home**).

## Editing Rollbase Mobile App projects

Editing a Rollbase Mobile App project is similar to editing a Rollbase Web App, but uses a Setup Mobile Application page to do the editing instead of a Setup Application page. For information on the similar features of the Setup Application page, see [Editing Applications](#) on page 105. In addition, when editing a Mobile App, you have the option to open the App in the Mobile App Builder to define its UI and to add or remove object views that you export to the Mobile App Builder to access Rollbase data. This section thus describes the editing features that are available on the Setup Mobile Application page.

Follow these steps to edit a Rollbase Mobile App project:

1. From the Rollbase header menu, select and open your Mobile App.

The **Setup Mobile Application** page appears. The following shows an example **Setup** page in hosted Rollbase:

## Setup Mobile Application: My Library

[Edit](#) [Publish](#) [Delete](#) [More actions...](#)

[Objects \(5\)](#) [Dependent Objects \(1\)](#) [Roles](#) [Seed Records](#) [Permissions](#) [Audit Trail](#)

### Application Details

Application Name **My Library**

[Design](#)

[Sync Metadata](#)

[Test](#)

[Mobile Hosting](#)

#### Mobile Hosting

Deployed

Hidden

Template Phone App

Version 1

#### Description

Core Objects [Check Out, Comment,](#)

[Member, Title, User](#)

Sync Metadata for

Dependent Objects

### System Information

Last Updated By [Girish Mahadevan](#)

Created By [Girish Mahadevan](#)

Last Updated At 11/24/2014 12:25 PM

Created At 11/24/2014 12:25 PM

ID 120890547

Original ID 120890547

## 2. Select:

- **Edit** for Mobile App metadata editing in Rollbase. This allows you to modify the **Deployment Status**, **Application Details**, **Authentication Details**, **Core objects**, **Dependent Objects**, and **User Roles** associated with the Mobile App. Most of the details are set at the time you create the Mobile App creation. The following options can only be set or modified in the edit page:
  - **Deployment Status:** In this area, you can deploy/undeploy or hide the application. Note that a deployed application cannot be deleted.
  - **Authentication Details:** This options is only available in Rollbase Private Cloud. In this area, you specify which users can access your Mobile App. By default, all users in the `User` object are authenticated. You can change the `Object` field value to specify another object with a Portal User attribute.

Additionally, you can allow any public user to access the mobile app anonymously. To enable Anonymous access, do the following :

### 1. Select **Anonymous\_Access**.

2. In Progress Mobile App Builder, change of the value of `authenticationModel` attribute in the `appconfig.js` file of your Mobile App from `Form` to `Anonymous`.

**Note:** If you choose to disable **Anonymous\_Access**, you must clear the **Anonymous\_Access** check box and change of the value of `authenticationModel` attribute in the `appconfig.js` file of your Mobile App from `Anonymous` to `Form`. In Mobile App Builder, the `appconfig.js` file is available in the `JavaScript` folder of your Mobile App.

- **Dependent Objects:** In this area, you can add any available dependent objects to your Mobile app if you want them to be included in your Mobile application before it is installed in another customer tenant.

---

**Note:** Rollbase adds certain dependent objects to your Mobile App to maintain application integrity. These dependent objects cannot be removed from the Mobile App.

---

- **Sync Metadata for Dependent Objects:** This option specifies that the selected dependent objects must also be synced as part of the **Sync metadata** process.
- **Publish** to publish the Mobile App to the Rollbase App directory.
- **Delete** to delete the Mobile App permanently.
- **More actions** to perform advanced actions. Most of these application actions are described under [Application Actions](#) on page 107. The other actions specific to Mobile Apps are:
  - **Export Web Resources** — This option (not shown in the page from hosted Rollbase) is only available in a Rollbase Private Cloud. Click this option to download a zip file that contains the Mobile App to your system directory. You can then host the Mobile App on your servers for access by users. For more information, see [Packaging and deploying hosted browser apps](#) on page 310.
  - **Generate Metadata File(s)** — This option shows a preview of metadata (JSON structure) in the application. You can view and download (a .zip file) the metadata information. You can use this option to troubleshoot any discrepancies before exporting your Mobile App to Mobile App Builder.
- **Design** to open the Mobile App Builder to the App project page, where you can update its UI, data sources, the mappings between them, and their logic.
- **Sync Metadata** transfers any new Rollbase object definitions, or changes in previously exported Rollbase object definitions, to the Mobile App Builder project for the App, and displays a confirmation dialog to verify that you want to do this.
- **Test** — Opens the listed App for testing in the device emulator of the Mobile App Builder, which displays in a separate Rollbase tab.
- **Mobile Hosting** — This option is only available in Hosted cloud. If the Mobile App is not hosted, click **Mobile Hosting** to specify a sub-domain name as part of a Rollbase-specified URL to host your App as a Mobile Web App. This action changes **Mobile Hosting** to **Mobile Hosting in progress** (disabled), then to **Disable Mobile Hosting** (enabled) and **Refresh Mobile Hosting** (enabled), depending on the state of the hosting action.

Clicking **Disable Mobile Hosting** removes the hosted Mobile App from the specified sub-domain. Clicking **Refresh Mobile Hosting** refreshes the Mobile App with the latest changes and then hosts the latest Mobile App on the same sub-domain as it was earlier hosted.

For more information, see [Packaging and deploying hosted browser apps](#) on page 310.

3. Create or edit **Objects**, **Depended Objects**, **Roles**, **Seed Records**, **Permissions**, and view **Audit Trail** from the respective areas.

## Adding objects to a Mobile App

To add an object to a Mobile App, you must add it to the application definition and synchronize the metadata as follows.

1. In Rollbase, navigate to **Application Settings** for the Mobile App.

The **Setup** page displays. The following screen shows an example **Setup** page from Rollbase Hosted Cloud:

Setup Mobile Application: My Library

Application Details

Mobile Hosting

Deployed ✓

Template Phone App

Description

Core Objects Check Out, Comment, Member, Title, User

Sync Metadata for Dependent Objects

System Information

Last Updated By Girish Mahadevan

Last Updated At 11/24/2014 12:25 PM

ID 120890547

Created By Girish Mahadevan

Created At 11/24/2014 12:25 PM

Original ID 120890547

2. Click **Edit**.

Application details page opens.

3. In the **Core Objects** or **Dependent Objects** areas, move the object you want to add to the **Selected Objects** pane.
4. Click **Save**. The **Setup** page appears.

## Testing Rollbase Mobile Apps

In the Mobile App Builder, the **Test** button launches an emulator based on the type of Mobile App you are building (Phone App or Tablet App for the Web or a specific iOS or Android device). You can then see the App run as it might appear on the target platform, but displayed in a new tab of your HTML5 Web browser. You can also select options for a test by clicking the drop-down next to **Test**. If you are in the Rollbase Mobile Application details page, then you can select **Test** to run the same test.

To test a hybrid app running on a device, package the app in a mobile device deployment file (IPA for Apple iOS or APK for Android), and deploy it for testing on the appropriate mobile device. For more information, see [Packaging and deploying Apps for iOS](#) on page 310 or [Packaging and deploying Apps for Android](#) on page 313.

# Packaging and Deployment Options

As described earlier, you can package and deploy the same mobile app in multiple ways. These include:

- [As an app mobile users access with a browser](#)
- As a native mobile app:
  - [For iOS](#)
  - [For Android](#)

## Packaging and deploying hosted browser apps

From the Mobile App Builder, you can export a package that you can deploy yourself on any web server. See

[http://documentation.progress.com/output/mobile-app-builder/index.html#/mab/mab\\_deploy](http://documentation.progress.com/output/mobile-app-builder/index.html#/mab/mab_deploy). Or, you can use the Rollbase environment to host an app. The options available from the Rollbase environment to deploy a hosted browser app depend on whether you are working in hosted Rollbase or in a Rollbase Private Cloud.

### Deploying a browser app in hosted Rollbase

To deploy a Mobile App to be hosted by Rollbase, select the **Mobile Hosting** action from the **Setup Mobile Application** page for your Mobile App. This action allows you to specify a sub-domain name as part of a Rollbase-specified URL to host your Mobile App as a Web App. You can distribute this URL to users who want to access your Mobile App.

After the Mobile App is hosted, you can view or navigate to the app from the **Mobile Hosting** field in the **Setup Mobile Application**. For more information about the Mobile hosting option, see [Editing Rollbase Mobile App projects](#) on page 306.

### Deploying a browser app in Rollbase Private Cloud

To deploy a browser app in a Rollbase Private Cloud, select the **Export Web Resources** action from the **More Actions** tab in the **Setup Mobile Application** page for your Mobile App. This action downloads a zip file that contains the Mobile App to your system directory. You can then host the Mobile App on your servers for access by users who want to access your Mobile App using a browser. You have only to provide the URL to users who want to access the App.

Note that while you might host your Mobile App on any server, it accesses Rollbase data in your Private Cloud, potentially across different Web domains. Therefore, you must ensure, if necessary, that the Rollbase Mobile Web applications that provide the data on your Rollbase server are configured to support cross-origin resource sharing (CORS). Otherwise, your Mobile App might not be able to access the Rollbase objects for which it is built.

## Packaging and deploying Apps for iOS

Packaging a hybrid app for iOS (IPA file) requires a certificate and a provisioning profile. These credentials are required during development as well as deployment. For more information on creating the certificate and provisioning profile, see [Deploying iOS Apps](#) on page 312.

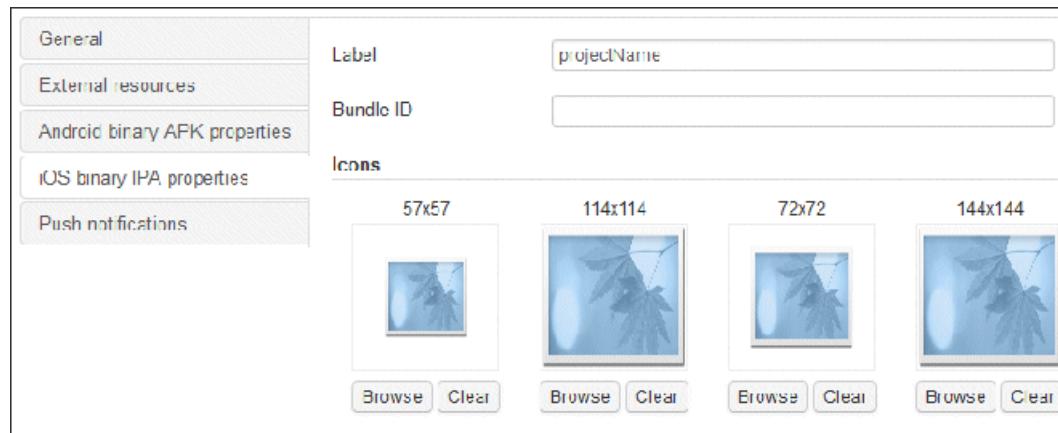
## Packaging iOS Apps

To build and package a hybrid App for iOS:

1. From the Rollbase Mobile application's details page, click **Design** for the Mobile App you want to deploy.

The Mobile App Builder opens.

2. In the Mobile App Builder, choose **Project > App settings** and click **iOS binary IPA properties**.



3. Enter the Mobile App name in the **Label** field.
4. Enter the Bundle ID in the **Bundle ID** field.
5. The icon is the graphic that will launch the Mobile App on an iOS device. Upload an icon file using the **Browse** button in the **Icons** area.
6. Upload a launch image file for the application using the **Browse** button in the **Launch Images** area.
7. To upload the certificate:
  - a) Click **Change** next to Certificate file.
  - b) Click **Upload file**.
  - c) Click **Upload a file** and locate the certificate. Once the upload is complete, click **Back to files list**.
  - d) Click on the certificate file and click **Select**.
8. Enter the certificate password.
9. To upload the provisioning profile:
  - a) Click **Change** next to Provisioning profile file.
  - b) Click **Upload file**.
  - c) Click **Upload a file** and locate the provisioning profile. Once the upload is complete, click **Back to files list**.
  - d) Click on the provisioning profile file and click **Select**.
10. Click **Save**.
11. To package the IPA, click **Export** to display the list of supported output formats, and choose **.ipa** under **Release Binary**.

The IPA file is generated and output to your default downloads folder.

### Testing the IPA directly on an iOS device

You can test your app during development by copying the IPA directly to an Apple device.

To copy the IPA to an iOS device:

1. Open iTunes on your computer.
2. Drag the IPA into the iTunes Library.
3. Plug the iOS device into the computer.
4. Drag the IPA from the iTunes Library onto the icon showing the device.

If you get an installation error, verify that the UUID of the device is registered correctly and that the provisioning profile includes that device. For more information, see [Deploying iOS Apps](#).

## Deploying iOS Apps

Distributing a hybrid app for iOS (IPA file) requires registration as an Apple Developer, a developer certificate, and a provisioning profile. The following sections reference information to help you create your own credentials.

---

**Note:** Though not required, you can do this more easily on a Mac.

---

For more detailed information visit:

<https://developer.apple.com/devcenter/ios/index.action>

### Apple Developer

To register as an Apple Developer, visit this website:

<https://developer.apple.com/programs/register/>

### iOS Provisioning Portal

To create the certificate and provisioning profile, use the iOS Provisioning Portal. This portal is part of the iOS Dev Center. To access the portal, log in at the following link with the Apple ID you used to register as a developer:

<https://developer.apple.com/devcenter/ios/index.action>

Click on the iOS Provisioning Portal link.

### Certificates

The following video has more information on certificates:

<https://developer.apple.com/ios/manage/overview/index.action>

Note that a certificate must be in P12 format in order to be used in the Mobile Application Builder.

### App IDs

An App ID has three parts: Description, Bundle Seed ID (App ID Prefix), and Bundle Identifier (App ID Suffix). The Description of the App ID can be anything, e.g., OEMobile. The Bundle Seed ID (App ID Prefix) is assigned for you and is assigned to be your Team ID. The Bundle Identifier can be unique for each application or use a wildcard for many applications. Although it is possible to use "\*" by itself, the recommended practice is to use the following notation:

com.progresssoftware.\* or com.progresssoftware.applicationname.

It is important to remember what you entered for the Bundle Identifier. This will be used when creating the IPA in the Mobile App Builder. The following link describes Bundle Identifiers:

[Uniform Type Identifier Concepts](#)

The following video has information about app IDs:

<https://developer.apple.com/ios/manage/overview/index.action>

## Devices

Each device must be registered. This is accomplished by using the device UUID. The UUID can be found by running a free UUID app on the device.

The following video has more information on devices:

<https://developer.apple.com/ios/manage/overview/index.action#>

## Provisioning Profiles

A Provisioning Profile has four parts: Profile Name, Certificate(s), App ID, and Device(s). The Profile Name can be anything, e.g., RollbaseMobile. The following video has more information on provisioning:

<https://developer.apple.com/ios/manage/overview/index.action#>

The following link also has more information:

[http://developer.apple.com/library/ios/#technotes/tn2250/\\_index.html](http://developer.apple.com/library/ios/#technotes/tn2250/_index.html)

## Packaging and deploying Apps for Android

Packaging a hybrid app (APK file) for Android requires a signing key/certificate, but during development and testing, the signing key/certificate inputs are not required. For more information on creating the signing key/certificate, see [Deploying Android Apps](#) on page 315.

## Packaging Android Apps

To build and package a Mobile Native App for Android:

1. From the Rollbase Mobile application's details page, click **Design** for the Mobile App you want to deploy.  
The Mobile App Builder opens.
2. In the Mobile App Builder, choose **Project > App settings** and click **Android binary APK properties**.

3. Enter the Mobile App name in the **Label** field.
4. The launcher icon is the visual representation of your app on the mobile device. To upload an icon:
  - a) Click **Change** next to **Icon**.
  - b) Click **Upload file**.
  - c) Click **Upload a file** and locate the icon file. Once the upload is complete, click **Back to images list**.
  - d) Click on the icon file and click **Select**.
  - e) Click **Upload a file** and locate the provisioning profile. Once the upload is complete, click **Back to files list**.
  - f) Click on the provisioning profile file and click **Select**.
5. Enter the Version code.
6. Enter the Version name.
7. Enter the Package name.
8. Enter the Minimum SDK version.
9. Enter the Target SDK version.
10. Enter the Maximum SDK version.
11. To upload the Keystore file:
  - a) Click **Change** in the Keystore file field. Click **Upload file**.
  - b) Click **Upload a file** and locate the Keystore. Once the upload is complete, click **Back to files list**.
  - c) Click on the Keystore file and click **Select**.
  - d) Enter the Key alias.
  - e) Enter the Key password.
  - f) Enter the Keystore password.

12. Click **Save**.
13. To package the APK, click **Export** to display the list of supported output formats, and choose **.apk** under **Release Binary**.

The APK file is generated and output to your default downloads folder.

## Testing the APK directly on an Android device

During development, there are several ways to get the newly created APK on an Android device for testing.

One common approach uses a Dropbox account:

1. Create an account on <http://dropbox.com>.
2. Click and drag the APK into Dropbox.
3. Open <http://dropbox.com> from a browser on the mobile device and log in.
4. Click on the APK from within the Dropbox application. This will begin downloading the application onto the device.
5. When the download is complete, click on the downloaded file on the mobile device. Follow the device directions for installing the app.

## Deploying Android Apps

For detailed information on Android development and deployment, visit this website:

<http://developer.android.com/index.html>

## Versioning your Mobile App

Versioning is important for application upgrades and maintenance. Users need to have specific information about the application version that is installed on their devices and the upgrade versions available for installation, and other applications and services might need to check the version of your application for compatibility issues. For more information go to:

<http://developer.android.com/tools/publishing/versioning.html>

## Package name

The Application package is an identifier of the application. This identifier is unique among all the apps installed on a device at a given moment; there cannot be two apps with the same Application package installed at the same time. It is also unique on the Android market; there cannot be two apps with the same Application package on the Market. Conflict over the Application package with unforeseen third-party apps is possible. Using the Java package name convention (`com.mydomain.myapp`) for the Application package name is recommended to avoid such conflict.

## Signing your application

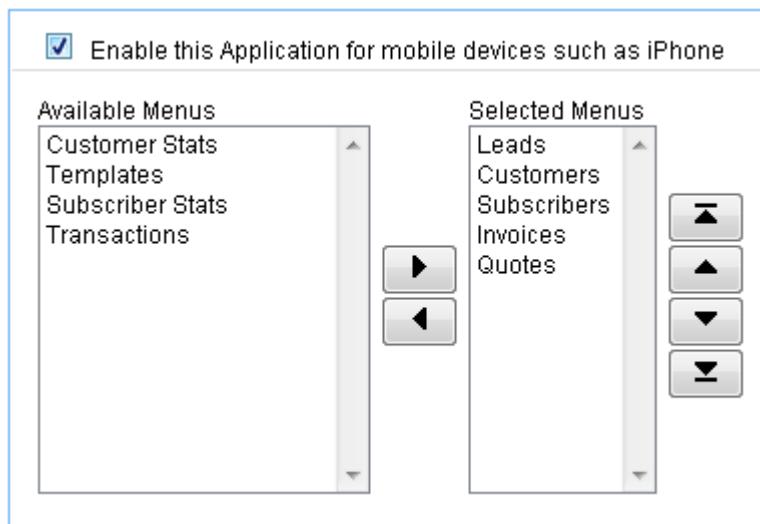
The Android system requires all installed applications be digitally signed with a certificate whose private key is held by the application's developer. To test and debug your application, the build tools sign your application with a special debug key. When you are ready to release your application for end-users you must sign it with a suitable private key. For more information go to:

<http://developer.android.com/tools/publishing/app-signing.html>

# Mobile-Web Enabled applications

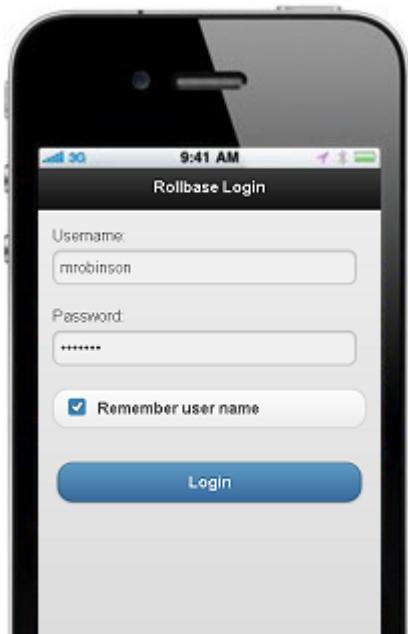
All Progress Rollbase applications can be easily accessed on iPhone and Android devices using a user interface specifically designed for smart phone browsers. The screenshots below illustrate the core capabilities of the Rollbase Mobile-Web user interface.

Existing Rollbase applications can be mobile-enabled in just a few clicks. Navigate to the application definition and from the **More Actions** drop-down menu, select **Set Mobile-Web Options**. Select the check box to enable Mobile-Web and move the menus, or tabs, that contain the views to expose to mobile users to the **Selected Menus** list.



## Secure Log In

Users can log in to the Rollbase.com Hosted Cloud or a Private Cloud instance via a dedicated mobile login page. The standard Rollbase login page automatically redirects smart phone browsers to this mobile-friendly page.



## Browse Applications

Users who have permissions to access a Mobile-Web enabled app will see it in their list of available applications when they log in.



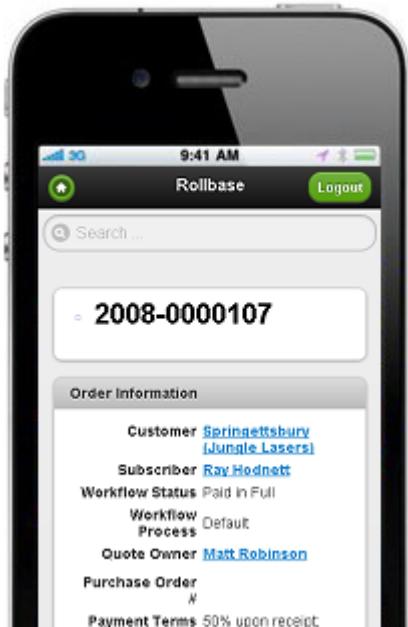
## Browse Records

Users can select views they have access to and navigate the records in those views. Flagged records will be shown with the standard flag icon and unviewed records will be shown in bold blue text.



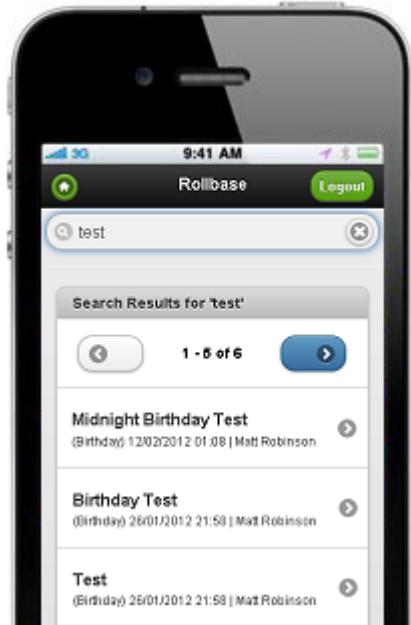
## View Records

Selecting a record displays a detail view showing each of the record's values organized into sections defined by the object's default view page layout. Administrators can customize how this page is organized. Users can select Email addresses to send emails and phone numbers to make a call directly from their phone.



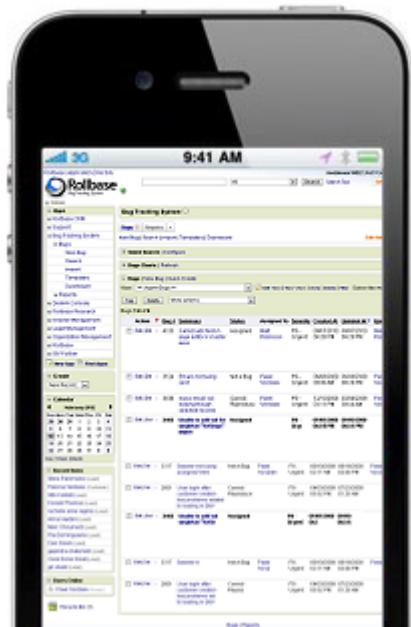
## Global Search

A global text search box at the top of every page in allows quick and easy searching through records for which the user has access. Results are shown in a pageable list from which the user can drill down to view individual records.



## Switch to Browser Version

Users can switch to the standard Rollbase browser interface to access all of the features not available in the Mobile UI by clicking **Switch to Browser Version** at the bottom of any page.





---

# Integrating with Outside Sources

---

Rollbase offers a variety of mechanisms for integration. You can bring external data and external structures into Rollbase applications and you can make Rollbase metadata and data available to external applications.

For details, see the following topics:

- [Creating Rollbase Objects from OpenEdge Services](#)
- [Using DataDirect Cloud to Access External Data](#)
- [Creating Rollbase Applications from Microsoft Access](#)
- [Creating Rollbase Applications from Salesforce Applications](#)
- [Using External Tables as Rollbase Objects](#)
- [Importing Data](#)
- [Deleting Multiple Records by Importing a Spreadsheet](#)
- [Exporting from Views and Reports](#)
- [Integrating with Google Applications](#)
- [Using SOAP or REST to Integrate with Rollbase](#)

# Creating Rollbase Objects from OpenEdge Services

External Rollbase objects can access business entities hosted on an OpenEdge AppServer. To accomplish this, The OpenEdge services must be exposed as REST services and defined in an OpenEdge JSDO Catalog file. When you associate these business entities with Rollbase external objects, the data persistence for these objects is handled by the OpenEdge AppServer, which runs the associated ABL logic prior to storage in the OpenEdge database. This feature enables organizations to integrated data from new or pre-existing OpenEdge applications with Rollbase applications.

To provide integration with Rollbase, OpenEdge developers need to create JSDO Catalogs describing the classes that they want to expose to users. This is described in the OpenEdge documentation, specifically OpenEdge® Development: Mobile Applications. The same mechanism, and the same objects in a catalog can also be used for both Rollbase and Mobile applications. Similarly, if you have exposed services for mobile or tablet interfaces, then those can be used as the basis of Rollbase Objects.

OpenEdge developers will also need to deploy the business entity interface to the OpenEdge REST Adapter. The endpoints will need to be accessible over HTTP(S) to the internet for Rollbase public cloud users or the Rollbase instance for private cloud users. The appropriate security settings are necessary to protect these endpoints. Using Basic Authentication over HTTPS is a recommended deployment option. Different security models can be used during development and deployment. For example, you could develop with anonymous access over HTTP and change the Service URI to HTTPS and add Basic-Authentication at a later date without re-importing the catalog of services.

Once the OpenEdge side is set up, you can import from an OpenEdge JSDO Catalog to create an external object or you can import from a JSDO Catalog to create a new Rollbase application. Either way, you need the following information before starting the integration:

- The file location of the JSDO Catalogs to import.
- The base Service URL where the JSDO OpenEdge Services are deployed.
- If using Basic Authentication, the username and password for these services

When using REST or SOAP APIs to create, update, or delete external objects linked to OpenEdge services, you need to supply both the object integration name as a string and an object ID. See the SOAP and REST APIs described in [Metadata API and XML Reference](#) on page 687, [Rollbase REST Methods](#) on page 732, and [Rollbase SOAP Methods](#) on page 779.

## Limitations

The following limitations apply when creating objects for OpenEdge Services:

- OpenEdge Service Objects are only certified with Progress OpenEdge 11.3.2 and 11.3.3.
- Only single-table data sets can map to a Rollbase object.
- Rollbase creates objects with fields that match OpenEdge resource columns. If a field has a name that cannot be used in Rollbase, such as a field that contains hyphens, it will be skipped.
- Arrays in OpenEdge are not imported.
- OpenEdge Service Objects do not support the following object attributes available in Rollbase:

- Document
- Organization
- Portal User
- Approval
- Survey
- Survey Taker
- Queue

For information about the object attributes that an OpenEdge Service Object supports, see [Enabling object attributes for an OpenEdge Service Object](#) on page 340.

- Relationships involving OpenEdge objects support only the following scenarios:
  - 1-1 or 1-N cardinality when establishing a relationship between an OpenEdge object and another OpenEdge object or a Rollbase Native object.
  - 1-1 or N-1 cardinality when establishing a relationship between a Rollbase Native object and an OpenEdge object.

## Supported Data Types

The following OpenEdge data types will be converted to Rollbase fields. You can change the field type in Rollbase when you import from the OpenEdge JSDO Catalog or at a later time using the Rollbase field **Convert** action.

CHARACTER	Text	This can be converted to many more specific Rollbase types
DATE	Date	string (ISO 8601 formatted string of the form "yyyy-mm-dd")
DATETIME	DateTime	string (ISO 8601 formatted string of the form "yyyy-mm-ddThh:mm:ss.sss")
DATETIME-TZ	DateTime	string (ISO 8601 formatted string of the form "yyyy-mm-ddThh:mm:ss.sss+hh:mm")
DECIMAL	Decimal	Can be converted to Currency or Percent
INT64 Integer		
INTEGER	Integer	
LOGICAL	Checkbox	

LONGCHAR	Text Area	Can be converted to other Text types
ROWID	Text	OE Special Type - typically read only - string (Base64 encoded)

## Linking a Rollbase Object to OpenEdge Data

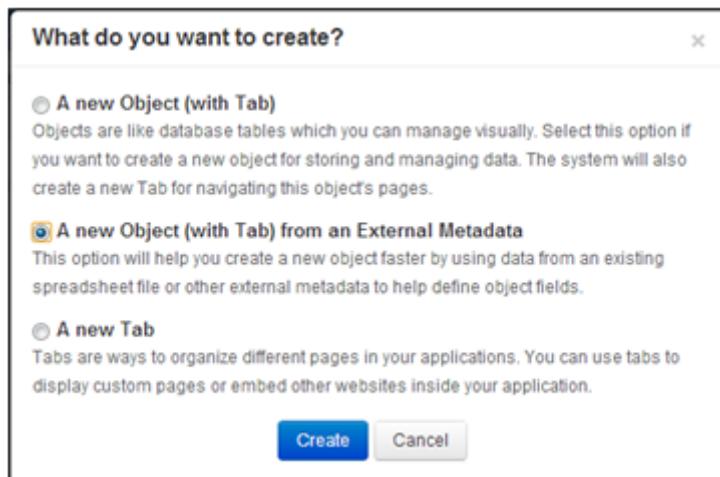
You can create Rollbase objects with data exposed by OpenEdge Services. You do this by specifying an OpenEdge JSDO Catalog file and an OpenEdge Resource URI, as described in [Creating Rollbase Objects from OpenEdge Services](#) on page 322.

After you create an external object, the data is linked. When the Rollbase application creates or modifies an object record, the data is also added to or modified in the OpenEdge database. If the data is modified in OpenEdge, the Rollbase record is updated when the page is refreshed.

To create an external object from an OpenEdge Service:

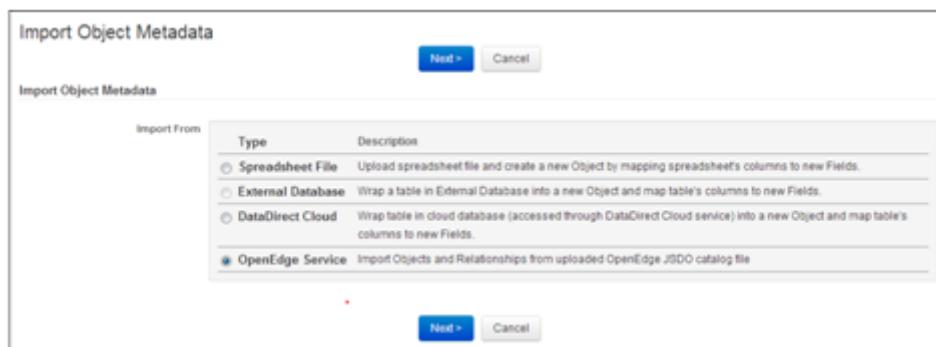
1. On the application's home page, click the add symbol (+).

The **What do you want to create?** dialog appears:



2. Select **A new Object (with Tab) from an External Metadata** and click **Create**.

The **Import Object Metadata** page appears:



3. Select **OpenEdge Service** and click **Next**.

The **Import Objects from OpenEdge Services** page appears.

4. Specify the following:

If you are using Rollbase public cloud:

- **OpenEdge JSDO Catalog** file that must define the object. Progress Rollbase recommends that you refer to the [Enabling support for filtering options and sorting](#) on page 328 to know about how a JSDO catalog file must be to fully utilize Rollbase functionalities.
- **Service URI** that contains the data to create instances of the new object.
- **No Authentication**: Specifies that no login credentials are required to access the service URI.
- **Basic Authentication**: Specifies that the login credentials you create be used to access the service URI. If you select this option, you must specify **Login Name** and **Password** of your choice that you want to use to access Rollbase.

If you are using Rollbase private cloud:

- **OpenEdge JSDO Catalog**: A JSDO file that defines the object. Progress Rollbase recommends that you refer to the [Enabling support for filtering options and sorting](#) on page 328 to know about how a JSDO catalog file must be to fully utilize Rollbase functionalities.
- **Service URI**: A URI that contains data, which creates instances of the new object.
- **No Authentication**: Specifies that no login credentials are required to access the service URI.
- **Use Current User (if OpenEdge Authentication is enabled)**: Specifies that Progress OpenEdge AppServer credentials are required to access the service URI. This is the default option for users accessing Rollbase using Progress OpenEdge AppServer credentials and this option is unavailable for non-OpenEdge users.

---

**Note:** If the OpenEdge service is configured with Single Point Authentication, Rollbase must also have OpenEdge Authentication configured for it, and as a pre-requisite, you (a Private Cloud user) must copy a set of JAR files from the OpenEdge library to the Rollbase library. For details on how to implement OpenEdge Authentication in Rollbase, see [Setting an Authentication Method](#) and [OpenEdge Authentication details](#) on page 384.

---

- **Basic Authentication**: Specifies that the login credentials you create be used to access the service URI. If you select this option, you must specify **Login Name** and **Password** of your choice that you want to use to access Rollbase.

5. Click **Next**. The components defined in the OpenEdge JSDO Catalog are displayed.

---

**Note:** Rollbase will open the Catalog and find appropriate REST services that can be mapped to Rollbase Objects. Only single-table data sets can map to a Rollbase Object. If a JSDO Catalog contains many resources, you can import all the available single-table resources at once, or select them one-by-one to import.

---

6. Select the components you want to use from the **Select Components** area, and then click **Next**.

The new object's details are displayed and the default values are inferred from the OpenEdge JSDO Catalog.

7. Accept or edit the default values, and for each object you create, specify the fields that should be used as the **Primary Key**. Then, click **Create** to create the object.

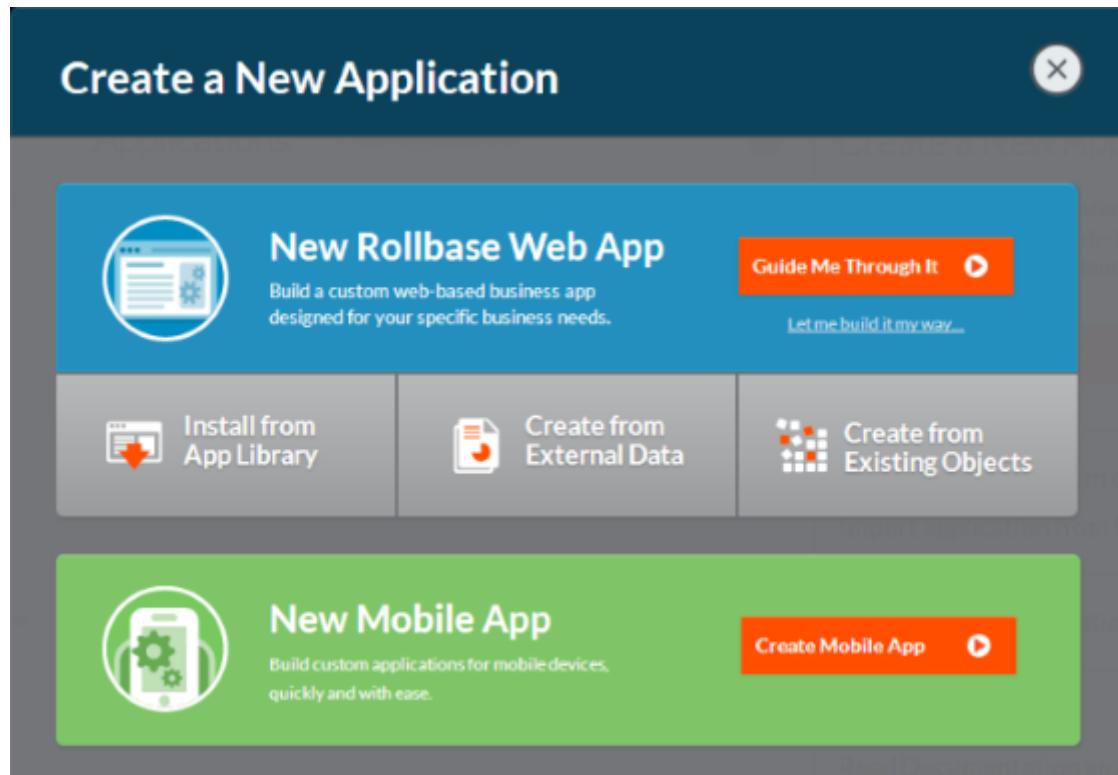
## Creating an Application from OpenEdge Data

You can also create the foundation data model for a Rollbase application (multiple objects) with data from OpenEdge Services. You do this by specifying an OpenEdge JSDO Catalog file and an OpenEdge Resource URI when you create the application.

To create a Rollbase application from an OpenEdge Service:

1. On the Rollbase sidebar, click **New Application**.

The **Create a New Application** dialog appears:



2. Select **Create from External Metadata**.

The **Import Application Metadata** page appears.

3. Select **OpenEdge Service** and click **Next**.

The **Import an Application from OpenEdge Services** page appears.

4. Specify the following:

If you are using Rollbase public cloud:

- **OpenEdge JSDO Catalog** file that must define the object.

Progress Rollbase recommends that you refer to the [Enabling support for filtering options and sorting](#) on page 328 and [Enabling object attributes for an OpenEdge Service Object](#) on page 340 to know about how a JSDO catalog file must be to fully utilize all the available Rollbase functionality for an OpenEdge Service Object.

- **Service URI** that contains the data to create instances of the new object.
- **No Authentication**: Specifies that no login credentials are required to access the service URI.
- **Basic Authentication**: Specifies that the login credentials you create be used to access the service URI. If you select this option, you must specify **Login Name** and **Password** of your choice that you want to use to access Rollbase.

If you are using Rollbase private cloud:

- **OpenEdge JSDO Catalog**: A JSDO file that defines the object.

Progress Rollbase recommends that you refer to [Enabling support for filtering options and sorting](#) on page 328 and [Enabling object attributes for an OpenEdge Service Object](#) on page 340 to know about how a JSDO catalog file must be to fully utilize all the available Rollbase functionality for an OpenEdge Service Object.

- **Service URI**: A URI that contains data, which creates instances of the new object.
- **No Authentication**: Specifies that no login credentials are required to access the service URI.
- **Use Current User (if OpenEdge Authentication is enabled)**: Specifies that Progress OpenEdge AppServer credentials are required to access the service URI. This is the default option for users accessing Rollbase using Progress OpenEdge AppServer credentials and this option is unavailable for non-OpenEdge users.

---

**Note:** If the OpenEdge service is configured with Single Point Authentication, Rollbase must also have OpenEdge Authentication configured for it, and as a pre-requisite, you (a Private Cloud user) must copy a set of JAR files from the OpenEdge library to the Rollbase library. For details on how to implement OpenEdge Authentication in Rollbase, see [Setting an Authentication Method](#) and [OpenEdge Authentication details](#) on page 384.

---

- **Basic Authentication**: Specifies that the login credentials you create be used to access the service URI. If you select this option, you must specify **Login Name** and **Password** of your choice that you want to use to access Rollbase.
5. Click **Next**. The **Import an Application from OpenEdge Services** page is updated to show the components defined in the OpenEdge JSDO Catalog.
  6. Select the objects you want to import and click **Next**.

The **Import an Application from OpenEdge Services** page is updated to show the application's details. The default values for the application and object names are inferred from the OpenEdge JSDO Catalog.

7. Select **Do you want to create a new Tab for the object now** to add a tab for your object and **Service object support complex filter and sorting** to enable sorting and filtering of fields when viewing application data. Accept the default values or modify them as per your requirements.

**Note:** For each object you create, you must have only one field specified as the **Record Name** field, and for each object that you create, specify the fields that should be used as the **Primary Key**. Neglecting this step will mean that some Rollbase capabilities that rely on finding triggers or related objects cannot be used. Moreover,

---

8. Click **Create**.

The application is created and added to the list of existing application.

## Enabling support for filtering options and sorting

In Progress OpenEdge, you create a Business Entity class and generate from it your JSDO catalog file. You use this JSDO catalog file to create a Rollbase object. You can enable support for filtering options and sorting by enhancing the Business Entity to accept a JSON object (JSON filter) as a filter parameter.

This section describes what you must include in your Business Entity class to enable sorting and filtering options in Rollbase.

To support sorting and filtering, you must ensure that your Business Entity and the associated include file ( .i) with the schema has support for the parameters that are contained in the JSON object:

- `ablfilter`: To support filtering of records.
- `id`: To support search by ID.
- `top` and `skip`: To support paging.
- `orderby`: To support sorting and grouping of records.

In the include file, the temp-table definitions include the fields, `id` and sequence (`seq`) to support JSON filter:

- The `id` field is used to support the `id` parameter in the JSON filter.
- The `seq` field is used to support the `orderby` parameter in the JSON filter.

The `Read` method contains the code to process the JSON filter and return the corresponding records. The code uses the JSON classes in ABL to process the JSON filter. The ABL code implements the logic to handle each parameter. An `AFTER-ROW-FILL` method callback is used to populate the values of `id` and `seq` fields. A local variable `iSeq` is used to obtain the values for the `seq` field.

---

**Note:** When you use the following code samples, verify that the formatting of the code is maintained. Alternatively, you can download the code samples from a [Progress Community page](#).

---

The following sample include file (`customer.i`) illustrates a temp-table definition with id and sequence fields:

```
/*-----  
  File      : customer.i  
  Purpose   :  
  Syntax    :  
  Description :  
  Author(s)  :  
  Created    :  
  Notes     :  
-----*/  
  
/* ***** Definitions ***** */  
  
/* ***** Preprocessor Definitions ***** */  
  
/* ***** Main Block ***** */  
  
/** Dynamically generated schema file **/  
  
@openapi.openedge.entity.primarykey (fields="CustNum") .  
DEFINE TEMP-TABLE ttCustomer BEFORE-TABLE bttCustomer  
FIELD id      AS CHARACTER  
FIELD seq     AS INTEGER  
FIELD CustNum AS INTEGER      INITIAL "0" LABEL "Cust Num"  
FIELD Name    AS CHARACTER    LABEL "Name"  
FIELD Address AS CHARACTER    LABEL "Address"  
FIELD Address2 AS CHARACTER   LABEL "Address2"  
FIELD Balance AS DECIMAL     INITIAL "0" LABEL "Balance"  
FIELD City    AS CHARACTER    LABEL "City"  
FIELD Comments AS CHARACTER   LABEL "Comments"  
FIELD Contact AS CHARACTER   LABEL "Contact"  
FIELD Country AS CHARACTER   INITIAL "USA" LABEL "Country"  
FIELD CreditLimit AS DECIMAL  INITIAL "1500" LABEL "Credit Limit"  
FIELD Discount AS INTEGER    INITIAL "0" LABEL "Discount"  
FIELD EmailAddress AS CHARACTER LABEL "Email"  
FIELD Fax     AS CHARACTER   LABEL "Fax"  
FIELD Phone   AS CHARACTER   LABEL "Phone"  
FIELD PostalCode AS CHARACTER LABEL "Postal Code"  
FIELD SalesRep AS CHARACTER  LABEL "Sales Rep"  
FIELD State   AS CHARACTER  LABEL "State"  
FIELD Terms   AS CHARACTER  INITIAL "Net30" LABEL "Terms"  
INDEX seq seq  
  
.  
  
DEFINE DATASET dsCustomer FOR ttCustomer.
```

In OpenEdge 11.3 Business Entity the code to process the JSON filter is called from the Read method and replaces the generated code for the Read method as well as the private `applyFillMethod()`.

The following Business Entity code from OpenEdge 11.3 illustrates how to process the JSON filter and enable filtering and sorting:

```

/*
  File      : Customer.cls
  Syntax   :
  Author(s) :
  Created  :
  Notes    :
-----*/
@program FILE(name="Customer.cls", module="AppServer").
@openapi.openedge.export FILE(type="REST", executionMode="singleton",
useReturnValue="false", writeDataSetBeforeImage="false").
@progress.service.resource FILE(name="Customer", URI="/Customer",
schemaName="dsCustomer", schemaFile="" .

USING Progress.Lang.*.
USING Progress.Json.ObjectModel.*.

BLOCK-LEVEL ON ERROR UNDO, THROW.

CLASS Customer:

/*
  Purpose:
  Notes:
-----*/
{ "customer.i" }

DEFINE DATA-SOURCE srcCustomer FOR Customer.
DEFINE VARIABLE iSeq          AS INTEGER      NO-UNDO.

/*
  Purpose: Get one or more records, based on a filter string
  Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="false").
@progress.service.resourceMapping(type="REST", operation="read",
URI="?filter=~{filter~}", alias="", mediaType="application/json").
METHOD PUBLIC VOID ReadCustomer(
  INPUT filter AS CHARACTER,
  OUTPUT DATASET dsCustomer):

  DEFINE VARIABLE jsonParser      AS ObjectModelParser      NO-UNDO.
  DEFINE VARIABLE jsonObject     AS JsonObject      NO-UNDO.
  DEFINE VARIABLE cWhere        AS CHARACTER      NO-UNDO.

  DEFINE VARIABLE hQuery        AS HANDLE      NO-UNDO.
  DEFINE VARIABLE lUseReposition AS LOGICAL      NO-UNDO.

  DEFINE VARIABLE iCount        AS INTEGER      NO-UNDO.

```

```
DEFINE VARIABLE ablFilter          AS CHARACTER          NO-UNDO.
DEFINE VARIABLE id                AS CHARACTER  INITIAL ?  NO-UNDO.
DEFINE VARIABLE iMaxRows          AS INTEGER    INITIAL ?  NO-UNDO.
DEFINE VARIABLE iSkipRows          AS INTEGER    INITIAL ?  NO-UNDO.

DEFINE VARIABLE cOrderBy          AS CHARACTER  INITIAL ""  NO-UNDO.

/* The filter parameter can be:
   - a WHERE clause: if it begins with WHERE
   - a JSON object representing a query if it begins with {
   - a free form filter specific to the business entity
*/
MESSAGE "DEBUG: " filter.

/* get rid of any existing data */
EMPTY TEMP-TABLE ttCustomer.
iSeq = 0.

IF filter BEGINS "WHERE " THEN
  cWhere = filter.
ELSE IF filter BEGINS "~{" THEN
DO:
  jsonParser = NEW ObjectModelParser().
  jsonObject = CAST(jsonParser:Parse(filter), jsonObject).
  iMaxRows = jsonObject:GetInteger("top")  NO-ERROR.

  iSkipRows = jsonObject:GetInteger("skip") NO-ERROR.
  ablFilter = jsonObject:GetCharacter("ablFilter") NO-ERROR.
  id = jsonObject:GetCharacter("id") NO-ERROR.
  cOrderBy = jsonObject:GetCharacter("orderBy") NO-ERROR.

  cWhere = "WHERE " + ablFilter.

  IF cOrderBy > "" THEN
DO:
  cOrderBy = REPLACE(cOrderBy, ",", " by ").
  cOrderBy = "by " + cOrderBy + " ".
  /* NOTE: id and seq fields should be removed from cWhere and
cOrderBy */
  cOrderBy = REPLACE(cOrderBy, "by id desc", "").
  cOrderBy = REPLACE(cOrderBy, "by id ", "").
  cOrderBy = REPLACE(cOrderBy, "by seq desc", "").
  cOrderBy = REPLACE(cOrderBy, "by seq ", "").
END.

  lUseReposition = iSkipRows <> ?.

END.
ELSE IF filter NE "" THEN
DO:
  /* Use filter as WHERE clause */
  cWhere = "WHERE " + filter.
END.

IF iMaxRows <> ? AND iMaxRows > 0 THEN
DO:
  BUFFER ttCustomer:HANDLE:BATCH-SIZE = iMaxRows.
END.
ELSE DO:
  IF id > "" THEN
    BUFFER ttCustomer:HANDLE:BATCH-SIZE = 1.
  ELSE
    BUFFER ttCustomer:HANDLE:BATCH-SIZE = 0.
END.
BUFFER ttCustomer:ATTACH-DATA-SOURCE (DATA-SOURCE srcCustomer:HANDLE).
```

```

        IF cOrderBy = ? THEN cOrderBy = "".
        cWhere = IF cWhere > "" THEN (cWhere + " " + cOrderBy) ELSE ("WHERE "
+ cOrderBy).
        MESSAGE "DEBUG: cWhere: " cWhere.
        MESSAGE "DEBUG: cOrderBy: " cOrderBy.
        DATA-SOURCE srcCustomer:FILL-WHERE-STRING = cWhere.

        IF lUseReposition THEN
        DO:
            hQuery = DATA-SOURCE srcCustomer:QUERY.
            hQuery:QUERY-OPEN.
            IF id > "" AND id <> "?" THEN
            DO:
                hQuery:REPOSITION-TO-ROWID(TO-ROWID(id)).
            END.
            ELSE IF iSkipRows <> ? AND iSkipRows > 0 THEN
            DO:
                hQuery:REPOSITION-TO-ROW(iSkipRows).
                IF NOT AVAILABLE Customer THEN
                    hQuery:GET-NEXT () NO-ERROR.
                END.
                iCount = 0.
                REPEAT WHILE NOT hQuery:QUERY-OFF-END AND iCount < iMaxRows:
                    hQuery:GET-NEXT () NO-ERROR.
                    IF AVAILABLE Customer THEN
                    DO:
                        CREATE ttCustomer.
                        BUFFER-COPY Customer TO ttCustomer.
                        ASSIGN ttCustomer.id = STRING(ROWID(Customer))
                        iSeq = iSeq + 1
                        ttCustomer.seq = iSeq.
                    END.
                    iCount = iCount + 1.
                END.
            END.
            ELSE DO:
                IF id > "" THEN DATA-SOURCE srcCustomer:RESTART-ROWID(1) = TO-ROWID
((id)).
                BUFFER ttCustomer:SET-CALLBACK ("AFTER-ROW-FILL", "AddIdField").

                DATASET dsCustomer:FILL().
            END.
            FINALLY:
                BUFFER ttCustomer:DETACH-DATA-SOURCE().
            END FINALLY.

        END METHOD.

        METHOD PUBLIC VOID AddIdField (INPUT DATASET dsCustomer):
        ASSIGN ttCustomer.id = STRING(ROWID(Customer))
        iSeq = iSeq + 1
        ttCustomer.seq = iSeq.
        END.

/*
-----*
Purpose: Create one or more new records
Notes:
-----*/
        @openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="false").
        @progress.service.resourceMapping(type="REST", operation="create", URI="",
alias="", mediaType="application/json").

```

```
METHOD PUBLIC VOID CreateCustomer(INPUT-OUTPUT DATASET dsCustomer):  
  
    THIS-OBJECT:CommitCustomer(INPUT "", INPUT ROW-CREATED).  
    RETURN.  
END METHOD.  
  
/*-----  
Purpose:  Update one or more records  
Notes:  
-----*/  
  
@openapi.openedge.export(type="REST", useReturnValue="false",  
writeDataSetBeforeImage="false").  
@progress.service.resourceMapping(type="REST", operation="update", URI="",  
alias="", mediaType="application/json").  
METHOD PUBLIC VOID UpdateCustomer(INPUT-OUTPUT DATASET dsCustomer):  
  
    THIS-OBJECT:CommitCustomer(INPUT "", INPUT ROW-MODIFIED).  
END METHOD.  
  
/*-----  
Purpose:  Delete a record  
Notes:  
-----*/  
  
@openapi.openedge.export(type="REST", useReturnValue="false",  
writeDataSetBeforeImage="false").  
@progress.service.resourceMapping(type="REST", operation="delete", URI="",  
alias="", mediaType="application/json").  
METHOD PUBLIC VOID DeleteCustomer(INPUT-OUTPUT DATASET dsCustomer):  
  
    THIS-OBJECT:CommitCustomer(INPUT "", INPUT ROW-DELETED).  
END METHOD.  
  
/*-----  
Purpose:  generic routine for creating/updating/deleting customers  
Notes:  
-----*/  
  
METHOD PRIVATE VOID commitCustomer(  
    INPUT pcFieldMapping AS CHARACTER,  
    INPUT piRowState AS INTEGER ):  
    DEFINE VARIABLE Skip-list AS CHAR NO-UNDO.  
    BUFFER ttCustomer:ATTACH-DATA-SOURCE (DATA-SOURCE srcCustomer:HANDLE,  
                                            pcFieldMapping).  
    FOR EACH ttCustomer.  
        BUFFER ttCustomer:MARK-ROW-STATE (piRowState).  
    END.

---


```

```
IF piRowState = ROW-CREATED THEN
    Skip-list = "CustNum".
FOR EACH bttCustomer:
    BUFFER bttCustomer:SAVE-ROW-CHANGES(1, Skip-list).
END.
FINALLY:
    BUFFER ttCustomer:DETACH-DATA-SOURCE().
    RETURN.
END FINALLY.
END METHOD.

END CLASS.
```

In OpenEdge 11.4 Business Entity, you can enable filtering and sorting by replacing the code in the `Read` method with the code that processes the JSON filter in the same way as described earlier for OpenEdge 11.3 Business Entity.

In Progress Developer Studio for OpenEdge 11.4, you can use an annotation to indicate that the `Read` method has support for the JSON filter:

```
@openapi.openedge.method.property (name="capabilities",
value="top,skip,id,orderBy").
```

---

**Note:** The "ablFilter" parameter is considered to always be supported and does not need to be listed in the values for the "capabilities" property.

---

The following is a Business Entity class file from OpenEdge 11.4 that contains the code to process the JSON filter:

```

/*
  File      : Customer.cls
  Syntax   :
  Author(s) :
  Created  :
  Notes    :
-----*/
@program FILE(name="Customer.cls", module="AppServer").
@openapi.openedge.export FILE(type="REST", executionMode="singleton",
useReturnValue="false", writeDataSetBeforeImage="false").
@progress.service.resource FILE(name="Customer", URI="/Customer",
schemaName="dsCustomer", schemaFile="Test/AppServer/customer.i").

USING OpenEdge.BusinessLogic.BusinessEntity.
USING Progress.Lang.*.
USING Progress.Json.ObjectModel.*.

BLOCK-LEVEL ON ERROR UNDO, THROW.

CLASS Customer INHERITS BusinessEntity:

/*
-----*
  Purpose:
  Notes:
-----*/
{ "customer.i" }

DEFINE DATA-SOURCE srcCustomer FOR sports2000.Customer.
DEFINE VARIABLE iSeq           AS INTEGER      NO-UNDO.

/*
-----*
  Purpose:
  Notes:
-----*/
CONSTRUCTOR PUBLIC Customer():

  DEFINE VAR hDataSourceArray AS HANDLE NO-UNDO EXTENT 1.
  DEFINE VAR cSkipListArray AS CHAR NO-UNDO EXTENT 1.

  SUPER (DATASET dsCustomer:HANDLE).

  /* Data Source for each table in dataset. Should be in table order as
defined
     in DataSet */

  hDataSourceArray[1] = DATA-SOURCE srcCustomer:HANDLE.

  /* Skip-list entry for each table in dataset. Should be in temp-table
order
-----*/

```

```

        as defined in DataSet */
/* Each skip-list entry is a comma-separated list of field names, to
be
        ignored in create stmt */

cSkipListArray[1] = "CustNum".

THIS-OBJECT:ProDataSource = hDataSourceArray.
THIS-OBJECT:SkipList = cSkipListArray.

END CONSTRUCTOR.

/*
-----*
Purpose: Get one or more records, based on a filter string
Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="read",
URI="?filter=~{filter~}", alias="", mediaType="application/json").
@openapi.openedge.method.property (name="capabilities",
value="top,skip,id,orderBy").
METHOD PUBLIC VOID ReadCustomer(
    INPUT filter AS CHARACTER,
    OUTPUT DATASET dsCustomer):

DEFINE VARIABLE jsonParser      AS ObjectModelParser      NO-UNDO.
DEFINE VARIABLE jsonObject     AS JsonObject          NO-UNDO.
DEFINE VARIABLE cWhere         AS CHARACTER           NO-UNDO.

DEFINE VARIABLE hQuery          AS HANDLE              NO-UNDO.
DEFINE VARIABLE lUseReposition  AS LOGICAL             NO-UNDO.

DEFINE VARIABLE iCount          AS INTEGER             NO-UNDO.

DEFINE VARIABLE ablFilter       AS CHARACTER           NO-UNDO.
DEFINE VARIABLE id              AS CHARACTER INITIAL ? NO-UNDO.
DEFINE VARIABLE iMaxRows        AS INTEGER  INITIAL ? NO-UNDO.
DEFINE VARIABLE iSkipRows       AS INTEGER  INITIAL ? NO-UNDO.

DEFINE VARIABLE cOrderBy        AS CHARACTER INITIAL "" NO-UNDO.

/* The filter parameter can be:
   - a WHERE clause: if it begins with WHERE
   - a JSON object representing a query if it begins with {
   - a free form filter specific to the business entity
*/
MESSAGE "DEBUG: " filter.

/* get rid of any existing data */
EMPTY TEMP-TABLE ttCustomer.
iSeq = 0.

IF filter BEGINS "WHERE " THEN
    cWhere = filter.
ELSE IF filter BEGINS "~{" THEN
DO:

```

```

        jsonParser = NEW ObjectModelParser().
        jsonObject = CAST(jsonParser:Parse(filter), jsonObject).
        iMaxRows = jsonObject:GetIntger("top")  NO-ERROR.

        iSkipRows = jsonObject:GetIntger("skip") NO-ERROR.
        ablFilter = jsonObject:GetCharacter("ablFilter") NO-ERROR.
        id = jsonObject:GetCharacter("id") NO-ERROR.
        cOrderBy = jsonObject:GetCharacter("orderBy") NO-ERROR.

        cWhere = "WHERE " + ablFilter.

        IF cOrderBy > "" THEN
        DO:
            cOrderBy = REPLACE(cOrderBy, ",", " by ").
            cOrderBy = "by " + cOrderBy + " ".
            /* NOTE: id and seq fields should be removed from cWhere and
            cOrderBy */
            cOrderBy = REPLACE(cOrderBy, "by id desc", "") .
            cOrderBy = REPLACE(cOrderBy, "by id ", "") .
            cOrderBy = REPLACE(cOrderBy, "by seq desc", "") .
            cOrderBy = REPLACE(cOrderBy, "by seq ", "") .
        END.

        lUseReposition = iSkipRows <> ?.

        END.
        ELSE IF filter NE "" THEN
        DO:
            /* Use filter as WHERE clause */
            cWhere = "WHERE " + filter.
        END.

        IF iMaxRows <> ? AND iMaxRows > 0 THEN
        DO:
            BUFFER ttCustomer:HANDLE:BATCH-SIZE = iMaxRows.
        END.
        ELSE DO:
        IF id > "" THEN
            BUFFER ttCustomer:HANDLE:BATCH-SIZE = 1.
        ELSE
            BUFFER ttCustomer:HANDLE:BATCH-SIZE = 0.
        END.
        BUFFER ttCustomer:ATTACH-DATA-SOURCE (DATA-SOURCE srcCustomer:HANDLE) .

        IF cOrderBy = ? THEN cOrderBy = "".
        cWhere = IF cWhere > "" THEN (cWhere + " " + cOrderBy) ELSE ("WHERE "
        + cOrderBy).
        MESSAGE "DEBUG: cWhere: " cWhere.
        MESSAGE "DEBUG: cOrderBy: " cOrderBy.
        DATA-SOURCE srcCustomer:FILL-WHERE-STRING = cWhere.

        IF lUseReposition THEN
        DO:
            hQuery = DATA-SOURCE srcCustomer:QUERY.
            hQuery:QUERY-OPEN.
            IF id > "" AND id <> "?" THEN
            DO:
                hQuery:REPOSITION-TO-ROWID(TO-ROWID(id)) .
            END.
            ELSE IF iSkipRows <> ? AND iSkipRows > 0 THEN
            DO:
                hQuery:REPOSITION-TO-ROW(iSkipRows) .
                IF NOT AVAILABLE Customer THEN
                    hQuery:GET-NEXT () NO-ERROR.
            END.
            iCount = 0.
            REPEAT WHILE NOT hQuery:QUERY-OFF-END AND iCount < iMaxRows:
                hQuery:GET-NEXT () NO-ERROR.

```

```

        IF AVAILABLE Customer THEN
        DO:
            CREATE ttCustomer.
            BUFFER-COPY Customer TO ttCustomer.
            ASSIGN ttCustomer.id = STRING(ROWID(Customer))
                iSeq = iSeq + 1
                ttCustomer.seq = iSeq.
            END.
            iCount = iCount + 1.
        END.
    END.
    ELSE DO:
        IF id > "" THEN DATA-SOURCE srcCustomer:RESTART-ROWID(1) = TO-ROWID
        ((id)).
            BUFFER ttCustomer:SET-CALLBACK ("AFTER-ROW-FILL", "AddIdField").

            DATASET dsCustomer:FILL().
        END.
    FINALLY:
        BUFFER ttCustomer:DETACH-DATA-SOURCE().
    END FINALLY.

    END METHOD.

METHOD PUBLIC VOID AddIdField (INPUT DATASET dsCustomer):
    ASSIGN ttCustomer.id = STRING(ROWID(Customer))
        iSeq = iSeq + 1
        ttCustomer.seq = iSeq.
END.

/*
-----*
Purpose: Create one or more new records
Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="create", URI="",
alias="", mediaType="application/json").
METHOD PUBLIC VOID CreateCustomer(INPUT-OUTPUT DATASET dsCustomer):

    DEFINE VAR hDataSet AS HANDLE NO-UNDO.
    hDataSet = DATASET dsCustomer:HANDLE.

    SUPER:CreateData(DATASET-HANDLE hDataSet BY-REFERENCE).
END METHOD.

/*
-----*
Purpose: Update one or more records
Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="update", URI="",
alias="", mediaType="application/json").
METHOD PUBLIC VOID UpdateCustomer(INPUT-OUTPUT DATASET dsCustomer):

```

```

    DEFINE VAR hDataSet AS HANDLE NO-UNDO.
    hDataSet = DATASET dsCustomer:HANDLE.

    SUPER:UpdateData(DATASET-HANDLE hDataSet BY-REFERENCE) .
    END METHOD.

/*
-----*
Purpose:      Delete a record
Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="delete", URI="",
alias="", mediaType="application/json").
METHOD PUBLIC VOID DeleteCustomer(INPUT-OUTPUT DATASET dsCustomer) :

    DEFINE VAR hDataSet AS HANDLE NO-UNDO.
    hDataSet = DATASET dsCustomer:HANDLE.

    SUPER:DeleteData(DATASET-HANDLE hDataSet BY-REFERENCE) .
    END METHOD.

/*
-----*
Purpose:      Submit a record
Notes:
-----*/
@openapi.openedge.export(type="REST", useReturnValue="false",
writeDataSetBeforeImage="true").
@progress.service.resourceMapping(type="REST", operation="submit",
URI="/SubmitCustomer", alias="", mediaType="application/json").
METHOD PUBLIC VOID SubmitCustomer(INPUT-OUTPUT DATASET dsCustomer) :

    /* Calling extending class's CUD methods instead of SUPER:Submit() in
    case
        customized functionality was added.
        Do deletes first, next modifies, and finally creates */
    THIS-OBJECT:DeleteCustomer(DATASET dsCustomer).
    THIS-OBJECT:UpdateCustomer(DATASET dsCustomer).
    THIS-OBJECT:CreateCustomer(DATASET dsCustomer).
    END METHOD.

END CLASS.

```

For information about sorting and grouping, see [Filtering by Search criteria](#) on page 243.

## Enabling object attributes for an OpenEdge Service Object

In Progress OpenEdge, you create a Business Entity class and generate from it your JSDO catalog file. You use the JSDO catalog file to create an OpenEdge Service object in Rollbase. For your OpenEdge Service object to support object attributes, you must code your OpenEdge Business Entity class in a way that it includes the fields required to enable the object attributes. For information about object attributes, see [Object Attributes](#) on page 84.

To enable object attributes for an OpenEdge Service Object:

1. Decide which Object attributes you must enable for your OpenEdge Service Object.
2. In Progress OpenEdge, build a JSDO catalog file that consists of the required fields for all the object attributes. This is done using the information provided in the below table.
3. In Progress Rollbase, using your JSDO catalog file, create an OpenEdge Services object (see [Creating an Application from OpenEdge Data](#) on page 326).
4. Edit the OpenEdge Service Object definition to include the required object attributes. This is done by selecting the object attribute and mapping the available fields required to set the object attribute (see [Viewing and Editing an Object Definition](#) on page 121).

The following table lists all the object attributes supported in an OpenEdge Service Object and the corresponding fields required in the JSDO catalog file:

Object attribute	Fields to be included in your JSDO Catalog file	Data Type of the Field
Workflow	Process	CHARACTER
	Status	CHARACTER
Contact	First Name	CHARACTER
	Middle Name	CHARACTER
	Last Name	CHARACTER
	Email	CHARACTER
	Fax	CHARACTER
	Phone	CHARACTER
Location	Street Address-1	CHARACTER
	Street Address-2	CHARACTER
	City	CHARACTER
	State/Province	CHARACTER
	Country	CHARACTER
Event	Event Subject	CHARACTER
	Duration	INT-64
	Date/Time	DATETIME-TZ
	Private	LOGICAL
	Description	CHARACTER
	Location	CHARACTER
	Pop-up Reminder	CHARACTER
	Reminded	LOGICAL
Task	Task Subject	CHARACTER
	Due Date	DATETIME-TZ
	Priority	INTEGER
	Private	LOGICAL
	Description	CHARACTER
Multi-Currency	Currency Code	CHARACTER
	Rate Date	DATE
Lockable	Is Locked	LOGICAL

You can selectively include the fields in your JSDO catalog file based on which object attributes you need in your OpenEdge Service Object. The field names can be of your choice but you must ensure that the number of fields (and it's data type) required for enabling an object attribute is as per the information provided in the above table. For example, to enable the **Multi-Currency** object attribute, you must have two available fields in your JSDO catalog file (one to store Currency Code and another to store Rate Date).

## An example explaining the entire process of enabling object attributes

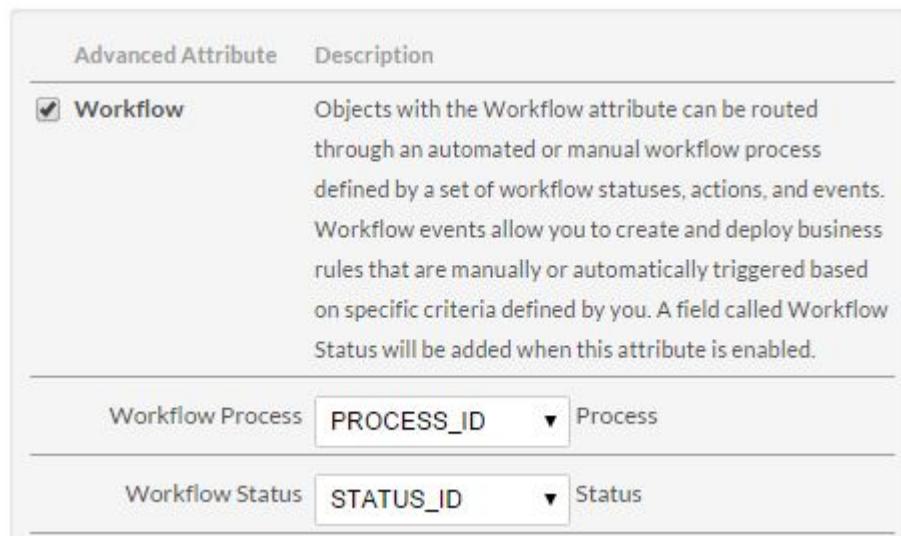
To enable **Workflow** attribute in your OpenEdge Service object, add two temp-table fields, **PROCESS\_ID** and **STATUS\_ID**, of CHARACTER data type in your OpenEdge business entity class. The following is the code-snippet from a JSDO catalog file (.json) that includes the two fields:

```

"version": "1.2",
"lastModified": "Thu Oct 30 03:03:24 PDT 2014",
"services": [
    {
        "name": "WorkflowobjService",
        "address": "\/rest\/WorkflowobjService",
        .
        .
        .
    },
    {
        "name": "WorkflowobjService",
        "address": "\/rest\/WorkflowobjService",
        .
        .
        .
    }
],
"PROCESS_ID": {
    "type": "string",
    "ablType": "CHARACTER",
    "default": "",
    "title": "PROCESSID"
},
"STATUS_ID": {
    "type": "string",
    "ablType": "CHARACTER",
    "default": "",
    "title": "STATUSID"
},
.
.
.

```

After creating the JSDO catalog file in your OpenEdge environment, you must use it to create an OpenEdge Service Object in Rollbase (see [Creating an Application from OpenEdge Data](#) on page 326). And then, edit your Object definition (see [Viewing and Editing an Object Definition](#) on page 121) to add the **Workflow** attribute and map the **PROCESS\_ID** and **STATUS\_ID** fields to **Workflow Process** and **Workflow Status**:



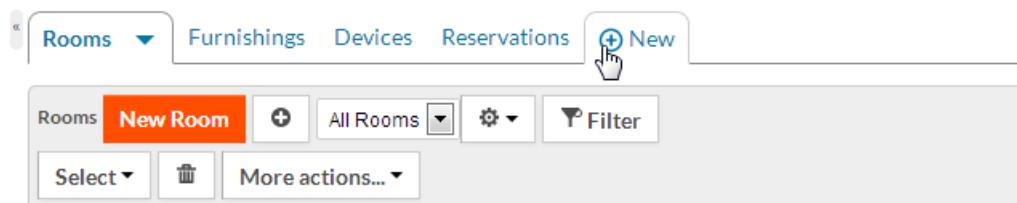
# Using DataDirect Cloud to Access External Data

DataDirect Cloud simplifies access to cloud data sources such as applications hosted on cloud platforms like Salesforce.com and Eloqua. You can use external objects together with DataDirect Cloud to easily bring external data into your Rollbase application. See the video posted at: [http://documentation.progress.com/output/ua/redir/RB\\_using\\_DDC\\_ext\\_data.html](http://documentation.progress.com/output/ua/redir/RB_using_DDC_ext_data.html) for a quick overview.

Each external object maps to a table in the external data source. Before you can follow these procedures, you must have defined a data source that connects to the required data using DataDirect Cloud.

To create an external object that uses DataDirect Cloud to access data in an external source:

1. Navigate to the target application.
2. Click the **New** tab that displays to the right of the existing object tabs:



The **What do you want to create?** dialog displays.

**What do you want to create?**

**A new Object (with Tab)**  
Objects are like database tables which you can manage visually. Select this option if you want to create a new object for storing and managing data. The system will also create a new Tab for navigating this object's pages.

**A new Object (with Tab) from an External Metadata**  
This option will help you create a new object faster by using data from an existing spreadsheet file or other external metadata to help define object fields.

**A new Tab**  
Tabs are ways to organize different pages in your applications. You can use tabs to display custom pages or embed other websites inside your application.

**Create** **Cancel**

3. Select **A new Object (with tab) from an External Metadata**.
4. Click **Create**.

The **Import Object Metadata** page displays.

5. For **Import Object Metadata** select **DataDirect Cloud**. Click **Create**.

The **Import from DataDirect Cloud** page displays.

6. On the **Import from DataDirect Cloud** page, specify the credentials for your DataDirect Cloud account and the name of the data source that contains the object to which the new external object will map. The data source must already be defined in DataDirect Cloud
7. Optionally, if you want to supply credentials for the data source each time the Rollbase application accesses the table, check the box to prompt for user credentials.
8. Click **Next**.

The **Create Object** page displays.

The screenshot shows the 'Object from ForceDD Database: Create Object' page. At the top, there is a tip: 'Tip: You can wrap selected table in ForceDD database and (to a large degree) manage them the same way as Rollbase object. To use this feature you need to have a clear understanding of your legacy database structure and good SQL knowledge. [Learn more](#)'.

**Select Database Table:** A dropdown menu labeled 'Select Table' with the placeholder 'Please select...'. Below it, a note: 'Using this tool you can enable Rollbase to access Cloud ForceDD database the same way as any other object. Select table in Cloud database. Later you will map database columns into fields of newly created object.'

**Object Properties:**

- Singular Name:** Account (Example: Project)
- Plural Name:** ACCOUNTS (Example: Projects)
- Record Name:** Account (Example: Project)
- Integration Name:** account (Example: Project)

**Description:** (This field is empty in the screenshot)

9. From the **Select Table** drop-down, choose one of the available tables.
10. Enter values for the object name and set the desired permissions.
11. Click **Create Object**.

The **Create Fields** page displays.

The screenshot shows the 'Object from ForceDD Database: Create Fields' page. At the top, there is a note: 'Select Column(s) for Primary Key'.

**Available:** A list of database columns: OWNERSHIP, PARENTID, PHONE, RATING, SHIPPINGCITY, SHIPPINGCOUNTRY, SHIPPINGPOSTALCODE, SHIPPINGSTATE, SHIPPINGSTREET, SIC, SICDESC, SITE.

**Primary Key:** A list of columns: ROWID.

**New Object "Account" from Table "ACCOUNT"**

Database Columns	What type of field should be created?	What should the label of this field be?	Integration Name
ACCOUNTNUMBER (40)	Text	Account Number	accountnumber
ACCOUNTSOURCE (40)	Text	Account Source	accountsource
ACTIVE (255)	Text	Active	active_
ANNUALREVENUE	Decimal	Annual Revenue	annualrevenue
BILLINGCITY (40)	Text	Billingcity	billingcity
BILLINGCOUNTRY (80)	Text	Billingcountry	billingcountry
BILLINGPOSTALCODE (20)	Text	Billingpostalcode	billingpostalcode
BILLINGSTATE (80)	Text	Billingstate	billingstate

12. Select the columns to use for creating the object's primary key and the options to create the object's fields.

13. Click **Create Fields**.

14. Select the appropriate Rollbase type and if desired, edit the label and integration names.

15. Click **Create Fields**.

The **Adjust SQL** page displays

16. Click **Preview Query** to see what the external object will look like.

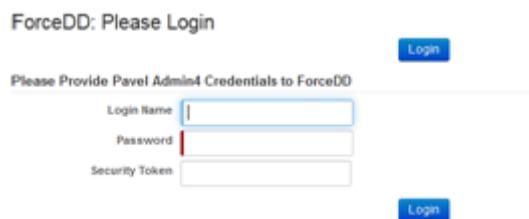
17. If the preview is not as expected, edit the SQL query and try again.

18. When you are satisfied with the results, click **Save**.

Rollbase creates an object similar to an external database object, but with the following limitations

- Related application features are disabled
- Only SELECT queries will be sent
- Relationships involving external objects support only the following scenarios:
  - 1-1 or 1-N cardinality when establishing a relationship between an external object and another external object or a Rollbase Native object.
  - 1-1 or N-1 cardinality when establishing a relationship between a Rollbase Native object and an external object.

Note, if you checked the **Prompt for individual user credentials** check box, every user will need to enter credentials each time they log in. These are not the DataDirect Cloud credentials, but credentials for the data source, such as Salesforce, Eloqua, or Hubspot.

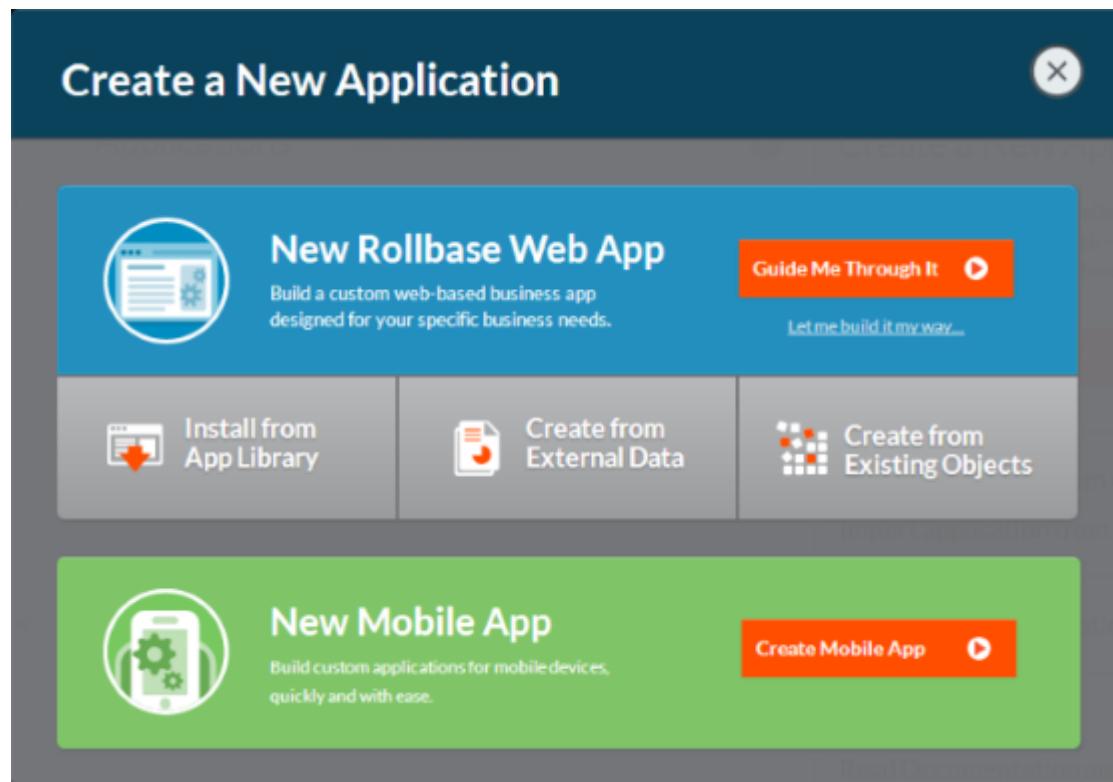


## Creating Rollbase Applications from Microsoft Access

The topics in this section explain how to create an entire Rollbase Application by uploading an MDB file (Microsoft Access Database format). For information on creating a Rollbase Object definition by uploading a spreadsheet in CSV or XLS format, see [Importing to Create a New Object](#) on page 363.

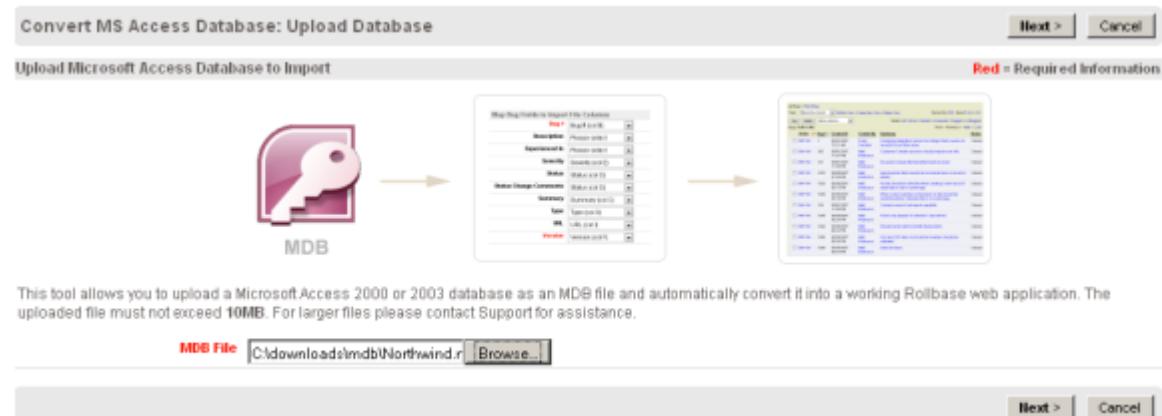
### Upload the MDB File

Click the New App link on the left side toolbar to display the Create Application dialog box:



Select the option, **Create from External Data**, and then select **Convert MS Access Database** option in the **Import Application Metadata** page to upload an MDB/ACCDB file. File size cannot exceed 10MB; please contact Rollbase support if you need to upload a larger file:

Important: The Rollbase conversion tool does not support MDB formats older than Access 2000.



## Create Objects from MDB Tables

Rollbase will read the structure of the uploaded MDB database. Each table in the MDB database can be converted into a new Rollbase Object and every row in that table may be converted into a Rollbase record. The wizard-style UI will assist you in Steps 2 and 3.

Enter the name for your new Application. The name of the uploaded database will be used by default.

Select which tables should be converted into Rollbase Objects. Select singular and plural names for these Objects (the table's name will be used by default). The number of records in the uploaded database is shown for each table for informational purposes.

As a first step to establish relationships among Objects, select the Primary Key column in each table (PK column names often end with an ID). You need to know the structure of the uploaded database to complete this task.

Select Objects for which associated Tabs will be created. Typically, every stand-alone Object (except for dependent ones) should have an associated Tab.

Finally, select Rollbase attributes for the newly created Objects. This is an optional step. We recommend using the Contact attribute for tables that represent peoples, Location for tables that represent things with a physical/geographical address and Workflow for Objects which later will be involved in Workflow processes.

For example, the resulting configuration for the Northwind database is shown below. Please note that:

- All but one Object has a Primary Keys selected.
- Three Objects have standard Rollbase attributes selected.
- The dependent Object (Order Details) has the Create Tab box unchecked.

Create Object?	Records	Singular Name	Plural Name	Primary Key	Create Tab?	Object Attributes
<input checked="" type="checkbox"/>	91	Customer	Customers	CUSTOMERID	<input checked="" type="checkbox"/>	Attributes:
<input checked="" type="checkbox"/>	77	Product	Products	PRODUCTID	<input checked="" type="checkbox"/>	Attributes:
<input checked="" type="checkbox"/>	9	Employee	Employees	EMPLOYEEID	<input checked="" type="checkbox"/>	Contact, Location,
<input checked="" type="checkbox"/>	2155	Order details	Order details	No PK	<input type="checkbox"/>	Attributes:
<input checked="" type="checkbox"/>	830	Order	Orders	ORDERID	<input checked="" type="checkbox"/>	Workflow, Location,
<input checked="" type="checkbox"/>	3	Shipper	Shippers	SHIPPERID	<input checked="" type="checkbox"/>	Attributes:
<input checked="" type="checkbox"/>	8	Category	Categories	CATEGORYID	<input checked="" type="checkbox"/>	Attributes:
<input checked="" type="checkbox"/>	29	Supplier	Suppliers	SUPPLIERID	<input checked="" type="checkbox"/>	Workflow, Location,

Click **Create Objects** to create eight Rollbase Objects and seven Tabs in an application called "Northwind".

## Create Fields and Records

At this point, all selected database tables have an associated Rollbase Object. Now we need to instruct the system what to do with the tables' columns. For each column we can choose from one of the following:

- Create a new Rollbase Field.
- Map a column to an existing Rollbase Field.
- Discard the column.

Important: You should map one meaningful column to a record's name Field. If you do not, all imported records will have no distinguishable name.

To establish relationships between Objects, select Foreign Keys (FK) in appropriate tables.

Note: Typically Foreign Keys in dependent tables have the same name as Primary Keys in main tables, e.g. SUPPLIERID in the Suppliers table (PK) matches SUPPLIERID in the Product table (FK). Correct selection of PK and FK is one of the keys to successful data import.

Select a name for the newly created Fields. By default, Rollbase derives this from the database's column name. And finally, you can mark some columns to prevent duplicate values in import.

The resulting mapping screens are shown below. Please note that in this example:

- Foreign Keys are selected to create relationships between Objects.
- Some columns are mapped to existing Fields (address and contact fields).
- Every Object has one column mapped to Record Name field.
- Although the type for new Fields is selected by default, you don't have to accept it. Sometimes a different type (Currency or Text Area) is more appropriate.
- Field names often require some editing for UI purposes.
- It is always a good idea to double-check your mappings before proceeding with data import.

The screenshot below shows what mapping for several Northwind tables (our example) looks like:

New Object "Product" from Table "Products"		New Object "Order details" from Table "Order Details"		New Object "Order" from Table "Orders"								
Database Columns	What do you want to do with this column?	What type of field should be created?	What should the label of this field be?	Database Columns	What do you want to do with this column?	What type of field should be created?	What should the label of this field be?	Database Columns	What do you want to do with this column?	What type of field should be created?	What should the label of this field be?	Do not a
PRODUCTID	New Field	Integer	Product ID	ORDERID	New Field	Orders.ORDERID	Order ID	ORDERID	New Field	Integer	Order ID	<input checked="" type="checkbox"/>
PRODUCTNAME	Product	Text	Product Name	PRODUCTID	Product	Products.PRODUCTID	Product ID	CUSTOMERID	Customer	Text	Customer ID	<input type="checkbox"/>
SUPPLIERID	New Field	Suppliers.SUPPLIERID	Supplier ID	UNITPRICE	Currency	Unit Price	<input type="checkbox"/>	EMPLOYEEID	New Field	Employees.EMPLOYEEID	Employee ID	<input type="checkbox"/>
CATEGORYID	New Field	Categories.CATEGORYID	Category ID	QUANTITYPERUNIT	Integer	Units In Stock	<input type="checkbox"/>	ORDERDATE	New Field	Date	Order Date	<input type="checkbox"/>
QUANTITYPERUNIT	New Field	Text	Quantity Per Unit	REQUIREDDATE	New Field	Date	<input type="checkbox"/>	SHIPPEDDATE	New Field	Date	Required Date	<input type="checkbox"/>
UNITPRICE	New Field	Currency	Unit Price	SHIPPEDDATE	New Field	Text	<input type="checkbox"/>	SHIPVIA	New Field	Integer	Shipped Date	<input type="checkbox"/>
UNITINSTOCK	New Field	Integer	Units In Stock	SHIPVIA	New Field	Text	<input type="checkbox"/>	FREIGHT	New Field	Integer	Ship Via	<input type="checkbox"/>
UNITSONORDER	New Field	Integer	Units On Order	FREIGHT	New Field	Currency	<input type="checkbox"/>	SHIPNAME	New Field	Text	Freight	<input type="checkbox"/>
REORDERLEVEL	New Field	Integer	Units On Order	SHIPNAME	New Field	Text	<input type="checkbox"/>	SHIPADDRESS	Street Address 1	Text	Ship Name	<input type="checkbox"/>
DISCONTINUED	New Field	Checkbox	Reorder Level	SHIPADDRESS	City	Text	<input type="checkbox"/>	SHIPCITY	City	Text	Ship Address	<input type="checkbox"/>
			Discontinued	SHIPCITY	State/Province	Text	<input type="checkbox"/>	SHIPREGION	State/Province	Text	Ship City	<input type="checkbox"/>
				SHIPREGION	ZIP/Postal Code	Text	<input type="checkbox"/>	SHIPPOSTALCODE	ZIP/Postal Code	Text	Ship Region	<input type="checkbox"/>
				SHIPPOSTALCODE	Country	Text	<input type="checkbox"/>	SHIPCOUNTRY	Country	Text	Ship Postal Code	<input type="checkbox"/>
				SHIPCOUNTRY			<input type="checkbox"/>				Ship Country	<input type="checkbox"/>

New Object "Suppliers" from Table "Suppliers"		What do you want to do with this column?		What type of field should be created?		What should the label of this field be?		Do not ask	
SUPPLIERID	New Field	Supplier	Text	Integer	Text	SupplierID	SupplierID	<input checked="" type="checkbox"/>	<input type="checkbox"/>
COMPANYNAME	New Field	New Field	Text	Text	Text	Companyname	Companyname	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CONTACTNAME	New Field	New Field	Text	Text	Text	Contactname	Contactname	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CONTACTTITLE	New Field	New Field	Text	Text	Text	Contacttitle	Contacttitle	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ADDRESS	Street Address 1	City	Text	Text	Text	Address	Address	<input type="checkbox"/>	<input checked="" type="checkbox"/>
CITY	City	City	Text	Text	Text	City	City	<input type="checkbox"/>	<input checked="" type="checkbox"/>
REGION	State/Province	ZIP/Postal Code	Text	Text	Text	Region	Region	<input type="checkbox"/>	<input checked="" type="checkbox"/>
POSTALCODE	ZIP/Postal Code	Country	Text	Text	Text	Postalcode	Postalcode	<input type="checkbox"/>	<input checked="" type="checkbox"/>
COUNTRY	Country	New Field	Text	Text	Text	Country	Country	<input type="checkbox"/>	<input checked="" type="checkbox"/>
PHONE	New Field	New Field	Text	Text	Text	Phone	Phone	<input type="checkbox"/>	<input checked="" type="checkbox"/>
FAX	New Field	New Field	Text	Text	Text	Fax	Fax	<input type="checkbox"/>	<input checked="" type="checkbox"/>
HOMEPAGE	New Field	New Field	Text	URL	Text	Homepage	Homepage	<input type="checkbox"/>	<input checked="" type="checkbox"/>

## Review Results

After submitting the Fields mapping, Rollbase creates the new Fields (if this option was selected for a particular database column). Then each database row will be converted into a Rollbase Object record.

To indicate completion of the import process, Rollbase sends you an email message:

3202 records have been imported.

From that point, you can use the fully functional Rollbase application that preserves both structure and data from the MDB database. Relationships between records (e.g., between Order and Order Details, between Employee and Orders) will be preserved (see screen shots below).

Order: Order

Tag Edit Delete More actions...

Order Info	System Info																				
<p><b>Order Information</b></p> <table> <tr> <td>Order</td> <td>Order</td> <td>Order Date</td> <td>04/23/1998</td> </tr> <tr> <td>Order ID</td> <td>11044</td> <td>Required Date</td> <td>05/21/1998</td> </tr> <tr> <td>Customer</td> <td>Wolski Zajazd</td> <td>Shipped Date</td> <td>05/01/1998</td> </tr> <tr> <td>Employee</td> <td>Margaret Peacock</td> <td>Freight</td> <td>\$8.72</td> </tr> <tr> <td>Shipper</td> <td>Speedy Express</td> <td>Ship Name</td> <td>Wolski Zajazd</td> </tr> </table>	Order	Order	Order Date	04/23/1998	Order ID	11044	Required Date	05/21/1998	Customer	Wolski Zajazd	Shipped Date	05/01/1998	Employee	Margaret Peacock	Freight	\$8.72	Shipper	Speedy Express	Ship Name	Wolski Zajazd	
Order	Order	Order Date	04/23/1998																		
Order ID	11044	Required Date	05/21/1998																		
Customer	Wolski Zajazd	Shipped Date	05/01/1998																		
Employee	Margaret Peacock	Freight	\$8.72																		
Shipper	Speedy Express	Ship Name	Wolski Zajazd																		
<p><b>Workflow Information</b></p> <table> <tr> <td>Workflow Process</td> <td>Default</td> </tr> <tr> <td>Workflow Status</td> <td>Created</td> </tr> <tr> <td>Workflow Actions</td> <td></td> </tr> </table>	Workflow Process	Default	Workflow Status	Created	Workflow Actions																
Workflow Process	Default																				
Workflow Status	Created																				
Workflow Actions																					
<p><b>Location and Address Information</b></p> <table> <tr> <td>Street Address 1</td> <td>ul. Filtrowa 68</td> <td>Street Address 2</td> <td></td> </tr> <tr> <td>City</td> <td>Warszawa</td> <td>State/Province</td> <td></td> </tr> <tr> <td>ZIP/Postal Code</td> <td>01-012</td> <td>Country</td> <td>Poland</td> </tr> </table>	Street Address 1	ul. Filtrowa 68	Street Address 2		City	Warszawa	State/Province		ZIP/Postal Code	01-012	Country	Poland									
Street Address 1	ul. Filtrowa 68	Street Address 2																			
City	Warszawa	State/Province																			
ZIP/Postal Code	01-012	Country	Poland																		
<p><b>Order Details</b>   New Order Detail   Quick Create   Attach Order Detail</p> <p>View (Related Records): All Order Details   Edit View   New View   Clone</p> <p>Subscribe: RSS Export: XLS   CSV   Google</p> <p>Tag Delete More actions...</p> <p>Select: All   None   Clear All</p> <p>Order Details 1-1 of 1</p> <table border="1"> <thead> <tr> <th>Action</th> <th>Product</th> <th>Quantity</th> <th>Unitprice</th> <th>Discount</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/> Edit   Del   Detach</td> <td>Tarte au sucre</td> <td>12</td> <td>\$49.30</td> <td>\$0.00</td> </tr> </tbody> </table>		Action	Product	Quantity	Unitprice	Discount	<input type="checkbox"/> Edit   Del   Detach	Tarte au sucre	12	\$49.30	\$0.00										
Action	Product	Quantity	Unitprice	Discount																	
<input type="checkbox"/> Edit   Del   Detach	Tarte au sucre	12	\$49.30	\$0.00																	

Employee: Margaret Peacock

**Employee Info** **System Info**

**Contact Information**

First Name	Margaret	Middle Name	
Last Name	Peacock	Title	Sales Representative
Phone	(206) 555-8122	Mobile Phone	
Fax		Email Address	
Contact Owner		Employee ID	4
Title Of Courtesy	Mrs.	Birth Date	09/19/1958
Hire Date	05/03/1993	Extension	5176
Notes	Margaret holds a BA in English literature from Concordia College and an MA from the American Institute of Culinary Arts. She was temporarily assigned to the London office before returning to her permanent post in Seattle.		

**Location and Address Information**

Street Address 1	4110 Old Redmond Rd.	Street Address 2	
City	Redmond	State Province	WA
ZIP/Postal Code	98052	Country	USA

**Orders** | New Order | Quick Create | Attach Order

View (Related Records): **All Orders** |   

Subscribe: [RSS](#) | [Export XLS](#) | [CSV](#) | [Google](#)

**Action** **Order** **Order Date** **Required Date** **Shipped Date** **Ship Name**

<input type="checkbox"/> <a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Detach</a>	Order	05/05/1998	06/03/1998		Bon app'
<input type="checkbox"/> <a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Detach</a>	Order	05/05/1998	06/02/1998		Ernst Handel
<input type="checkbox"/> <a href="#">Edit</a>   <a href="#">Delete</a>   <a href="#">Detach</a>	Order	04/30/1998	05/28/1998		Reggiani Caseifici

Select All | None | Clear All

[Next](#) | [Last](#)

## Creating Rollbase Applications from Salesforce Applications

In a matter of minutes, you can create a new Rollbase Application by migrating an existing Salesforce.com CRM instance, or any other application built on the Force.com platform, along with all of its data.

The migrated application will include:

- Object definitions (including fields)
- Object Menus
- Relationships between imported objects
- Pages (including sections and related lists)

**Note:** Because Rollbase and Salesforce.com use different languages for formulas, you will need to substantially re-write formulas after completion of the migration process. Only in the simplest cases can formulas be used as-is.

For a video demonstration of migrating applications from Salesforce.com and the Force.com platform see <http://www.rollbase.com/deforce.shtml>.

**Note:** Note: Salesforce.com limits number of API calls a particular customer can make in 24 hours span. If you're receiving REQUEST\_LIMIT\_EXCEEDED error that means that your limit has been exceeded during the import process.

## Migrating the Application

Rollbase uses Web Service APIs to extract information from your Salesforce.com or Force.com account. The migration process builds a new Rollbase application that contains all of your data. Before attempting migration, ensure that your organization has access to Salesforce.com APIs.

1. Before creating the new application in Rollbase, log in to your Salesforce or Force account and select the application you want to migrate (for example, if you want to migrate your Salesforce.com CRM app be sure to select the "Sales" app). Rollbase migrates the currently active application.
2. When logged into Rollbase, click **New App** to create a new application.
3. Select **Import from External Metadata**.
4. Click **Create**.
5. For **Type**, select **Salesforce (Force.com)**
6. Provide your Salesforce.com credentials (Note: Rollbase will not store or re-use your Salesforce.com credentials. They are only used once to temporarily retrieve your data):
  - User name (with sufficient permissions to access Web API).
  - Password.
  - Security token. To retrieve your Salesforce.com API security token, login to your Salesforce.com account and click **Setup > My Personal Information > Reset Security Token**.
7. Rollbase will use Web API calls and retrieve the following metadata from your most recently accessed Salesforce.com application:
  - Object Menus (Web and system menus cannot be extracted because Force.com does not expose them as metadata).
  - Custom objects and most Standard objects, including their components:
    - Field definitions (including formula fields)
    - Relationships
    - Page layout definitions (which are converted into Rollbase Pages)
8. Use the checkboxes to identify the objects, fields and relationships you want to import from Salesforce.com into your new Rollbase application.

---

**Note:** If a Salesforce object already exists in your Customer, the migration tool will not attempt to import any components.

---

9. Click **Create** to complete the migration.

After automatically creating and configuring your new Application, Rollbase will proceed to import the actual data from your Salesforce.com account (providing that you've selected this option). The data import is done in asynchronous fashion and you will receive an email with import results when the process is completed. Each call to Salesforce API brings back up to 500 records of a particular type. Notice that not only are all of your object definitions, fields, relationships and data migrated, but the exact layout of all of your application pages is maintained in your new Rollbase application. At this point you can start working with your new application just like any other Rollbase app, by adding data, editing objects, pages, etc.

## Using External Tables as Rollbase Objects

In a private cloud environment, external objects allow you to integrate tables managed by other applications into a Rollbase application. Each external object maps to an external table. External objects have almost all of the same features as native Rollbase objects, but the data remains stored in the external table, rather than in Rollbase tables.

---

**Note:** The use of external database tables as Rollbase objects is beta functionality and should not be used in production environments.

---

For other ways to integrate data using external objects, see [Using DataDirect Cloud to Access External Data](#) on page 343 or [Creating Rollbase Objects from OpenEdge Services](#) on page 322.

The following topics describe how to use external tables in a private cloud environment:

## Using an External Database and External Objects for Private Cloud

The [Configuring a Supported Database](#) on page 455 topic explains how to create a fresh Rollbase database by running the default SQL creation script. If you already have data in an existing database, you can wrap these existing tables and their data into a Rollbase external object definition. This method allows you to use existing without impacting the underlying table or any existing applications that rely on it. External database support is available for MySQL, OpenEdge, DataDirect Cloud, Oracle, SQL Server databases.

Rollbase tables must reside in the same database where existing tables you want to integrate with reside because Rollbase transactions are complex and often include multiple updates in several tables. Running these updates across more than one database has proven to be inefficient. Instead of creating the Rollbase database tables (schema in Oracle terms) in a new empty database, you will run the default Rollbase SQL creation script in your existing database that contains the existing tables you want to leverage. The Rollbase SQL creation script will add 24 new tables, with an `RB_` name prefix, without affecting existing tables.

[Managing Databases](#) on page 476 explains how to describe Rollbase databases in the `databases.xml` configuration file. In this way you can have multiple databases for different tenants based on different external tables you want to integrate this with. Using this approach to describe a new database, add a new XML attribute `isExternal`:

```
<Database name="RB_CUST1" isExternal="yes">..</Database>
```

Marking a database as external allows for the creation of external objects for Rollbase customer tenants assigned to that database.

## External Object Overview

You can add external objects to applications, publish, and install these applications. Make sure that target customers have access to the external tables mapped to the external objects. You can edit external objects to change their name and other properties. External objects and their fields can be configured and used in as native Rollbase objects, with the following limitations:

- The field creation process is limited as explained in [External Object Fields and Attributes](#) on page 353
- Full-text search and search engine indexing is not available for external object fields
- External records cannot be used as application seed records

The functionality available for external objects and their fields includes, but is not limited to:

- Configurable Pages
- Configurable Views
- Detailed Search and Filtering
- Email and Document Templates
- Triggers and Workflows
- Input validation through Triggers and Field-level Validation
- Reports, Charts and Gauges
- Unique Indexes
- Client-side and Server-side API Usage
- REST and SOAP API

## External Object Fields and Attributes

There is a fundamental difference between the way new fields are created for external objects and native Rollbase Objects. In the latter case new fields can be created either through enabling a new object attribute (such as Contact or Location) or by creating a field manually in the Rollbase user interface.

External objects are stored in an existing table with a structure which cannot be modified. For this reason new fields in external objects can only be created in one of the following cases:

- They rely on a column in the external table that exists, but has not yet been mapped to a Rollbase field in that External Object
- They do not require a database column in the external table to store data, such as: Formulas, Templates, Roll-Up Summary, Integration Links

The Fields configuration process is almost identical to native Rollbase object fields. As usual, newly created fields on external objects can be added to pages, views, reports, and referenced anywhere normal object fields can be such as formulas, templates, triggers, and validations. Creation or deletion of a field in an external table most likely will require adjustment in the SQL queries as described in [SQL Queries for External Objects](#) on page 355. For your convenience, the system will redirect you to the SQL Adjustment page after field is created or deleted.

You can assign object attributes by mapping fields from particular attribute (such as Contact) to an available data column. If an Attribute box is checked for an external object, all of its fields must be mapped and no column can be mapped to more than one field. Some fields must be mapped to columns with a specific name. For example, a workflow **Status** must be mapped to a column named **STATUS\_ID**.

The screenshot shows a configuration interface for a 'Workflow' object. The 'Workflow' attribute is checked, and its description states: 'Objects with the Workflow attribute can be routed through an automated or manual workflow process defined by a set of workflow statuses, actions, and events. Workflow events allow you to create and deploy business rules that are manually or automatically triggered based on specific criteria defined by you. A field called Workflow Status will be added when this attribute is enabled.' Below this, there are two mapping sections:

- Workflow Process:** Manager\_ID is mapped to Process PROCESS\_ID. A red message at the bottom of this section states: 'This field must be mapped to PROCESS\_ID column. If such column does not exist you cannot use this Attribute.'
- Workflow Status:** -- Please select -- is mapped to Status STATUS\_ID. A red message at the bottom of this section states: 'This field must be mapped to STATUS\_ID column. If such column does not exist you cannot use this Attribute.'

## External Relationships

Database tables typically have primary key to foreign key relationships between records. Rollbase can wrap these relationships the same way as native relationships between objects.

A field of an external object (database column) can serve as a primary key if:

- Its type is text or integer
- It has a "unique value" attribute

When creating fields for new external object, you can select a **Foreign Key** data type for text or integer database columns. In this case you need to select a **Primary Key** too.

If you do not define a relationship when you create the external object, you can do it later. On the object view page, from the **Relationships** section, click **New External Relationship**. Select an unused DB column which can serve as a foreign key (if any), and select a primary key on another external object. The types of the two key columns must match. In both cases, the system will create a relationship between two external objects that works much the same way as normal object relationships. The relationship will be in the form of primary key to foreign key and only one-to-one and one-to-many relationships are supported.

The screenshot shows the 'Choose Related Object' dialog. At the top, it says 'Choose Related Object' and 'Red = Required Information'. Below that, 'Foreign Key' is selected and 'GROUP\_CODE' is chosen. A message says 'Choose the external object to establish a relationship with through Primary Key - Foreign Key fields.' The 'Related Object' section shows two options:

Object Name	Table	Primary Key
Employee (hierarchy)	EMPLOYEE	<input type="radio"/> Employee Id <input type="radio"/> Email
Employees Group	EMPL_GROUP	<input type="radio"/> Group Id <input type="radio"/> Group Code

Note that relationships involving external objects support only the following scenarios:

- 1-1 or 1-N cardinality when establishing a relationship between an external object and another external object or a Rollbase Native object.
- 1-1 or N-1 cardinality when establishing a relationship between a Rollbase Native object and an external object.

## SQL Queries for External Objects

Although Rollbase Private cloud can automatically read the structure of an external table, the actual design of the external database remains unknown to Rollbase. For this reason, during the process of creating external objects, you have the option to adjust the SQL queries that Rollbase generates automatically. If Rollbase experiences a SQL error while working with external objects, an error will be displayed. The **Adjust SQL** page allows you to fix those errors by editing the SQL directly. For existing external objects, the **Adjust SQL** page is available directly from the **More Actions** drop-down list on an object view page. If you add or modify a field for an external object, the **Adjust SQL** page displays automatically.

Adjust SQL Queries Red = Required Information

<b>SQL SELECT</b>	<pre>SELECT EMPLOYEE_ID, LAST_NAME, DESCRIPTION, EMAIL, FIRST_NAME, GROUP_ID, HIRE_DATE, SALARY FROM EMPLOYEE</pre>
WHERE ..... ORDER BY .....	
<b>SQL INSERT</b>	<pre>INSERT INTO EMPLOYEE (EMPLOYEE_ID, LAST_NAME, DESCRIPTION, EMAIL, FIRST_NAME, GROUP_ID, HIRE_DATE, SALARY) VALUES (!id), (!name), (!description), (!email), (!first_name), (!R806272), (!hire_date), (!salary))</pre>
.....	
<b>SQL UPDATE</b>	<pre>UPDATE EMPLOYEE SET EMPLOYEE_ID=!id, LAST_NAME=!name, DESCRIPTION='!#CURR_USER.id', EMAIL=!email, FIRST_NAME=!first_name, GROUP_ID=!R806272, HIRE_DATE=!hire_date, SALARY=!salary WHERE EMPLOYEE_ID=!id</pre>
.....	
<b>SQL DELETE</b>	<pre>DELETE FROM EMPLOYEE WHERE EMPLOYEE_ID=!id</pre>
.....	
<b>ID for New Record</b>	<pre>SELECT MAX(EMPLOYEE_ID)+1 FROM EMPLOYEE</pre>
.....	

To access and manipulate data records in external tables, Rollbase uses SELECT, INSERT, UPDATE, and DELETE SQL queries. To adjust SQL queries, you must be familiar with SQL, details of your database implementation, and how Rollbase formulates SQL queries.

Note the following about how Rollbase generates and handles an SQL query:

- Rollbase encloses table and column names in double quotes whenever the names contain characters not recognized as an identifier by SQL syntax or when the names are reserved keywords. This conforms to the ANSI standards. Therefore, you must ensure that your external database supports double quotes in an SQL query.

By default, the latest versions of the databases certified by Rollbase support double quotes in queries. If your database does not support double quotes for tables and column names, then you must configure it to do so. For example, as MySQL does not support double quotes by default, you must add `ANSI` to the comma-separated values of the `SQL Mode` property.

- If Rollbase encounters an identifier that is not listed as a reserved SQL keyword or a special character in its `shared.properties` file, but in reality is an SQL reserved keyword or a special character in your database, then the SQL query might result in an error because the column or table name will not be enclosed with double quotes in the SQL query.

In such cases, you must manually edit the SQL query in the **Adjust SQL** page or add those SQL keywords and special characters to `shared.properties`.

Progress recommends that you verify and add any reserved keywords and special characters in the `shared.properties` file before generating SQL queries for External objects. In `shared.properties`, you must locate `SQLKeywords` and `SQLSpecialChars` code snippets and then add comma-separated keywords and special characters respectively. You must restart Rollbase for any update in `shared.properties` to take effect.

You can use stored procedures with parameters to replace the default `INSERT`, `UPDATE`, and `DELETE` SQL queries. Rollbase will use these queries exactly the way they're provided. However since `SELECT` queries are necessary for filtering and sorting, a `SELECT` query cannot be replaced with a stored procedure. The query or stored procedure must fetch an ID for new records. For Oracle databases, the query can include a sequence to fetch record IDs.

SQL queries must use actual database column names. To supply data to the database, `INSERT` and `UPDATE` queries must use template tokens for Rollbase fields. To preview available tokens and corresponding column names use the **View Table Columns** button which brings up helper information as shown below. The **Preview Query** button allows you to see whether a query is working correctly.

#### Columns of EMPLOYEE Table Mapped to Employee Fields

Field	Field Type	Integration Name	Column Name
Description	Text Area	{description}	DESCRIPTION
Email	Email Address	{email}	EMAIL
Employee Id	Integer	{id}	EMPLOYEE_ID
First Name	Text (50)	{first_name}	FIRST_NAME
Group Id	Lookup (Employee Group)	{R806272}	GROUP_ID
Hire Date	Date	{hire_date}	HIRE_DATE
Last Name	Record Name	{name}	LAST_NAME
Salary	Currency	{salary}	SALARY

Helpers	
{CURR_CUSTM.id}	Customer ID
{CURR_CUSTM.name}	Customer Name

If you are using an external database in a multi-tenant environment you need to ensure isolation of customers' data. You can use the helper tokens `{ !#CURR_CUSTM.id }` and `{ !#CURR_CUSTM.name }` in WHERE clauses. For instance, if you're using column `CUST_ID` to store customer's ID, add the following to the SELECT query:

```
.. WHERE CUST_ID={ !#CURR_CUSTM.id }
```

Please use only tokens listed in the helper table. Other tokens will be ignored, making the SQL syntax invalid. At runtime these tokens will be replaced with actual values from the records. This offers the most flexible way to build queries which may include calls to stored procedures.

The following table explains the formats used by different types of data fields for tokens in external queries.

Field Type	Format	Example
String	Text in single quotes. Single quote inside text replaced by two.	'TEST' 'O''Neal'
Numeric	Number	123.45
Date	'yyyy-mm-dd'	'2012-06-15'
Date/Time	'yyyy-mm-dd hh:mm:ss.000'	'2012-06-15 18:45:12.000'
Foreign Key	Number or text in single quotes	123456 or 'XYZ'

## Creating an External Object from an External Database Table

If your customer tenant resides in a database marked as external you can create object definitions that map to an existing table from that database. To create an external object from an external database table, follow these steps:

1. On the application home page, click the add symbol (+) in the tab area.  
The **What do you want to create?** dialog displays.
2. Select **External Database**.
3. Click **Next**.
4. Select the table to use as the basis for your new external object. This can be any table other than the following:
  - A Rollbase table (name starts with RB\_)
  - A system table (name starts with "~" symbol)
  - A table used by an existing external object
5. For the selected table, optionally select the text column to be used as the **Record Name** field.

If no column is selected, Rollbase creates the name from the **Record Name Template** or from the object single name if no template is provided.

Select Table	EX_USER	<input type="button" value="▼"/>
Select Name Column	-- Please select --	<input type="button" value="▼"/>

6. If desired, specify a single numeric unique column as the table's primary key.
7. Click **Next**.
8. If you have not specified a primary key, select one or more columns from the **Available** list and move them to the **Composite PK** list with the Right arrow. Reorder the **Composite PK** list as desired with the up and down arrows.

Object from CUST1 Database: Create Fields

Select Columns for Composite Primary Key

<b>Available</b> GROUP_NAME	<b>Composite PK</b> CUSTOMER_ID GROUP_CODE
<input type="button" value="&gt;"/> <input type="button" value="&lt;"/>	<input type="button" value="x"/> <input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="x"/>

New Object "g1" from Table "EXT\_GROUP"

Database Columns	What type of field should be created?	What should the label of this field be?	Integration Name	Do in this
CUSTOMER_ID	Integer	Customer Id	customer_id	<input type="checkbox"/>
GROUP_CODE (50)	Text	Group Code	group_code	<input type="checkbox"/>
GROUP_NAME (50)	Text	Group Name	group_name	<input type="checkbox"/>

9. Specify the usual object definition attributes:

- **Singular Name**
- **Plural Name**
- **Integration Name**
- **Optional Description**

10. Map columns in the selected table to new fields in the newly created external object. This is similar to creation of fields while importing data from a CSV file, Excel, MS Access database or Salesforce.com object.

New Object "Employee" from Table "EMPLOYEE" Red = Required Information

Database Columns	What type of field should be created?	What should the label of this field be?	Do not allow duplicate values in this field.
CLUB_MEMBER (1)	Checkbox	Club Member	<input type="checkbox"/>
DESCRIPTION	Text Area	Description	<input type="checkbox"/>
EMAIL (100)	Text	Email	<input type="checkbox"/>
EMPLOYEE_ID	ID	Employee Id	<input checked="" type="checkbox"/>
FIRST_NAME (50)	Text	First Name	<input type="checkbox"/>
GROUP_NAME (45)	Text	Group Name	<input type="checkbox"/>
HIRE_DATE	Date	Hire Date	<input type="checkbox"/>
LAST_NAME (50)	Record Name	Last Name	<input type="checkbox"/>
SALARY	Decimal	Salary	<input type="checkbox"/>

Each database column can be mapped to a field in newly created external object. These fields will be added to the following pages: View, New record, and Edit. If you do not map column to a new field you can create fields from unused columns later.

## Importing Data

Importing data is an easy way to add new records, update or delete existing records in bulk, or create new objects and add records for them. You can import data from spreadsheets and CSV (comma-separated value) files. For existing objects, the **Import** option is available from the drop-down menu on the tab. You can also configure list views to display an import link in the section header using the **Page Editor**. For creating new objects and associated records, see [Importing to Create a New Object](#) on page 363.

Please understand the following concepts before attempting an import:

- Spreadsheets must be saved in `.xls` format.
- The file to be imported cannot exceed 1MB. For Rollbase Private Cloud customers, the maximum file size is configurable.
- Only data on the first worksheet of spreadsheet files will be imported.
- Rollbase assumes that each row starting from row 2 in a spreadsheet represents a record to be created or updated. Row 1 is always interpreted as the header. When creating an object definition from a spreadsheet, the header row provides the field names. The following shows a sample spreadsheet formatted for importing into Rollbase:

	A	B	C	D
1	Lead	Company	State/Province	Email Add
2	William THOMPSON	Oracle	KY	
3	William MANROD		CT	
4	William HENSLER		GA	

- For picklist fields, Rollbase tries to find existing picklist values with the integration code or display name that matches the value in the spreadsheet. If none exist and the **Create new picklist items during import** checkbox is checked, Rollbase creates a new picklist value. However, as a precaution against potential mismatches, Rollbase will not create new picklist values in picklists with a shared source (such as country and state/province).

- For lookup fields where duplicate values are not allowed, such as those corresponding to the object name or ID, the file to import from must have a column or field with unique values for each record. (To determine which fields these are, navigate to the field definition in setup and view the field's **Advanced Field Properties**. **Do not allow duplicate values in this field** will be checked.) When you map such a field during import, the system will find any corresponding records with the matching record name or ID value and attach them if found. To map more than one value to a lookup field, separate values with the pipe "|" character, such as 123456|789101. You can import complex data structures with Primary Keys (PK) and Foreign Keys (FK) and replace the PK - FK with Rollbase relationships, see [Importing Related Objects](#) on page 365.
- Only one import job can be running in a customer tenant at a time; the current job must finish before anyone can start a new job. Rollbase enforces this limitation for performance reasons.
- You can save the map you use to import with and manage the stored maps from the **Data Maps** section of the object definition. In addition to a name, you can set an integration code for your map. Import maps can also be used for data import through the REST API. For information on REST methods, see [Rollbase REST Methods](#) on page 732.
- Before importing a large amount of data, Progress recommends testing the process by selecting the **Test** import mode that creates a detailed report for the first five rows in your spreadsheet. When you are satisfied with the results, repeat the import with a larger amount of data.
- For large imports, Rollbase creates a job and places it into a queue for asynchronous processing and then sends an email to you with results of the import once the job is completed. This message includes a report detailing successfully imported records and errors encountered during the import process.
- If a particular spreadsheet row contains errors such as an empty value for a required field, or unparsable data, that row is ignored, but is reflected in the import report described next.

## Import Report

For audit purposes, you can find import reports on the **Administration Setup** page, in the **System Event Logs** section. The first line of an import report contains a timestamp indicating when the import process started. Due to the queuing mechanism, there may be a delay between when you submit the import and when it actually begins. Next, the results of each of the first five spreadsheet rows are displayed, by line number:

```

Started at 06/21/2010 06:16 PM
2: Import error: Field Permit No. cannot accept null values
3: Permit "Test 1" has been created
4: Permit "Test 2" has been created
5: Import error: Error importing data for mechanical_cost Field: For input
string: "1200-1300"
6: Import error: Error importing data for electrical_admin Field: multiple
points>

```

## Compatible Import Data Types

When adding records to existing objects or modifying records, the format of data in the spreadsheet must be compatible with the data types of object fields. The following table describes the types of data expected for Rollbase fields:

Type	Description
Numeric	Single number

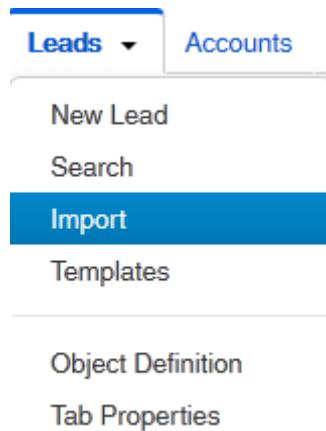
Type	Description
Check Box	"true", "T", "y", "yes", "1" for true, false otherwise
Date	Date in user's selected format or ISO 8601 format
Date/Time	Date/Time in user's selected format or ISO 8601 format
Time	Time in user's selected format
Duration	Number of milliseconds for time interval
Lookup	Name, ID, or selected unique field (see above) for related record separated by ' '
Object Field Type	Data Type to Import
Picklist (single), Radio Buttons	Integration code or name of lookup item
Picklist (multiple), Group of Check Boxes	Integration codes or names of lookup items separated by commas or ' '
Workflow Status	Integration code or ID of workflow status
Workflow Process	Name or ID of workflow process
User's Role	Integration code for role
Template	Select ID of template
Date/Time Format	Selector Number of selected format (from 0 - standard US format)
Time Zone	Selector Valid ID of time zone (such as "America/Los_Angeles")
Parent Object	ID of record to which Communication Log record belongs

## Importing for Existing Objects

Before importing, be sure to understand the requirements and how import works, as described in [Importing Data](#) on page 359.

To import a spreadsheet or CSV file to create new records, update existing records, or delete records, follow these steps:

1. Navigate to the tab or view of the object you want to import.
2. From the drop-down menu on the object tab, select **Import**:



3. Click **Choose File** to select an Excel Spreadsheet (.XLS) or a Comma-Separater Values (.CSV) file location.
4. If you selected a CSV file, specify its **File Encoding** format and **CSV File Separator**.

**Note:** By default, the file encoding is ISO-8859-1 and the file separator is comma.

5. Choose the appropriate action for creating and/or updating records.
6. If you selected any action other than **Create new records**, select a **Unique field** from the drop-down list.

The value in this field will be used to ensure that the correct record is modified.

7. Click **Next**
8. Map the fields of the destination object to columns in the uploaded spreadsheet by selecting the desired spreadsheet column from the drop-down list on the right of each Field. Rollbase makes a best guess at automatically selecting field mappings based on the column headers in Row 1, but it is up to you to verify and change mappings as needed.
  - **Not mapped** prevents the field from being populated
  - **Field default** uses the default value set for new record creation. You must explicitly set this if you want to use default values.

Read-only fields do not participate in mappings. Fields with the attribute, **This field is required in all forms** are highlighted in red and must be mapped to one of the spreadsheet columns in order to proceed.

9. Select the **Import Mode**:
  - **Normal**: Runs all triggers on creation of new records and creates picklist values while importing rows. This mode works best for medium-size imports.
  - **Test**: Limits your import to the first five records and displays a detailed report. Use this mode to test out your mapping before launching large imports.
  - **Bulk**: Optimized for faster processing of large imports. In this mode, triggers are not executed and new picklist values are not automatically created.

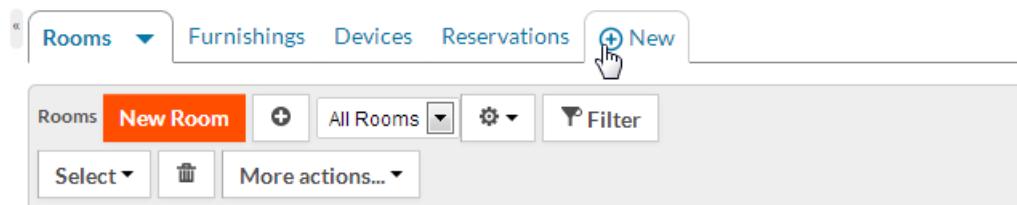
10. Optionally, leave the **Create new picklist items during import** box checked if you want the system to create new picklist items during import. This box is checked by default, and is not available in Bulk mode.
11. Optionally, click **Save Map** to use the same column to field mappings for other imports.
12. Click **Submit** to start the import process. Files less than 20KB in size will be processed immediately and the results will display onscreen.

## Importing to Create a New Object

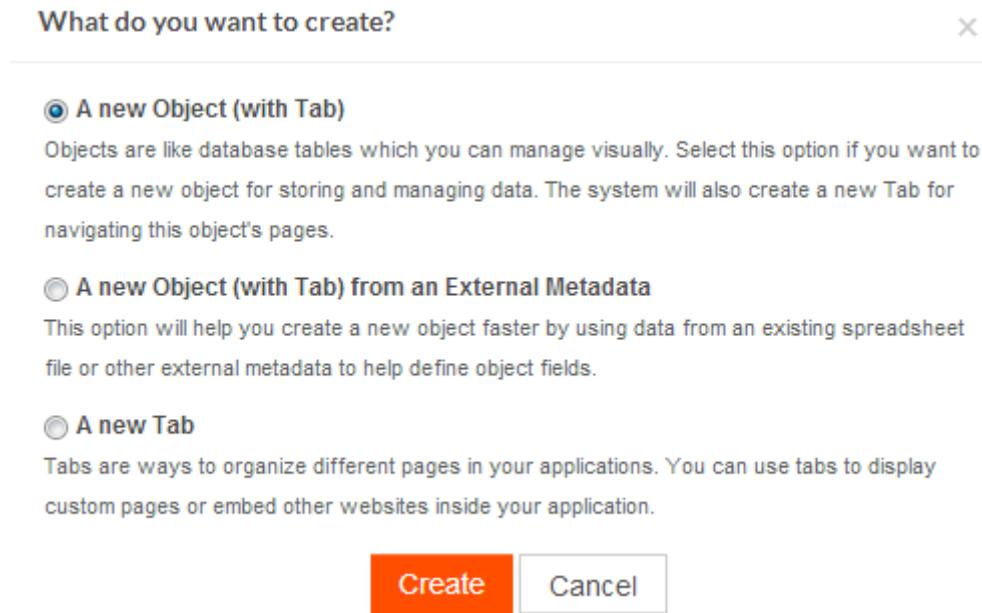
Before starting, determine the singular and plural names for the new object, and any object attributes to apply. For information on objects, see [Object Definition Overview](#) on page 82. And then, prepare a spreadsheet you wish to import in XLS (Microsoft Excel) or Comma-Separated Format (CSV) format.

To perform the import, follow these steps:

1. Navigate to the application.
2. Click the **New** tab that displays to the right of the existing object tabs:



The **What do you want to create?** dialog displays:



3. Select **A new Object (With Tab) from External Metadata** and click **Create**.
4. Select a **Spreadsheet File** and click **Next**.
5. Click **Choose File** to select an Excel Spreadsheet (.XLS) or a Comma-Separate Values (.CSV) file location.
6. Select the file and click **Open**.
7. If you selected a CSV file, specify its **File Encoding** format and **CSV File Separator**.

**Note:** By default, the file encoding is ISO-8859-1 and the file separator is comma.

8. Enter the **Singular Name** and **Plural Name** for the new object definition and choose the appropriate properties and attributes.

Personal Setup Applications Setup Administration Setup << return to Room Reservation

Application Setup > Objects > New Object

A new Object from a Spreadsheet: Create Object

**Create Object** **Cancel**

Upload Spreadsheet to Import



The Import tool allows you to create a new Object definition from an Excel spreadsheet (XLS) or any comma delimited file (CSV), and simultaneously import all of your object data. Provide the file to import below as well as the properties and attributes for your new object. You can then be asked to map each column in your spreadsheet to a new or pre-defined field and you can specify the type of each new field to be created along with the field's label. The import tool will create one record for each row of data in the file.

Spreadsheet File  TestImport.xls (50 MB max)

**Object Properties**

Define a singular and plural name for this object definition. These names will be used throughout your account to refer to one or more records of this type.

Singular Name	<input type="text" value="Title"/>	Example: Project
Plural Name	<input type="text" value="Titles"/>	Example: Projects

**Optional Object Properties**

Select the appropriate properties based on how you want records of this object type to behave:

Property	Description
<input type="checkbox"/> Audit Trail	Select this option to enable creation of Audit logs when values of selected fields are changes or by invoking triggers and API.
<input type="checkbox"/> Flagging	Select this option if you want the ability to flag records for follow-up. Flagging applies uniquely to each user.
<input type="checkbox"/> Viewed Tracking	Select this option if you want to visually differentiate records that have been viewed from those that have not.

9. Click **Create Object**.

10. For each spreadsheet column, select the appropriate action in the first drop-down menu:

- **New Field** — Create a new field and import a column's data into that field.
- **Discard** — Rollbase will not use the values in this column.

- **Object\_name** — The lookup field. You must map one column to this menu item as shown in the example below for a new Title object. Map a column that uniquely identifies each record.

11. From the **What type of field should be created** menu, select a data type for each field.
12. Optionally, change the default names for the new fields.
13. For fields that should contain unique values, check **Do not allow duplicates in this column**
14. Click **Create Fields**.

Next, the data import will proceed and for small spreadsheets, you will see results on the screen. For larger imports, you will receive an email.

## Importing Related Objects

Importing from a spreadsheet limits you to importing one type of object at a time. It is often useful to be able to set up relationships between imported objects. For example, you may have a database table of Vendors with a foreign key and a database table of Products with a foreign key pointing to the primary key of Vendors. To import this data structure and preserve those relationships, follow these general steps:

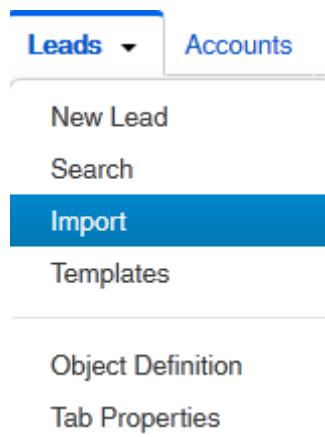
1. Create a Vendor object definition.
2. Create a text field to hold the Vendor's primary key. Make sure to check the attribute, **Do not allow duplicate values in this field**.
3. Import Vendor data from a spreadsheet or CSV file and populate the fields, including values for each primary key.
4. Create a Product object definition with a text field to hold the foreign key.
5. Create a many-to-one relationship between Products and a Vendor.
6. Import Products from a spreadsheet or CSV file. When importing, map the foreign key field to the unique primary key field in the Vendor Object.

If you no longer need the foreign and primary key fields, delete them. Rollbase replaces them with internal IDs.

# Deleting Multiple Records by Importing a Spreadsheet

To delete records, create a spreadsheet that lists the unique identifiers for the records you want to delete. Use one spreadsheet for each object type. The process is similar to that used for importing. Follow these steps to delete records using a spreadsheet:

1. Navigate to the list of objects. One way to do this is as follows:
  - a) Select the application that contains the type of objects you want to view.  
The application object tabs display.
  - b) Select the tab for the object type of interest.
2. From the drop-down menu on the object tab, select **Import**:



3. Click **Browse** to select the spreadsheet to import from.
4. Select the appropriate options for **File Encoding** and **CSV file separator**.
5. For **Action**, choose **Delete existing records**.
6. Select a **Unique field** from the drop-down list.  
The value in this field will be used to identify the records to be deleted.
7. Click **Next**
8. Select the **Import Mode**:
  - **Normal**: Runs all triggers on creation of new records and creates picklist values while importing rows. This mode is best for medium-size imports.
  - **Test**: Limits your import to the first five records and displays a detailed report. Use this mode to test out your mapping before launching large imports.
  - **Bulk**: Optimized for faster processing of large imports. In this mode triggers are not executed and new picklist values are not automatically created.
9. Click **Submit** to start the deletion process. Small files (less than 20KB) are processed immediately and the results will display onscreen.

# Exporting from Views and Reports

Rollbase provides export capabilities from Views and Reports. The header part of each View component contains links to export the View's records in XLS, CVS, or to a spreadsheet in Google Docs:

Subscribe: [RSS](#) Export: [XLS](#) | [CSV](#) | [Google](#)

The first two options are straightforward: all records and columns in the currently selected View or Report are exported and available for viewing in a separate window that can be a Microsoft Excel or your browser, depending on the client machine's configuration.

Important: Exports created in this way are limited to 1000 rows for performance reasons.

For information on exporting to Google spreadsheets, see [Integrating with Google Applications](#) on page 367.

# Integrating with Google Applications

Rollbase provides an integration with Google applications as described in the following topics:

## Incoming Gmail

Once you have provided your Google credentials and enabled IMAP, you can also gain limited access to your incoming emails from within Rollbase. Rollbase will read your Google inbox and display your messages there. The messages are ready dynamically and you cannot interact with them, nor are they stored in Rollbase.

Tip: If you do not see the "Gmail" tab in your default "Rollbase" application use the Page Editor and add the "Incoming Emails (Gmail)" component to any Generic page.

The list of messages in your Gmail inbox displays the sender's email address, as well as the subject and date of the email. Unread messages are shown in bold and starred messages have a star as you would see them in your Gmail account. This list is sorted by date in descending order and can be scrolled back and forth.



The screenshot shows a list of incoming emails. The first message is from 'Master Zone' with the subject 'New Comment from Subscriber'. The message is marked as unread. The interface includes a 'Gmail | Refresh' button, a search bar with 'rollbase.com', and navigation links for '1 - 50 of 799' and 'Older > Oldest >'. There are also 'From', 'Subject', and 'Date/Time' columns.

From	Subject	Date/Time
Master Zone	New Comment from Subscriber	11/11/2010 10:27 AM

Click on a message's subject to view details:

★ New question: "Fire trigger on send email"

« Back to pavel@rollbase.com

Communication Log Delete < Newer Older >

Message Details

From: Get Satisfaction! (Rollbase)

To: Robinson, Matt, William Spenser

Received: 17/02/2011 16:37

Subject: New question: "Fire trigger on send email"

Related To

Message Body

Matt Ireland just asked this question in Rollbase:

[Fire trigger on send email](#)

Hi Guys

From the Message View page you can:

- Star/Un-star the message by clicking on the star icon.
- Use the button in the upper right corner to create a Communication Log record from this message and attach that record to a selected Rollbase record (user, customer, etc.) or to record found by email address in incoming email message.
- Use the button in the upper right corner to delete the email message (move it to the Trash folder in your Gmail account).

Tip: Rollbase will try to match the email addresses in a message to the records in your Tenant. If a match is found, Rollbase will display a link to the matched record.

Tip: If a message in your inbox was originated by Rollbase and relates to a particular data record (such as an Invoice) Rollbase will resolve this and display a link to that record (see "Related To" field in the above screenshot). This is generally true even if the original message was replied or forwarded before landing in your mailbox.

## Outgoing Gmail

On the "Personal Settings" page (Setup > Personal Settings) you can provide credentials to your Google Apps account (individual or corporate) and select whether:

- Gmail (rather than Rollbase email server) should be used to send emails for this User.
- Google Calendar should be synchronized when Rollbase records with the Event or Task attribute (related to the current User) are created or updated.

Google Apps Settings

Google User Name	user@rollbase.com	Use Gmail	<input checked="" type="checkbox"/>
Google Password	*****	Synch Calendar	<input checked="" type="checkbox"/>
Confirm Password	*****		

Tip: These fields may not be on your page by default. In this case use "Edit Page" link to add "Google User Name" and other fields to the page.

Important: To use Gmail integration, beyond providing your credentials you also need to do the following setup in Gmail:

1. Login to your Gmail account
2. Click "Settings" then click "Forwarding and POP/IMAP" tab
3. In the IMAP Access section make sure "Enable IMAP" is selected.

Now if you provided the correct credentials and checked the "Use Gmail" checkbox, Rollbase will use Gmail to send emails on your behalf. This means that you will now see all Rollbase-generated emails in your "Sent Mail" folder. Also, when a recipient replies, Gmail will group reply message with your original message for convenience.

Important: Whenever you are sending an email using Gmail integration, Rollbase will display a reminder about it and give you a chance to test your Gmail connections to ensure that your stored credentials are up-to-date with your Google account.

## Google Spreadsheets

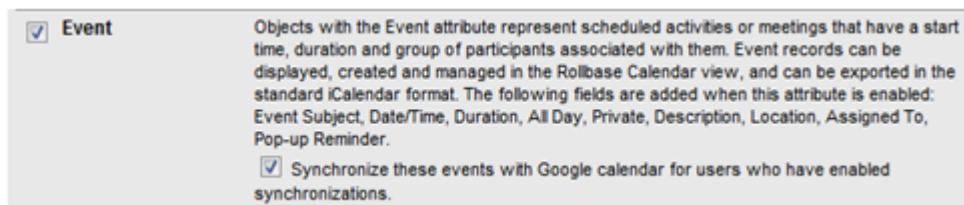
If you have provided valid credentials to your Google account, you will have an option to export your Views as Google Spreadsheets (along with XLS and CSV). Simply click on the "Google" link. All of the data in your View will be displayed as a new Spreadsheet in your Google Documents. In this way you can have all of the convenience of Google Spreadsheets to work with and analyze your data, including the ability to share your data with other users who may not necessarily have access to your Rollbase applications.

## Google Calendar

Events in the Rollbase calendar can be synchronized with a Google calendar. Events display in the Rollbase calendar for object records that have the **Event** attribute enabled. To synchronize events with a Google calendar requires enabling synchronization in the object definition and per user, in their personal settings.

With synchronization enabled, when an event is created or updated, the creator's Google calendar will be automatically synchronized. Events marked as **Private** will also be private on the Google calendar. You might need to refresh the Google calendar to view the result of synchronization.

To enable synchronization, the object definition must have the **Event** attribute and the checkbox must be selected for synchronization as shown below:



Each user that wants to see events from those objects needs to enable synchronization in their personal settings. They can do this by navigating to **Setup > Personal Setup > My Settings**. The **Synch Calendar** checkbox is in the **Google Apps Settings** section toward the bottom of the screen:

User: Adam Ministrator

**Contact Information**

First Name	Adam	Email Address	jtcarrollmysterywriter@gr...
Middle Name		Phone	5128286610
Last Name	Ministrator	Mobile Phone	
Job Title	Knowledge Architect	Fax	

**Login Name**

Login Name	myprogressid
------------	--------------

**User Preferences**

Rows Per Page		Do Not Animate Collapse <input type="checkbox"/>
---------------	--	--

**User Settings**

Language	English
Date Format	04/02/2014 01:03 PM
Time Zone	(GMT-05:00) Eastern Standard Time (EST) America/New_York
Email Footer	   
Email Encoding	ISO-8859-1

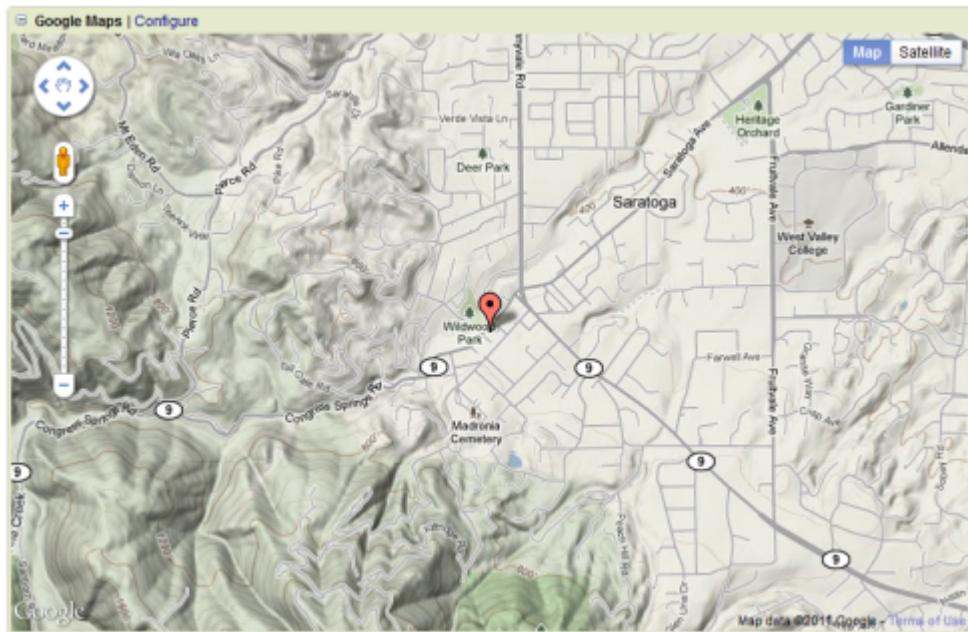
**Google Apps Settings**

Google User Name <input type="text" value="adm.admin1@gmail.com"/>	Google Password <input type="password" value="....."/>
	Confirm Password <input type="password" value="....."/>
Password must be at least 6 characters long.	
Use Gmail <input checked="" type="checkbox"/>	Synch Calendar <input checked="" type="checkbox"/>

## Google Maps

You can use Google Maps to visually represent a location associated with a record of an object with the "Location" attribute. A Google Map will automatically be rendered by a UI component which will be added to the View page by default (when you select the "Location" attribute) or by using the Page Editor.

The Google Maps component will render a map of the record's address as shown here:



You can configure the Google Maps component to select:

- Height of component in pixels (it will take 100% of available width by default)
- Default zoom level (can be changed at run time)
- Default map's type (street, satellite, etc.; can be changed at run time)

## Using SOAP or REST to Integrate with Rollbase

The topics in this section discuss how to access Rollbase application data programmatically for integration and extension purposes. When building integrations or external data manipulation tools that need to communicate with Rollbase programmatically, you can use the Rollbase SOAP or REST APIs. The SOAP and REST APIs expose the same functionality and use the same method names. A metadata API for both SOAP and REST provides methods for creating and manipulating definitions of applications, objects, fields, and relationships. The account under which metadata methods are invoked must have administrative privileges.

### SOAP API

Rollbase uses literal WSDL encoding. If your SOAP infrastructure does not support Literal WSDL encoding, consider using the REST API. To make SOAP calls, the client must have a valid Rollbase user account with login credentials. API users must have permission to view, create, update, or delete records to perform these actions via API calls. Permissions cannot be set via the API. They can only be set by an administrator using the Rollbase user interface in the Setup area.

The Rollbase SOAP API uses the same workflow mechanism as the standard Rollbase user interface. For instance, triggers designed to run on object record creation will run if a record is created through SOAP API. The Rollbase SOAP API uses the same permissions mechanism as the standard Rollbase user interface. Changes in triggers, views, etc. might not immediately be updated on the Web Services server. There might be some latency due to caching.

To establish a SOAP API session, call the `login()` method, which takes user name and password credentials. The `login()` call returns a session ID that must be used as the first parameter in all subsequent API calls. To end a SOAP API session, use the `logout()` method.

To protect Rollbase resources from overuse, the number of API calls is limited. The number of calls, or hits, increments over a 24-hour period for each Rollbase customer and then resets the next day. The actual number of API hits allowed is determined by your subscription. You can find this information in the **Subscription Details** page. Please contact Progress Rollbase Support if you need to increase this limit for any reason.

See [Rollbase SOAP Methods](#) on page 779 and [SOAP Metadata Methods](#) on page 706 for descriptions of the available SOAP methods.

## REST API

REST calls require authentication and are subject to the same security procedures as normal user log ins, including IP address whitelists and external authentication rules. To use this API requires a valid Rollbase user account with login credentials. The account must have permission to view, create, update, or delete records to perform these actions using REST calls. Permissions cannot be set via the API; they can only be set by an administrator logged into a Rollbase account.

Supply user credentials in one of the following ways:

- Using a session ID: Call the `login` AP with valid credentials to receive a session ID. Supply that session ID in every REST call as an HTTP header or URL parameter. At the end of session, call `logout` to end the REST session. For example, this PHP code sets the HTTP header:  
`header('sessionId: '.$sessionId);` This example passes the session ID in the URL:  
`&sessionId=1776eb2d56384f2d9d62f1bf83821b6d@5857`
- By providing basic HTTP authentication through the HTTP header. Append @ and and the customer ID to the login name. A PHP code example:

```
$header = base64_encode($userName.'@'.$custId.':'.$password);
header('Authorization: Basic '.$header);
```

Private cloud Master Server users with login permissions can use their credentials to call the REST API on a specified customer. REST API calls using a session ID or basic authentication are considered to be made from behalf of the logged in user. Permissions of that user are checked for each subsequent call. For instance, to update a record logged in user must have EDIT permissions on that record, of API call will fail. Use of a session ID is preferred in terms of performance and security.

Changes in triggers, views, etc. might not immediately affect the REST server. There might be some latency due to the caching mechanism.

See [Rollbase REST Methods](#) on page 732 and [REST Metadata Methods](#) on page 720 for descriptions of the available REST methods.

## Limits on API Calls

The number of API calls (hits) allowed in a period of time is limited, depending on your subscription. The system calculates the number of API calls (excluding login and logout calls) in 60-minute periods. If the number of calls during the current period exceeds the maximum limit, no new calls are allowed until the current period ends and a new counter starts.

## Monitoring SOAP Calls

The Rollbase **SOAP API** page allows you to see the activity that has been occurring with SOAP calls. View this page by navigating to **Setup > Applications Setup** and clicking **SOAP APIs**. This page provides the following information:

- **Runtime Info:**
  - Maximum number of hits per 60 minute time slot.
  - Link to a log file that has records about each call performed. To see detailed log records for every API call, go to the **Account Settings** page and verify that the **Enable API Log** box is checked.
- **WebAPI Server**
  - URL of the WebAPI Server.
  - Number of hits in the current time slot.
  - Start time of current time slot.
- **Development:**
  - A link to the WSDL describing the elements in an object definition.
  - A link to the XML schema for an object definition.
  - A link to this documentation.
  - A link to [the Code Generator](#).

## Monitoring REST Calls

The Rollbase **REST API** page allows you to see the activity that has been occurring with REST calls. View this page by navigating to **<uicontrol>Setup > Applications Setup</uicontrol>** and clicking **<uicontrol>REST APIs</uicontrol>**. This page provides the following information: <

It contains two sections:

- **Runtime Info:**
  - Maximum number of hits per 60 minute time slot.
  - Link to a log file that has records about each call performed. To see detailed log records for every API call, go to the **Account Settings** page and verify that the **Enable API Log** box is checked.
- **WebAPI Server**
  - URL of the WebAPI Server.
  - Number of hits in the current time slot.
  - Start time of current time slot.
- **Development:**

- A link to this documentation.
- A link to the [Code Generator](#).

---

## Security and Access Control

---

The Progress Rollbase infrastructure is hosted in a secure server environment that uses firewalls and other security technology to prevent interference or access from outside intruders. When you access the Rollbase service via HTTPS, Secure Socket Layer (SSL) technology protects your information using both server authentication and data encryption, ensuring that your data is safe, secure and available only to registered Users in your account. As long as your authentication information (username and password) are kept safe, your data remains inaccessible to unauthorized viewing and use.

You can allow Rollbase to handle authentication when users log on, or you can handle authentication with a system external to Rollbase. In addition, you can customize specific security settings and the Access Control Layer (ACL), each of which we will discuss in this chapter.

For details, see the following topics:

- [Supported Methods of Authenticating Users](#)
- [Built-in Security Levels](#)
- [Setting the Authentication Method](#)
- [Rollbase User Authentication](#)
- [External Authentication](#)
- [User Roles and Permissions](#)
- [Location/Department/Function Permissions](#)
- [Enabling an administrative user to log into a customer tenant](#)

# Supported Methods of Authenticating Users

Rollbase enables you to handle user authentication using the password authentication mechanism that is configured internally or using an external system that stores and authenticates users to access Rollbase.

Rollbase Private Cloud customers may implement their own custom authentication engine. The authentication mechanisms are categorized as:

- **Password Authentication:** This mechanism enables Rollbase to handle log in authentication. You can then use the built-in security levels and mechanisms described in [Security and Access Control](#) on page 375 and when a new user is created, configure Rollbase to generate temporary password and send it to the user's email accounts (by default, an email with "Welcome to Progress Rollbase" subject).

---

**Note:** The "Welcome to Progress Rollbase" email is based on a template included in default Rollbase application and can be modified as per your requirements. On first log in, the Rollbase user is requested to change the user password against the old/temporary password. There is no field or template token corresponding to user's password because Rollbase does not store actual users' passwords but an encrypted (salted and hashed) password instead.

---

- **External Authentication:** This mechanism enables Rollbase to external authentication mechanisms wherein user information is stored and authenticated by an external system. The following external authentication mechanism are supported in Rollbase:
  - **LDAP:** Progress Rollbase authenticates users based on LDAP
  - **HTTP Post:** Progress Rollbase authenticates users based on HTTP POST
  - **HTTP Get:** Progress Rollbase authenticates users based on HTTP GET
  - **OpenEdge:** Progress Rollbase authenticates OpenEdge AppServer users

The implementation of the different user authentication mechanisms are discussed in detailed in the following sections.

If you choose an external authentication mechanism, the parameters are treated as templates and accept the following tokens:

- `{!loginName}` for log in name entered by User
- `{!password}` for password entered by User
- `{!ipAddress}` for IP address used User who is trying to log in
- Any field token from USER object such as `{!lastName}`

# Built-in Security Levels

Rollbase supports three security levels per application, **Low**, **Medium**, and **High**. Private cloud customers can both configure security and add more levels if desired (see [securitylevel.xml](#) on page 503). The standard Rollbase levels and the restrictions they enforce are described in the table below:

Security Level	Low (default)	Medium	High
Password length (characters)	6+	8+	8+
Password is case-sensitive	No	Yes	Yes
Password can include sequential or repeating characters (like '123456' or 'aaaaaa')	Yes	No	No
Passwords must include non-alphabetical character	No	No	Yes
Block user account after N unsuccessful login attempts	Never	10	5
Duration of block	N/A	30 minutes	60 minutes
Minutes of inactivity before expiring user session	240 (4 hours)	240 (4 hours)	240 (4 hours)
Minutes of usage before forcing user to re-login	480 (8 hours)	480 (8 hours)	480 (8 hours)
Minutes to wait before expiring record lock	240 (4 hours)	60 (1 hour)	30 (1/2 hour)

## API Access

With **API Only Access** a user's credentials can only be used to access REST and Web API.

Note: Create regular User first, and then check "API Only Access". If you don't see this check box on User Edit page - use Page Editor to add this field to the page.

# Setting the Authentication Method

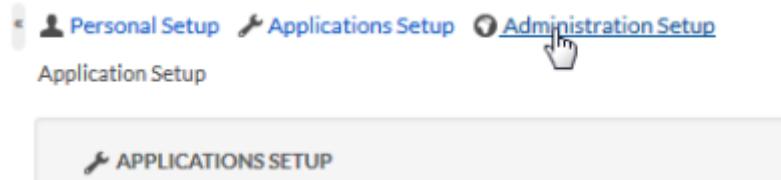
The following steps describe how to set authentication method when logged in as an administrator in a customer tenant:

---

**Note:** These authentication methods are only available for Rollbase Private Cloud users.

---

1. From the application menu in the banner, select **Setup Home**.
2. Click **Administration Setup**.



The **Administration Setup** page appears:

### ADMINISTRATION SETUP

- USERS**  
Create and manage user accounts
- ROLES**  
Create and manage user roles
- TRANSFER OWNERS**  
Transfer ownership of object records from one user to another
- USER ACCESS LOG**  
Recent user login history

### ACCOUNT ADMINISTRATION

- SETTINGS**  
Company-wide settings
- ACCOUNT SETTINGS**  
Manage your organization's account settings and customize page appearance
- CURRENCY CODES**  
Manage currency codes.
- EXCHANGE RATES**  
Manage currency exchange rates.

### SECURITY SETTINGS

- WHITELIST**  
Restrict login access to your account to a list of selected IP addresses.
- SUPPORT ACCESS**  
Grant login access to support personnel

### BACKUP AND MAINTENANCE

- BACKUP**  
Create and download a full backup of all your data.
- BATCH JOBS**  
Create and configure batch jobs to run periodically for maintenance and other purposes.
- GLOBAL TEXT SEARCH**  
Manage full-text search index, view related log files.

### RUNTIME STATUS

- SYSTEM JOBS**  
View running system jobs such as Spreadsheet imports
- PORTAL VISITORS ONLINE**  
View list of Authenticated Portal Visitors who are online now

### SYSTEM LOGS

- SYSTEM EVENTS LOG**  
System Events Log shows important customer-related logs, including records about finishing large imports of data.
- SYSTEM ERRORS LOG**  
View list of system and user-related Errors and Exceptions

### 3. In the **Security Settings** box, click **Authentication**.

The **Authentication types** page appears:

Authentication Type	Type	Description
	<input checked="" type="radio"/> <b>Password</b>	Progress Rollbase manages passwords and use them for Users' authentication
	<input type="radio"/> <b>LDAP</b>	Progress Rollbase performs LDAP call to verify validity of entered password
	<input type="radio"/> <b>LDAP Advanced</b>	Progress Rollbase performs LDAP Advanced call to verify validity of entered password
	<input type="radio"/> <b>HTTP POST</b>	Progress Rollbase performs HTTP POST call to verify validity of entered password
	<input type="radio"/> <b>HTTP GET</b>	Progress Rollbase performs HTTP GET call to verify validity of entered password
	<input type="radio"/> <b>OpenEdge</b>	Progress OpenEdge manages passwords(OpenEdge Single Point of Authentication)

4. Choose one of the following authentication types:
  - **Password**: For details, see [Supported Methods of Authenticating Users](#).
  - **LDAP**: For details, see [LDAP Authentication details](#) on page 380. This only supports authentication for a particular LDAP sub-tree. To authenticate users across multiple LDAP user groups, use **LDAP Advanced** authentication.
  - **LDAP Advanced**: For details, see [LDAP Advanced Authentication details](#) on page 381. This supports authentication across multiple user groups.
  - **HTTP Post**: For details, see [HTTP POST Authentication details](#) on page 383.
  - **HTTP Get**: For details, see [HTTP GET Authentication details](#) on page 384.
  - **OpenEdge**: For details, see [OpenEdge Authentication details](#) on page 384.
5. Click **Next**.
6. Under **Test External Authentication**, specify a valid **Login Name** and **Password**, and click **Test External Authentication** to check whether authentication succeeds using the type you choose. Note that you cannot save your changes until the test succeeds.
7. Click **Save** to apply the authentication type.

## LDAP Authentication details

If you choose **LDAP** as your authentication method while [Setting the Authentication Method](#) on page 378, specify the following values to configure your LDAP system to authenticate users to access Rollbase:

**Table 2:**

Field	Description
<b>Target URL</b>	URL to access the LDAP system (typically, <i>ldap://&lt;host-address&gt;</i> )
<b>SECURITY_AUTHENTICATION</b>	The authentication mechanism to implement. For example, for a Sun LDAP service provider, this can be one of the following strings: "none", "simple", sasl_mech, where sasl_mech is a space-separated list of SASL mechanism names. The default value for this field is "simple".
<b>SECURITY_PRINCIPAL</b>	Name of the user or program doing the authentication. Typically, a query string to search the LDAP database.
<b>SECURITY_CREDENTIALS</b>	Credentials of the user or program doing the authentication.
<b>Additional fields</b>	Any other additional details required to set up an LDAP call.

After specifying the above values, you must test your authentication. To test your authentication, see [6](#) on page 380.

## LDAP Advanced Authentication details

The **LDAP Advanced** authentication type supports authentication across multiple LDAP user groups. In contrast, the LDAP authentication type only works for users in a particular sub-tree. For example, an LDAP directory of employees that is divided into groups based on their location would require LDAP Advanced authentication.

If you choose **LDAP Advanced** as your authentication method while [Setting the Authentication Method](#) on page 378, specify the following values to configure your LDAP system to authenticate users to access Rollbase:

Field	Description
<b>Target URL</b>	URL to access the LDAP system (typically, <code>ldap://&lt;host-address&gt;</code> )
<b>Base Distinguished Name</b>	<p>The root distinguished name (DN) to use while running queries against your directory server. Example:</p> <ul style="list-style-type: none"> <li>• <code>o=example,c=com</code></li> <li>• <code>cn=users, dc=ad, dc=example,dc=com</code></li> <li>• For Microsoft Active Directory, specify the base DN in the following format: <code>dc=domain1,dc=local</code>. You will need to replace the <code>domain1</code> and <code>local</code> for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the LDAP structure of your server.</li> </ul>
<b>Additional User DN</b>	<p>The value to be used in addition to the base DN when searching for and loading users. If no value is supplied, the sub-tree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> <li>• <code>ou=Users</code></li> </ul> <p>If, for example, an LDAP directory has users as well as printers in it and you want to query only the users in the directory, then you can pass this additional filter in this field.</p>
<b>Search Mode</b>	<p>The LDAP authentication requirements to search for and get results from a search query. You can specify the following based on your LDAP configuration:</p> <ul style="list-style-type: none"> <li>• <b>Anonymous:</b> For LDAP directories that support queries from a source that is not logged in.</li> <li>• <b>Authentication:</b> Only authenticated users can query the LDAP directory. If you choose this option, you must specify the <b>Admin Security Principle Username</b> and <b>Admin Security Principle Password</b>.</li> </ul>

Field	Description
<b>Use Name Attribute</b>	The attribute field to use when loading the username. Example: <ul style="list-style-type: none"> <li>• cn</li> <li>• uid</li> </ul>
<b>Additional fields</b>	Any other additional details required to set up an LDAP call.

After specifying the above values, you must test your authentication. To test your authentication, see [6](#) on page 380.

## HTTP POST Authentication details

If you choose **HTTP POST** as your authentication method while [Setting the Authentication Method](#) on page 378, specify values for the following fields to set up a Post call::

**Table 3:**

Field	Description
<b>Target URL</b>	URL that identifies the HTTP device and port (typically, <i>http://device:port-number/...</i> )
<b>HTTP Body</b>	The template for body of HTTP POST request (typically SOAP call). This must include tokens for users' input.
<b>HTTP Headers</b>	Valid HTTP request headers to pass as part of the HTTP POST call. This is an optional field.  <b>Note:</b> The <b>Content-Type</b> header is available only for backward compatibility. You can leave the header value as blank to not include it as part of your HTTP POST call.
<b>Response Text</b>	Text that must be present in HTTP response to indicate if the authentication was successful (for example, <Authenticated>true</Authenticated>).

After specifying the above values, you must test your authentication. To test your authentication, see [6](#) on page 380.

## HTTP GET Authentication details

If you choose **HTTP GET** as your authentication method while [Setting the Authentication Method](#) on page 378, specify values for the following fields to set up a Get call:

**Table 4:**

Field	Description
<b>Target URL</b>	URL that identifies the HTTP device and port (typically, <i>http://device:port-number/...</i> )
<b>HTTP Headers</b>	Valid HTTP request headers to pass as part of the HTTP GET call. This is an optional field.
<b>Response Text</b>	Text that must be present in the HTTP response to indicate if the authentication was successful (for example, <Authenticated>true</Authenticated> ).

After specifying the above values, you must test your authentication. To test your authentication, see [6](#) on page 380.

## OpenEdge Authentication details

If you choose **OpenEdge** while [Setting the Authentication Method](#) on page 378, specify values for the following fields to implement OpenEdge Single Point of Authentication (SPA):

---

**Note:** You should understand how OpenEdge SPA works before configuring Rollbase to use this type of authentication. For more information on OpenEdge SPA, see the Progress OpenEdge AppServer Administration documentation.

---

Field	Description
<b>OpenEdge Realm URL</b>	The URL to connect users to the state-free AppServer. <b>orealm</b> is the name of the OpenEdge State-free AppServer where you deploy the OpenEdge realm. (typically, <code>appserver://host-name:port-number/orealm</code> ).
<b>OpenEdge Realm Class</b>	The realm service interface's class path. SPA security implementation for an OpenEdge REST Web application must specify the <b>HybridRealm</b> interface class.
<b>OpenEdge Domain</b>	The name of the domain that the OpenEdge user must belong to. Note that, only a single domain name can be specified and that only the users in that domain will be authenticated.
<b>OpenEdge Domain Access Code</b>	The code or a key required for a OpenEdge user to access the OpenEdge domain. In that, in a REST service call, this code seals the client principal token that validates and authenticates users.
<b>Realm Password Algorithm</b>	The format of the password passed to the realm service interface for validation.
<b>Realm Token File</b>	The file name that holds a serialized <b>ClientPrincipal</b> used to authenticate the realm service interface.
<b>CA Certificate Store File</b>	The security certificate from the certificate store required for user authentication.
	<p><b>Note:</b> If your Realm URL is secure and it requires certificate, you must provide your certificate in the <b>CA Certificate Store File</b> field for Rollbase to access the URL. For example, <code>AppserverDC://hostname/brokername</code> is not secure and doesn't require a Certificate, and <code>AppserverS://hostname/brokername</code> is secure and requires a certificate for access.</p>

After specifying the above values, you must test your authentication. To test your authentication, see [6](#) on page 380.

# Rollbase User Authentication

Rollbase provides each user in your organization with a unique user name and password that must be entered each time a user logs in. Rollbase issues a session "cookie" to record encrypted authentication information for the duration of a specific session. The session "cookie" does not include the password of the user. Rollbase does not use "cookies" to store other confidential user and session information, but instead implements more advanced security methods based on dynamic data and encoded session IDs. Additionally, Rollbase implements **HTTPOnly** cookies that direct browsers to expose the cookie only to HTTP and HTTPS requests.

Requirements for user passwords depend on the Security Level, as described in [Built-in Security Levels](#) on page 377.

When a new user account is created, the system automatically generates a new temporary password and sends this password in a welcome email to the user. This password must be changed during the first login. Neither the administrator nor Rollbase personnel have access to user passwords.

To login as a regular user, the user is directed to the Rollbase login Page:  
<https://www.rollbase.com/router/login/login.jsp>

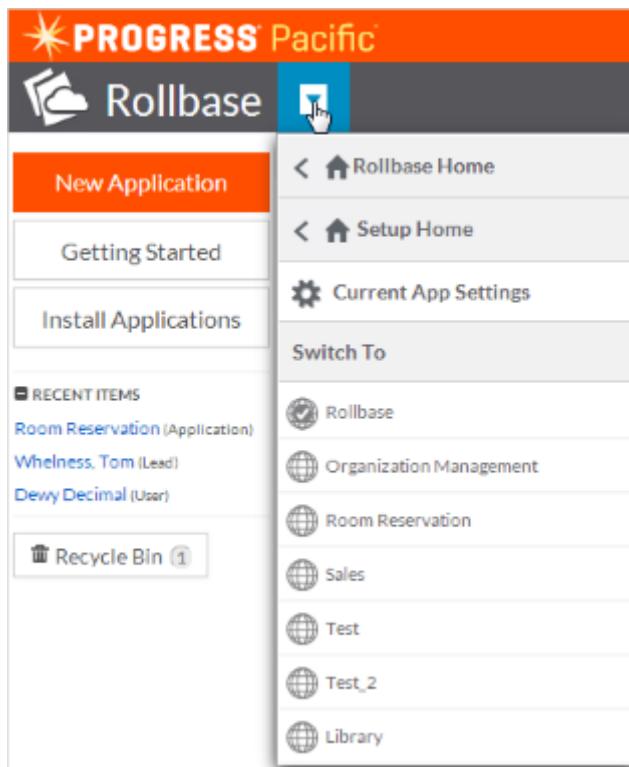
Warning: A user can have only one Rollbase session open at a time. If a user logs in again in a different browser Rollbase terminates any previously opened Rollbase sessions (the only exception to this is API sessions).

## Setting and Changing Security Levels

Security settings apply per customer tenant and control login session behavior and password requirements. [Built-in Security Levels](#) on page 377 describes the available levels.

To set and change security settings for your tenant, follow these steps:

1. From the header menu, select **Setup Home**.



2. In the **Administration Setup** section, click **Account Settings**.

The **Account Settings** page displays.

3. From the **Security Level** menu, select the desired level and optionally, change the **Expiration Policy** for passwords.

4. Click **Submit**.

A message will confirm an update to the settings.

## Enabling Single Click Log In

When Progress Rollbase sends emails with links to selected object records, these links can allow a single-click login under certain conditions. By clicking on that link, the user bypasses user name/password authentication and is taken directly to the record view page.

To enable single click log in, follow the steps below:

1. Prepare an email template with a "Link to View Page" token such as {!#LINK.\$APPROVAL}
2. Make sure the email recipient fields uniquely identify a single user. This is possible if you have:
  - No CC or BCC addresses.
  - A single TO address.

- No users with duplicate email addresses.

If these conditions are met, #LINK URL is replaced with a specially constructed URL which allows single-click login when you send the email message.

3. To increase security, further limitations apply:

- The single-click URL will expire after 30 days.
- The URL can be used only once.

If any of the above conditions is not met, the user will be asked to provide a user name and password on the regular login page.

## Password Expiration Policies

You can set a Password Expiration policy for your organization using the Account Administration page (see [Advanced Setup and Administration](#) on page 421). By setting the Expiration Policy field to the number of days before password expiration, user passwords will expire after that number of days. Progress Rollbase will prompt the user to enter a new password during the next login attempt.

Tip: Leave the Expiration Policy field blank to disable the password expiration policy.

You can enable email notification prior to password expiration by editing the "Send Password Expiration Notification" trigger in the User object definition page. Check the "Deployed" box and select the number of days after the date of the last password update to send the notification message. The email template for that message can be customized the same as any other email template.

Tip: You can create more than one notification trigger if desired.

## Custom Validation Rules

In addition to validation rules based on Security Level as described in [Security and Access Control](#) on page 375 you can define custom rule to ensure that users' passwords satisfy your security policy. On "Authentication" page enter body of JavaScript function (with single input parameter password) which returns error message for invalid password or NULL for valid.

The following example ensures that user's password includes at least one special character '@' or '#':

```
if (password.indexOf('@')<0 && password.indexOf('#')<0)
    return "Password must include special character";
```

## Whitelist IP Addresses

As a security precaution you can restrict login to a list of IP addresses. The IP address of any user trying to log in to your customer tenant will be checked against this list. If the IP address does not match, the log in will be denied. Use the **Whitelist** option and specify IP addresses in one of the following ways:

- The exact address in x.x.x.x format
- A group of addresses (use \* for the common part of the address)
- A host name to be resolved into an IP address

Whitelist of IP Addresses: MyCompany

Save Cancel

Whitelist of IP Addresses

Your current IP Address: 131.239.31.2

IP Addresses

153.345.21.340  
143.\*.21.10  
www.mysterywriters.com

Examples of whitelist items:

- 1.2.3.4 numeric IP address
- 1.\*.3.4 masked IP address
- www.mycompany.com host name to be resolved in IP address

You can limit whitelist control to a group of selected roles. If you wish to apply the whitelist to all roles, do not make any selection.

## Security Questions for Authentication

Often security provided by password-based login is insufficient for customers, especially in the financial sector. In this case an administrator can set up Security Questions. After providing correct login names and passwords, users must answer randomly selected question(s) before proceeding to application pages.

Security questions will be asked:

- After login on computer without secure cookie
- When user's password was reset
- When user has changed answers to security questions
- When password has expired (if expiration policy was setup)
- If secure cookie was deleted or expired

To set up Security Questions check "Use Security Questions to authenticate users" on Password Authentication page. Then select:

- Number of questions user must answer in his/her profile
- Number of questions user must answer to be authenticated

- Up to 12 security questions

Security Questions **NEW**

Use Security Questions to authenticate users

User must answer **2** questions in his/her profile

User must answer **1** previously answered questions to be authenticated

Please list of security questions used for authentication. Do not change previously answered questions since this may confuse users.

1. What is name of your pet?
2. Your mother's maiden name?
- 3.

Warning: Do not change wording of existing questions (unless it's fixing of typo) - users who already have answered these questions will be confused.

Users can select questions they want to answer and give actual answers on Personal Setup > Change Password page.

Security Questions

**Question 1** Your phone number?

**Answer 1** 555-555-5555

**Question 2** Maiden name?

**Answer 2**

If a user's profile does not contain answers to security questions, these answers will be collected on first user's login.

After that, the user will be prompted to answer to randomly selected security question(s) after entering correct login names and passwords on the login page.

Please Answer Security Questions

**Question 1** Your phone number?

**Answer 1**

Remember this computer and skip security questions next time I login

Warning: do not use this option on public computers

If questions were answered correctly and "Remember this computer" box checked, the system will set a security cookie on user's computer. Next time, if this cookie is present the system will bypass security questions.

Warning: Users should only use "Remember this computer" option on their personal machines.

## Forgotten Password

If a user forgets his or her password, he or she can use the "Forgot Password" page. User should enter a user name and email address (as specified in valid Rollbase user account). If that info matches Rollbase records, the Rollbase password will be reset and a new temporary password will be sent to user's email address. User will be asked to change password at first login.

---

**Note:** A similar procedure is available for Portal Users (see [Rollbase Portals](#) on page 257).

---

## Portal Security

Rollbase portals can use the HTTPS protocol if you select this option for a particular portal on the Portal Edit page.

Portal pages can use Portal Users' authentication via portal Login pages. For information on Portals, see [Rollbase Portals](#) on page 257.

When you select the Portal User attribute on an object definition, Rollbase creates a login name and password fields to enable the visitor to login to the Portal. The password field allows two settings:

- Minimum password length.
- Password must contain both letters and digits.

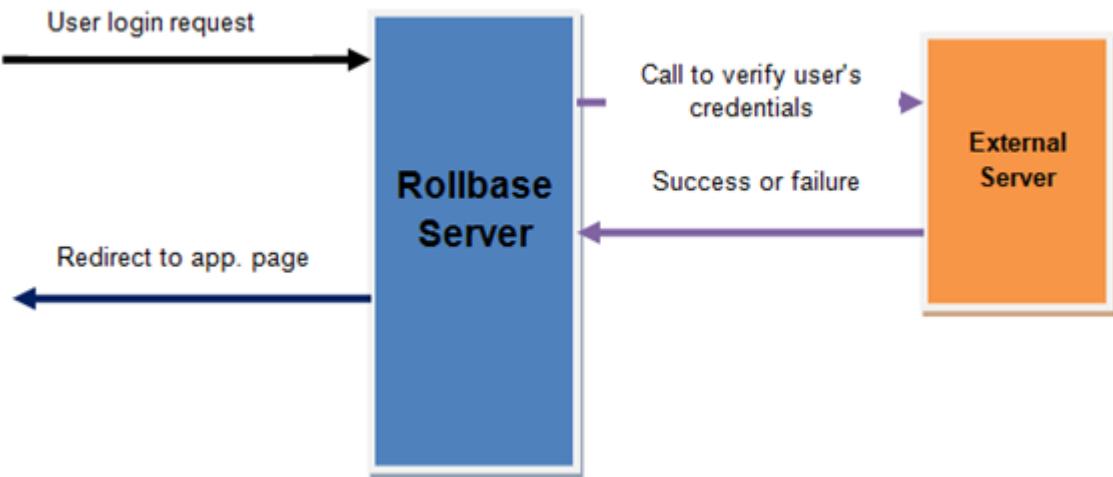
You can include template token corresponding to the password field into email templates. In this way a Visitor can be notified by email about password change. However this field token works only immediately after the change. Later actual password's value is not available for viewing or retrieving through API. This is done as security precaution.

A Portal User enters login name and password on Portal login page. A Visitor can reset the password by clicking "Forget Password" link. In this case the system will generate a new password and send email notification to stored visitor's email address.

Note: Visitor can only login if "Is Active" box is checked (this box is checked by default).

## External Authentication

Instead of verifying password stored in Rollbase database the system can perform external call to verify user's password on existing Rollbase user account. The following picture illustrates the process:



**Note:** If external authentication is set up Rollbase is no longer managing users' passwords. In this case "Change My Password" link is disabled. We strongly recommend that you modify an email template which welcomes new users and clearly indicate selected authentication method.

## External System Single Sign On Example

Consider the following example of Single Sign-On configuration:

- External system has set of user accounts synchronized with Rollbase instance.
- External system uses HTTP session IDs created during user login.
- Logged in user of External System should be able to access Rollbase instance without entering login name and password.

To do that first create a link on External System page:

`https://{!hostName}/router/servlet/Router?act=login&loginName={!userName}&password={!sessionId}`

where:

`{!hostName}` - host name of Rollbase installation (example: `www.rollbase.com`)  
`{!userName}` - user name of Rollbase user account (presumably shared with External System)  
`{!sessionId}` - Session ID of the Rollbase instance

Please note that although URL uses parameter "password" actual password is securely stored in External System and not exposed.

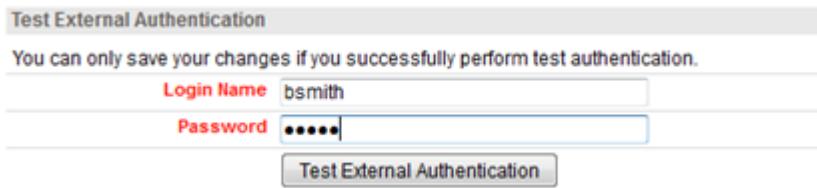
Next configure Rollbase Customer to use external authentication through an HTTP GET or HTTP POST request. That request will include a user name and session Id supplied in URL above. The External System must verify that both values are valid and correspond to each other.

If these conditions are met, upon clicking constructed URL, the user of the External System will access the Rollbase instance without a need to enter a password, which remains securely stored in External System.

## Verifying External Authentication

Please be very careful when configuring external authentication, since if it is configured incorrectly, no user will be able to login. With that in mind you can only save your changes for external authentication if you were able successfully perform test authentication.

Enter a sample user name and password on the bottom of page and click "Test External Authentication". Pop-up window will display call details and result similar to Debugger window used for HTTP triggers. If the call was successful, the "Save" button will be enabled.



The screenshot shows a dialog box titled "Test External Authentication". It contains a message: "You can only save your changes if you successfully perform test authentication." Below this is a "Login Name" field with the value "bsmith" and a "Password" field with the value "\*\*\*\*\*". At the bottom is a "Test External Authentication" button.

## User Roles and Permissions

Role is a central concept in Rollbase security. Each user is assigned one role. User Roles define access privileges.

Note: In addition to role-based permissions, Rollbase allows individual permissions to be assigned directly to particular users.

Progress Rollbase checks permissions at the following times:

- When displaying Applications and Menus available to the current user.
- When displaying a list of records in a View or Chart. If the user does not have access to certain records (because of relationship-based permissions or LDF), they will not be shown.
- When displaying a Page to View or edit a particular record. If the user is trying to access a record without authorization, Rollbase displays an "Access Denied" error message.
- When presenting a list of records to create relationships (either in a popup window or a drop-down list).
- When displaying search results.

Note: When displaying links to related records, Rollbase does not check permissions for the current user. Permissions will be checked, however, if the user tries to drill down on a related record by clicking on the link.

Rollbase initially creates three system roles for you:

- Administrator: A user with full access to all Objects and all customization features.
- Portal User: Assigned to authenticated users of external-facing Portals.
- Portal Guest: Assigned to non-authenticated users of external-facing Portals. Unlike Portal users, Portal Guests cannot log into a portal. Portal Guests can only access a public portal page.

- Server API: Used to check permissions in Query API calls when the current user is not authorized to perform an operation.
- AJAX API: Used to check permissions in AJAX API calls when the current user is not authorized to perform an operation.

Tip: User-created Roles can be added to Applications and published along with other Application components, such as Objects. Publishing a Role will include permissions assigned to these Roles.

Important: in addition to Role-based permissions, you can assign individual permissions by user. Click on a user's link anywhere it is available in the setup screens, click the Permissions tab and set the permissions as desired.

## User Roles

The following permissions can be assigned by Role:

- Permissions to access selected Applications from the Apps section on the Rollbase sidebar.
- Administrative permissions (See list below)

Administrative Permission	Granted by Default
Manage (create, modify and delete) Views	No
Manage (create, modify and delete) Charts	No
Manage (create, modify and delete) Reports	No
Manage Currency exchange rates	No
Send individual and mass emails	Yes
Use "Print", "PDF", and Spreadsheet Export links	Yes
Convert records to different Object type	Yes
Manage (create, modify and delete) Import Maps	Yes

Warning: If a particular Role does not have access to at least one Application, a user with that Role will not be able to log in to Rollbase.

Note: If a particular Role has access to an Application, that Role will be given default access to newly created components of that Application unless you override this access.

Note: Merge and Convert functionality also require permissions to create a new record of selected Object type. 1. Object permissions: View, create, edit and delete permissions per Object. In addition, you can assign Role-based permissions to access Views, Charts, Reports and Workflow actions to the Role. The list of Objects includes system entities that may be used in reports, such as Activity Trails, Comments and Login History records. 2. Menus (including sub-menus) access permissions.

Note: A user has all permissions assigned to a user's assigned Role. In addition, you can assign individual permissions to a particular user. Use the Permissions section on the User's View page (Setup > Administrative Setup > Users) to assign individual permissions.

The detailed setting of all these permissions can be a tedious task, but it gives you full control over user access to all data in Rollbase. To simplify navigation, the Rollbase UI helps you:

- Filter Objects and menus by Application.

- Select a full set of permissions per Application.

- Select all or none of the lists of sub-items for Objects and menus.

## User Hierarchy of Permissions

Permissions through relationships are only granted to the users with direct relationship to the record. However, in many business cases, users are in hierarchical relationships, such as manager-subordinate. In these situations it is unusual to give access to lower levels of the hierarchy without providing access to users higher up in the reporting structure.

To solve this issue, Rollbase provides a "Hierarchy of Users" relationship: "Direct Reports" (multiple) and "Reports To" (single). The following shows an example Second Admin user who reports to the First Admin user:

User: Second Admin

**Save** **Save & New**

### Contact Information

First Name

Middle Name

Last Name

Job Title

### Login and Role

Login Name

User Role

### Reporting Structure

Reports To    
 [First Admin](#)

Rollbase calculates a sub-tree of users who report (directly or indirectly) to the current user. All relationship-based permissions given to that sub-tree are also delegated to the current user.

Consider the following user hierarchy:

- Joe Recruiter
- Mike Sancilardi (reports to Joe)
- Taras Bulba (reports to Mike)

There are three orders in the system with different owners:

Orders 1-3 of 3					
Action	Order	Buyer	Order Total	Owner	Updated At
<input type="checkbox"/> Edit   Del	O-2010-00005	Hardware Supply	\$2,000.00	Mike Sancillardi	08/04/2010 08:34 AM
<input type="checkbox"/> Edit   Del	O-2010-00001	Yamaguchi Supply	\$5,600.00	Simon Esguerra	08/04/2010 08:34 AM
<input type="checkbox"/> Edit   Del	O-2010-00002	Olympia Design	\$41,700.00	Taras Bulba	08/04/2010 08:33 AM
				\$49,300.00	

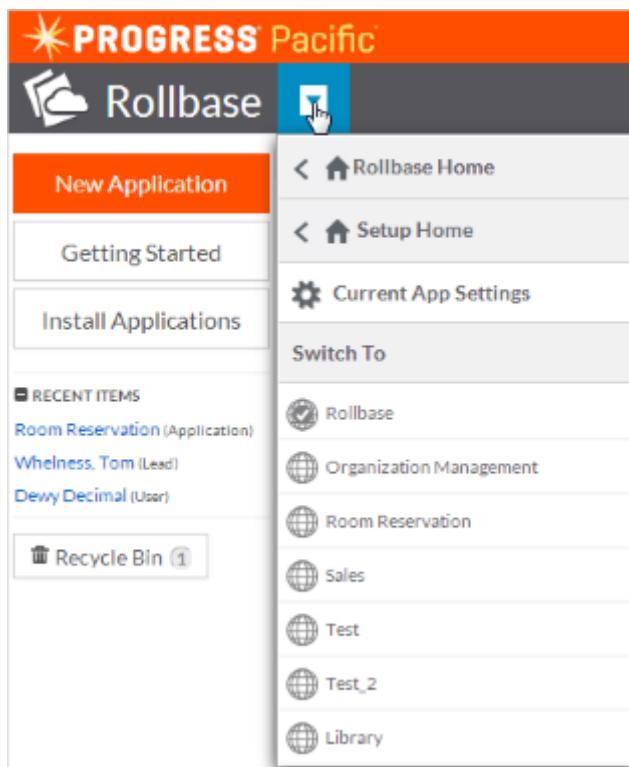
After logging in, Joe Recruiter will see two orders (even if he does not own them directly). Access to these orders is granted through ownership of direct and indirect users below him in the hierarchy):

Orders 1-2 of 2					
Action	Order	Buyer	Order Total	Owner	Updated At
<input type="checkbox"/> Edit   Del	O-2010-00005	Hardware Supply	\$2,000.00	Mike Sancillardi	08/03/2010 07:04 PM
<input type="checkbox"/> Edit   Del	O-2010-00002	Olympia Design	\$41,700.00	Taras Bulba	08/03/2010 07:03 PM
				\$43,700.00	

## Managing User Roles

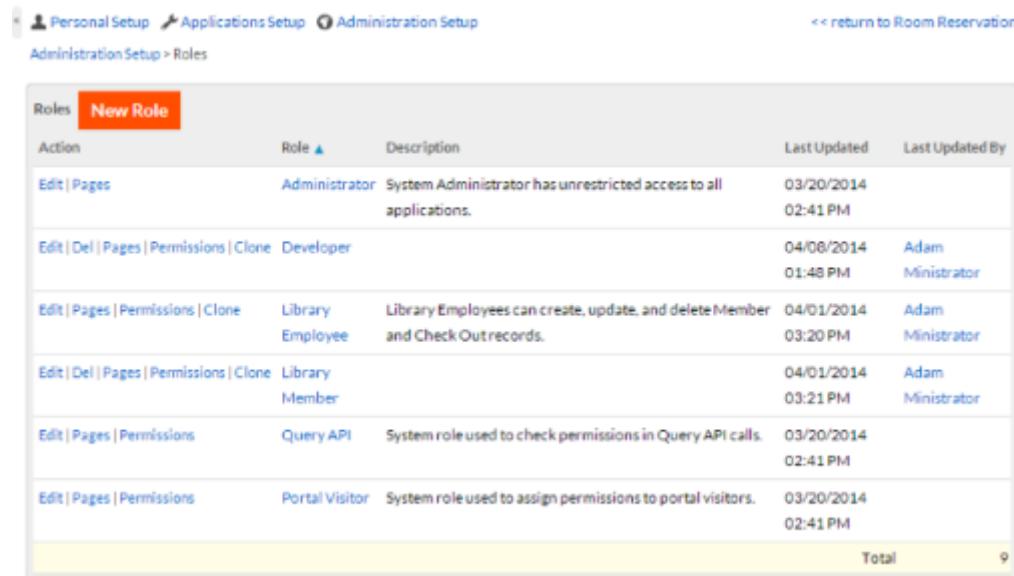
You can add a new role or edit an existing role as follows:

1. From the header menu, select **Setup Home**.



2. In the Administration Setup section, click **Roles**.

The list of currently defined roles displays:

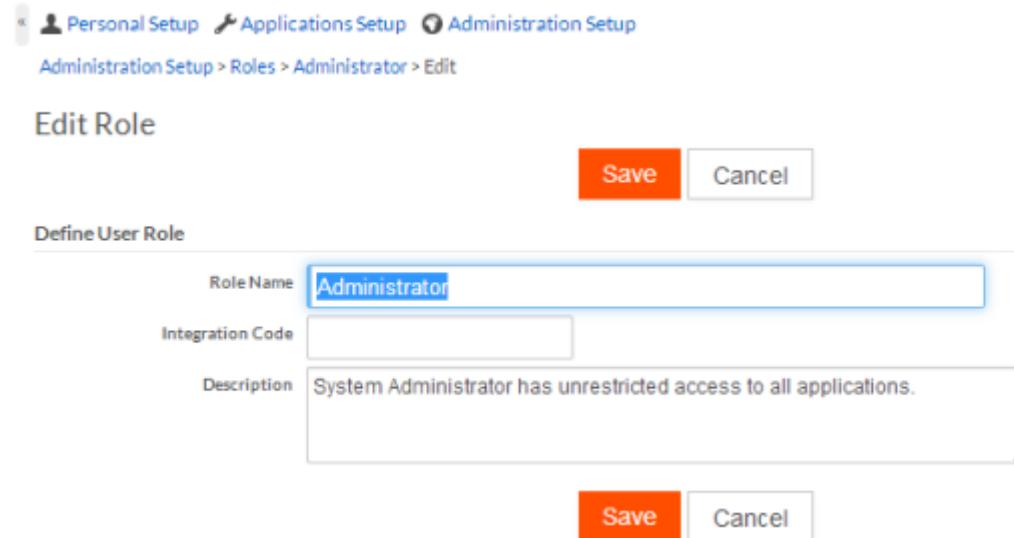


Role	New Role			
Action	Role	Description	Last Updated	Last Updated By
Edit   Pages	Administrator	System Administrator has unrestricted access to all applications.	03/20/2014 02:41 PM	
Edit   Del   Pages   Permissions   Clone	Developer		04/08/2014 01:48 PM	Adam Ministrator
Edit   Pages   Permissions   Clone	Library Employee	Library Employees can create, update, and delete Member and Check Out records.	04/01/2014 03:20 PM	Adam Ministrator
Edit   Del   Pages   Permissions   Clone	Library Member		04/01/2014 03:21 PM	Adam Ministrator
Edit   Pages   Permissions	Query API	System role used to check permissions in Query API calls.	03/20/2014 02:41 PM	
Edit   Pages   Permissions	Portal Visitor	System role used to assign permissions to portal visitors.	03/20/2014 02:41 PM	
Total				9

3. Perform one of the following:

- To define a new role, click **New Role**.
- To change an existing role, click **Edit**.
- To view a role, click the role name.

The following screen shows the default **Administrator** role.



Define User Role

Role Name	Administrator
Integration Code	
Description	System Administrator has unrestricted access to all applications.

Save Cancel

Save Cancel

4. The available fields include:

- A name for display purposes.
- An optional **Integration Code**. Specify a value to use this role in formulas and triggers.
- A description of the role.

## Private Attribute

Calendar Events and Tasks, as well as Reports and Templates, provide an Is Private attribute that introduces additional visibility limitations:

- Private reports are only visible to their creator and administrative users.
- Private email and document templates are only visible to their creator and administrative users.
- Private events and tasks are only visible to their creator, users with direct relationships (assigned to) and administrative users.

Note: The Private attribute for Events and Tasks applies atop of all other View permissions (described in more detail below). It does not substitute them.

## Component-level Permissions

Although you can set all Role-based permissions in one place, you can also set them on an individual Object, Menu, View, Report, or Workflow Action. The Setup pages to view and edit these components include a Permissions section.

**Note:** Limiting access to Views can be useful for limiting user access to certain kinds of records, provided that the user does not have access to creating or editing Views. However, we do not recommend this approach. Using relationship-based permissions (see section 8.2 below) or Location, Department and Function (LDF) based permissions is more secure and reliable.

On an Object Definition, you can set view, create, edit and delete permissions:

Location/Department/Function Permissions: NO				
User Roles				
Administrator	✓ View	✓ Create	✓ Edit	✓ Delete
Developer	✓ View	✓ Create	✓ Edit	✓ Delete
Library Employee	✓ View	✓ Create	✓ Edit	✓ Delete
Library Member	✓ View	Create	Edit	Delete
Query API	✓ View	Create	Edit	Delete
Record Creator	✓ View		✓ Edit	✓ Delete
Portal Visitor	✓ View	Create	Edit	Delete
Individual Users				

On a Menu, View, Report, or Workflow Action component, you can set a single View (i.e. whether or not this component can be accessed) permission:

Assigned To	Titles	New title	Search	Import	Templates
<b>User Roles</b>					
Administrator	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access
Developer	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access
Library Employee	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access
Library Member	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access
Query API	Access	✓ Access	Access	✓ Access	Access
Portal Visitor	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access
<b>Individual Users</b>					
Dewy Decimal	✓ Access	✓ Access	✓ Access	✓ Access	✓ Access

## Permissions Through Relationships

You can use relationships between a User or Portal User and records to give that user access to related records only, rather than to all records of that type.

You can assign Permissions through Relationships when editing Object-related permissions (or use the Permissions link on the Relationships section):

Relationships between Visitor and User					
Users   Select All   Select None		<input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit <input type="checkbox"/> Delete			

Relationships between Item and Visitor <small>NEW</small>					
Visitors   Select All   Select None		<input checked="" type="checkbox"/> View <input checked="" type="checkbox"/> Edit <input type="checkbox"/> Delete			

Consider the following example: You want to limit the access of users in the "Account Manager" Role to Order records. There is a one-to-many relationship between User (called "Owner" for the sake of this relationship) and Order. To achieve this, give following permissions:

- To "Account Manager" Role: Create only
- To "Owner" relationship: View, Edit and Delete

Tip: For dependent records, such as Order Line Items in the example above, use Role-based permissions. This strategy works because the user can only access these dependent records through the parent record, access to which is controlled through the relationship.

## Field Level Permissions

You can also limit access to specific Fields for certain non-administrator Roles. On any object settings page, navigate to the **Fields** table and click **Permissions**:

Action	Field Label ▲	Data Type	Integration Name	Def. Value	Text Index	Tr
Edit   Del   Convert   Clone   Validation   Events   Permissions	Author	Text	author			✓
Edit   Validation   Events   Permissions	Check Out	Lookup (Check Out)	R53367			
Edit   Events   Permissions	Comments	Text Area	comment			
Edit   Permissions	Created At	Date/Time	createdAt			
Edit   Permissions	Created By	User Link	createdBy			
Edit   Permissions	ID	Integer	id			
Edit   Del   Convert   Clone   Validation   Events   Permissions	Publisher	Text	publisher			
Edit   Permissions	Tags	Search Tag	tag			
Edit   Validation   Events   Permissions	Title	Record Name	name			✓

From here, you can hide selected fields from pages and views by un-checking, or show fields but make them read-only.

**Note:** The Read-Only option is only available for visible Fields.

Field Level Permissions			
Role	View	Read Only	
Developer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Library Employee	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Library Member	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Titles | Users | Reports | Comments | Library Members | Check Outs

## Record Creator Pseudo Role

Consider the following example: Visitors to your "Customer Service" Portal can create Support Request records, as well as browse and manage a list of their own requests. To achieve this access, set up the following permissions on the Support Request Object:

- To "Portal User" Role: Create only
- To "Record Creator" pseudo-role: View, Edit and Delete

Record Creator   <a href="#">Select All</a>   <a href="#">Select None</a>	<input checked="" type="checkbox"/> View	<input checked="" type="checkbox"/> Edit	<input checked="" type="checkbox"/> Delete
Portal Visitor   <a href="#">Select All</a>   <a href="#">Select None</a>	<input type="checkbox"/> View	<input checked="" type="checkbox"/> Create	<input type="checkbox"/> Edit

## Page Versions

When you create an object Definition, Rollbase creates a complete set of pages for viewing, editing, and creating records for that object. You can create different versions of these pages using the **Clone** link in the **Pages** table. Use the **Page Editor** to modify these cloned pages.

Pages				
Action	Page Name	Page Type	Last Updated	Last Updated By
View	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Sync</a>   <a href="#">Properties</a>	View title	ViewTitle [title_lb]	06/18/2014 05:26 PM
Edit	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Sync</a>   <a href="#">Properties</a>	Edit title	Edit Title [title_lb]	06/18/2014 05:26 PM
New   <a href="#">Menu: Newtitle</a>	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Sync</a>   <a href="#">Properties</a>	New title	New Title [title_lb]	06/19/2014 11:20 AM
Search Results	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	Search Results	Search Results Title [title_lb]	06/18/2014 05:26 PM
Status	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	Status Change	Status Title [title_lb]	06/18/2014 05:26 PM
Selector	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	titles Selector List	Selector Title [title_lb]	06/18/2014 05:26 PM
Mass Update	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	Mass Update	Mass Update Title [title_lb]	06/18/2014 05:26 PM
Quick Create	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	Quick Create	Quick Create Title [title_lb]	06/18/2014 05:26 PM
Records List   <a href="#">Menu: Titles</a>	<a href="#">View</a>   <a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Properties</a>	titles List	Records List Title [title_lb]	06/18/2014 05:26 PM

Later, you can assign these cloned pages to be used by individual users or those with particular roles. Use the Pages link in the **Roles** list or on the User View Page to assign pages by Role and User:

<a href="#">Personal Setup</a>	<a href="#">Applications Setup</a>	<a href="#">Administration Setup</a>	<a href="#">&lt;&lt; return to Room Reservation</a>
Administration Setup > Roles			
<a href="#">Roles</a> <a href="#">New Role</a>			
Action	Role	Description	Last Updated
<a href="#">Edit</a>   <a href="#">Pages</a>	Administrator	System Administrator has unrestricted access to all applications.	03/20/2014 02:41 PM
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Pages</a>   <a href="#">Permissions</a>   <a href="#">Clone</a>	Developer		04/08/2014 01:48 PM
<a href="#">Edit</a>   <a href="#">Pages</a>   <a href="#">Permissions</a>   <a href="#">Clone</a>	Library Employee	Library Employees can create, update, and delete Member and Check Out records.	04/01/2014 03:20 PM
<a href="#">Edit</a>   <a href="#">Del</a>   <a href="#">Pages</a>   <a href="#">Permissions</a>   <a href="#">Clone</a>	Library Member		04/01/2014 03:21 PM
<a href="#">Edit</a>   <a href="#">Pages</a>   <a href="#">Permissions</a>	Query API	System role used to check permissions in Query API calls.	03/20/2014 02:41 PM
<a href="#">Edit</a>   <a href="#">Pages</a>   <a href="#">Permissions</a>	Portal Visitor	System role used to assign permissions to portal visitors.	03/20/2014 02:41 PM
Total			9

On the **Assign Page Versions** screen, expand the object you want to assign pages for and then select the pages you want to assign for the Role or User. As a result, different Users or Roles can use different pages to access the same records. This approach can be used to limit a user's access to certain object fields and/or to personalize a the experience by User and Role.

Assign Page Versions: Library Employee

**Save** **Cancel**

Assign Page Versions

[Expand All](#) | [Collapse All](#)

Account

Case

CatalogItem

Check Out

View Check Out	View Check Out
Edit Check Out	Edit Check Out
New Check Out	New Check Out
Search Results	Search Results
Status Change	Status Change
Check Outs Selector List	Check Outs Selector List
Mass Update	Mass Update
Quick Create	Quick Create
Check Outs List	Check Outs List

Client

## Location/Department/Function Permissions

Location/Department/Function (LDF) permissions are the most complex Rollbase permissions, used to cater to the security needs of larger organizations with more complex internal structures.

Note: LDF is essentially a filter that is applied before Role-based and relationship-based permissions discussed above. Use of LDF-based permissions requires explicit understanding of this mechanism.

Note: System object USER by default has LDF permissions. If you do not wish to support LDF permissions edit your USER object and un-check "Organization" attribute.

Important: Adding LDF permissions to large number of objects may affect application's performance. We do not recommend adding LDF permissions to dependent objects (such as document like items) which are accessible only through master object (document).

To use LDF permissions, install the Organization Management standard Application if it was not automatically installed when you created your Rollbase account. This Application installs four Objects (with tabs):

- Locations
- Departments

- Functions
- Groups

Location, Department and Function represent three dimensions with a tree-like internal structure. When you install the Organization Management Application, these Objects come pre-populated with a set of base records:

You can interpret LDF dimensions in any way that matches your organization's needs. For example, on any View Page for LDF dimension records, you can add a list of records of a certain type related to the current LDF record (e.g., a list of Users at a current Department).

Check the Organization attribute for the Object on which you wish to enable LDF-based permissions:

<input checked="" type="checkbox"/> Organization	Objects with the Organization attribute have additional lookup fields for associating records with a Location, Department and Function in your organization. If selected, attributes will be used for LDF-based access control.
--	---

Then you can assign any or all LDF records to particular records of your Object:

## LDF Field Default Values

We recommend that you only use the Organization attribute on your object definitions when it makes good business sense to do so. For instance, an Invoice object may have Organization attribute and be subjected to LDF access control, while Invoice Line Item may not (assuming they are only accessible through the parent Invoice record).

Assigning LDF attributes for all records can be a tedious task. To simplify it please keep in mind the following rules for assigning LDF attributes by default for new records:

- LDF lookup fields may use default LDF values assigned to the current user if this option is checked on the lookup Field's Edit page for that object:

Use user's Location value as default for new records

- If a new related record is created and LDF fields are absent from the New record page, the LDF values from the parent record will be used by default.

## LDF Groups

The Organization Management Application also contains a Group Object that allows you to define combinations of LDF values as groups and assign one or more groups to users to create sophisticated organization-based permissions structures.

First, define an LDF group by giving it a name and assigning any or all of the LDF attributes:

Group Information

Group	Recruiters
Location	<input type="text"/>
Department	<input type="text"/> Management
Function	<input type="text"/>

Next, assign individual users to the group and/or assign (attach) groups to a particular user. The example below shows a user who is a member of three groups:

Groups 1-3 of 3				
Action	Group	Location	Department	Function
<input type="checkbox"/> Edit   Del   Detach	Recruiters		Management	
<input type="checkbox"/> Edit   Del   Detach	Taskforce Omega	San Francisco, CA - Headquarters		VP Human Resources
<input type="checkbox"/> Edit   Del   Detach	Washington Office	Washington, D.C.		

Each group has zero to three assigned LDF dimensions. This means members of that group have access to all LDF records (regardless of their Object type) that match these dimensions (root node or any node below).

Warning: A non-Administrator user who does not belong to any group will have no access to any LDF records.

Note: Within each group, attributes are connected by a logical AND. Groups themselves are connected by a logical OR. The special read-only Field named "LDF Filter" shows the exact access criteria for a particular user or LDF record.

Tip: If you create a group without any LDF attributes, members of that group will have full access to all LDF records. Only users in that group or an administrator can access a record with no assigned LDF attributes.

The following table summarizes access granted to a user, related to the example showing him as a member of three groups:

Group	Access
Recruiters	Location: ANY: Management (or below) Function: ANY
Taskforce Omega	Location: Washington, D.C. (or below) Department: ANY Function: ANY
Washington Office	Location: San Francisco, CA - Headquarters (or below) Department: ANY Function: VP Human Resources (or below)

## Setting Up LDF

From a mathematical perspective, each User Group defines a sub-set in a three-dimensional set of all LDF nodes that exist in a Customer. From this perspective, one Group can consume another (if the root of the second group lies within first Group) and two groups can be merged (if they define the same and only one, dimension such as Location). Because of this, the "LDF Filter" can show fewer groups than those of which a particular user is a member.

The user Object always has an Organization attribute, so you can assign LDF attributes directly to users. This does not grant any LDF permissions, as permissions need to be granted through Groups. Instead, these attributes can be used as the default when a particular user creates other LDF records. To enable this default behavior, check "Use user's Location value as default for new records" for the Object's Location lookup Field (similarly for its Department and Function lookup fields).

**Note:** If you create a new LDF record as a child of another LDF record (e.g., Invoice Line Item) and do not assign any LDF attributes to that new record, it will inherit LDF attributes from the parent record (e.g. Invoice).

The following steps summarize actions needed to use LDF functionality:

1. Install the Organization Management Application.
2. Create records in the Location, Department and Function trees, as needed.
3. Enable Organization attributes on desired Objects.
4. Assign LDF attributes to your records.
5. Define Groups for your users and assign LDF attributes to them.
6. Use the "LDF Filter" Field to verify assigned permissions.

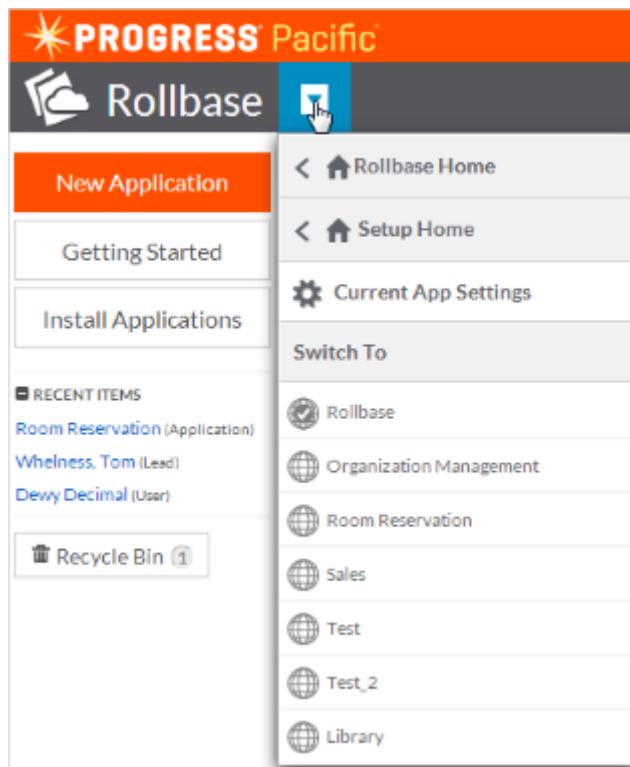
From this point on, Rollbase automatically checks LDF permissions to all LDF Objects and all users (except those with the Administrator Role) before checking all regular permissions (i.e. role-based, user-based and relationships-based).

# Enabling an administrative user to log into a customer tenant

The administrator of a customer tenant can allow master tenant users to access its environment for a specified duration. Typically, if a customer tenant requires that the master users troubleshoot a problem in its account, the customer tenant gives access to the master users for them to rectify the problem. After getting the access, the master user can access the customer details page and log into the customer tenant. For more information about how master users log into a customer tenant, see [Working with Customer Records](#) on page 471.

To enable a master tenant to log into a customer tenant:

1. From the customer tenant environment, as an administrative user, select **Setup Home** from the list of applications:



2. Under the **Administration Setup** options, select **Support Access**.

The **Support Access** page appears.

3. Click **Edit** to modify the support access settings in your tenant.
4. Select **Enable** to allow master users to access and log into all the users in your customer tenant.
5. Specify, in the **Give Access for** field, the duration for which you want to grant the access.
6. Click **Save**.

---

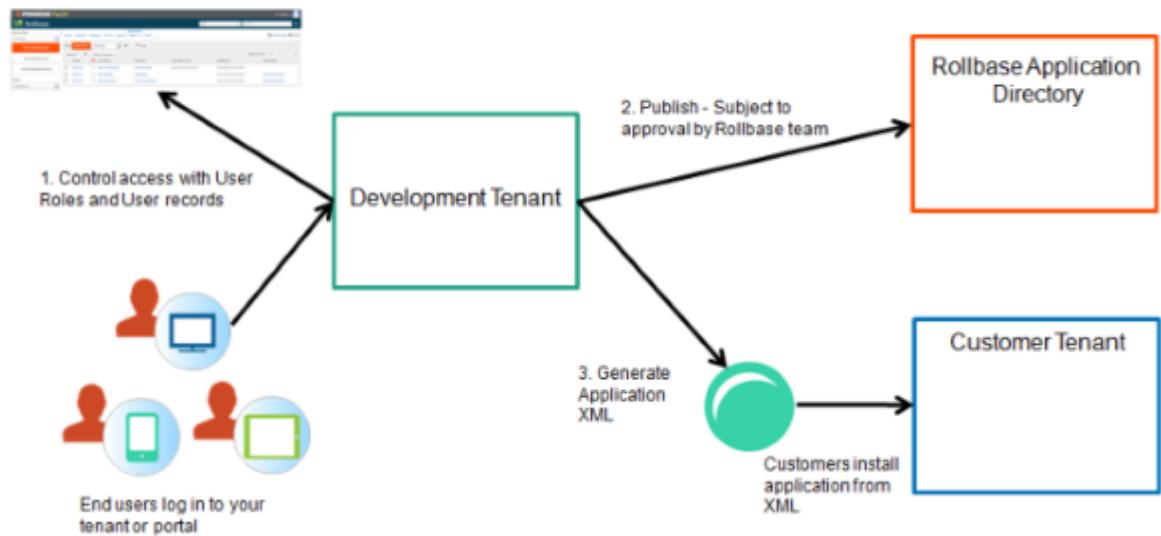
## Publishing and Distributing Applications

---

As illustrated in the following graphic, Rollbase provides three ways to distribute applications to end users:

1. Within your own tenant, by assigning specific users or user roles the appropriate application permissions. See [User Roles and Permissions](#) on page 393.
2. Publish the application in the Rollbase **Application Directory**, which makes it freely available to other Rollbase users.
3. Generate an XML version of the application, which you can distribute to any Rollbase tenant.

## Rollbase Application Publishing Options



[Rollbase User Authentication](#) on page 386 describes the first. The latter two methods, generating XML and publishing, use the same mechanism, XML serialization. Topics in this section cover publishing and XML generation.

For details, see the following topics:

- [Design and Development Considerations for Distributing as XML](#)
- [Administrative Management of Published Applications](#)
- [Generating Application XML](#)
- [Publishing to the Rollbase Application Directory](#)
- [Troubleshooting Published Applications](#)

# Design and Development Considerations for Distributing as XML

A Rollbase application XML file includes the definitions of all components associated with the application, such as objects, tabs, roles, hosted files, batch jobs and seed records. It is important to understand which components are included in an application when you publish it or generate XML.

You will want to track changes as you incrementally add new features or modify application functionality. Rollbase automatically increments the application version number each time you publish or generate XML.

The topics in this section describe application distribution options and best practices.

## Components Included in an Application XML File

Applications include core and dependent objects:

- When you create a new object for an application, by default it becomes a core object. Core objects are published and installed with applications.
- Dependent objects are pre-requisites, so, they must be installed in a tenant before the application depending on them is installed.

Relationships and conversion maps are only published with an application if both the source and destination objects are included in that application, either as core or dependent objects.

You can explicitly assign dependent objects to an application. Rollbase automatically adds dependent objects to maintain application integrity in the following cases:

- The **User** object is a dependent object for all applications. If a **User Role** is given application permissions, the permissions assigned to that role are published as part of the application XML. When the application is installed in another Customer, the role and permissions are installed.
- The **Approval** object will be added as a dependent object if any core objects have the **Approval** attribute.
- The **Communication Log** object will be added as a dependent object if any core objects have the **Contact** attribute.
- Objects not included as core objects that are related to objects included on application tabs will be added as dependent objects.

**Note:** The dependent objects that Rollbase adds to the application cannot be removed from the application.

The application view lists objects assigned explicitly and those included implicitly through tabs. Explicitly assigned objects have a **Remove** action link. Implicitly assigned objects can be removed only after removing the corresponding tab. For example, if there is an application that has Accounts and Contacts tabs, and the two tabs are related and the application only explicitly assigns an Account detail object. When this application is published, it will include the following objects: Account, Contact and Account Detail (without a Tab).

The following table summarizes the conditions under which components are included in the Rollbase application XML file.

Component	Included During Publication if:
Menu and sub-menus	Owning object (if any) belongs to the package
Relationship definition	Objects on both sides of the relationship belong to the package
List	View object belongs to the package
Conversion Map	Objects on both sides of the conversion belong to the package
Report	Object belongs to the package

Component	Included During Publication if:
Chart	Objects used in X and Y axis belong to the package
Gauge	Object belongs to the package
Related List	Parent object belongs to the package
Report Link	Report belongs to an object in the package
Lookup Field	Lookup object belongs to the package
Related Field	Related object belongs to the package
User Role	Assigned to the application
Portal	Assigned to the application
Hosted File	Assigned to the application
Batch Job	Assigned to the application
MobileApps	MobileApps belong to the package

## Use of Original IDs

When you distribute an application to be installed into other Rollbase tenants, Rollbase creates new IDs for all application components. Rollbase uses these local IDs to manage updates. Each application component and subcomponent has an ID that is unique within a tenant, and a unique **Original ID**, which is guaranteed to be unique across all tenants.

You should design your application to use original IDs and tokens to rather than the local ID. For example:

- Use integration codes for picklist items and workflow statuses in formula fields, templates and triggers; do not use IDs.
- Use template tokens such as **Link to View Page** `{!#LINK.order}` or a reference to the id, such as `{!id}`. Do not explicitly use IDs in Template Fields or Integration Link Fields (used to build dynamic URLs).

The **System Information** section of component definitions contains the local ID and the **Original ID**. For example, to find a component's **Original ID**, follow these steps:

1. From the **Applications** menu, select **Setup**.
2. From the **Applications Setup** section, click the type of component of interest. You will find subcomponents such as fields, templates, and relationships under their parent object.
3. Click the name of the component. For object subcomponents, scroll down to the appropriate section and click the subcomponent name.

The component definition displays. The **System Information** section contains the two IDs:

Object Properties	
Singular Name	Settings
Plural Name	Settings
Record Name	ObjectName
Record Name Template	
Integration Name	\$SETTINGS
Default Process	
Applications	Progress Rollbase
Description	Company-wide settings
Object Attributes	
Contact	Location
Event	Task
Document	Organization
Workflow	Portal Visitor
Approval	Survey
Survey Taker	Queue
Multi-Currency	Lockable
System Information	
Last Updated By	Created By
Last Updated At	04/23/2014 03:49 PM
ID	6484
Original ID	3251186
Is System	<input checked="" type="checkbox"/>

## Locking Applications

To protect your intellectual property, you can lock part or all of applications you create. This prevents others who install the application from changing it and guarantees that future updates will safely overwrite existing components. Applications, objects, and portals installed from locked applications can only be published and republished from the tenant in which they were originally developed. If an application is unlocked or partially locked, those installing it can selectively choose components to install or update.

Application lock options include:

- **Unlocked:** Administrative users that install the application can modify all components. Progress recommends this option if you do not intend to provide updates to this application, or if it is critical that your users be able to modify every aspect of the application.
- **Partially Locked:** Administrative users that install the application can modify all objects, menus, and portals except those that are marked as locked. Object subcomponents, such as fields and triggers can be locked independently, leaving the parent object unlocked. We recommend selecting this option if you plan to provide updates to specific components of your app while letting your users modify other components as needed.
- **Locked:** Administrative users that install the application cannot modify anything. We recommend using this option if you need to maintain complete control over all aspects of the application from update to update, and want to prevent your users from modifying it.

The only way to modify a locked component is to log in to the original tenant where it was created as super-admin from the Master system (by using the **Login** button from a Customer View page). Once installed in another tenant, an application's lock status can only be modified if the publisher modifies it and publishes an update.

## Attaching Seed Records

You can attach object records to your application as seed records. These records will be published and installed as part of the application. Once seed records are installed, they will not be updated during application updates. Seed records work best for testing and demonstration purposes. Because you attach records one at a time, this mechanism is not appropriate for distributing complete data sets.

To attach seed records:

1. Navigate to the application settings.
2. Scroll down to the **Seed Records** section.
3. Click **Attach Records**.

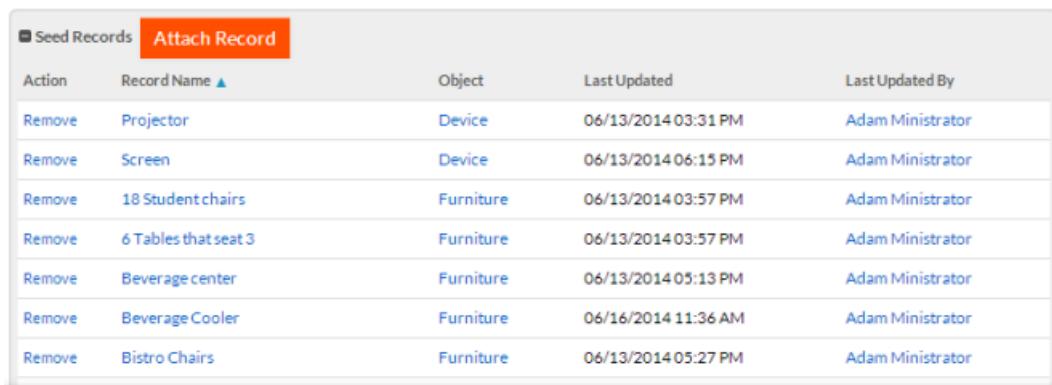
The **Record Selector** window displays.

4. From the **Attach Record** drop-down list, select the type of object you want to attach.
5. Click each record you want to add. You can select another type of object without closing the window.

The records will be added to the application if they are not attached already.

6. When you finish adding records, close the **Record Selector** window.

The seed records will display.



Action	Record Name	Object	Last Updated	Last Updated By
Remove	Projector	Device	06/13/2014 03:31 PM	Adam Ministrator
Remove	Screen	Device	06/13/2014 06:15 PM	Adam Ministrator
Remove	18 Student chairs	Furniture	06/13/2014 03:57 PM	Adam Ministrator
Remove	6 Tables that seat 3	Furniture	06/13/2014 03:57 PM	Adam Ministrator
Remove	Beverage center	Furniture	06/13/2014 05:13 PM	Adam Ministrator
Remove	Beverage Cooler	Furniture	06/16/2014 11:36 AM	Adam Ministrator
Remove	Bistro Chairs	Furniture	06/13/2014 05:27 PM	Adam Ministrator

**Note:** If a **System Settings** object is attached to the application, an **Attach Settings** button will be available to add the settings as a seed record.

## Providing a Test Drive

The **Application Directory** for hosted and private cloud installations includes **Test Drive** capability. With test drive enabled, users can preview application functionality in read-only mode without logging in. When users test drive an application, they access it with the **Portal User** role. In **Test Drive** mode, users cannot create or modify data records.

To enable **Test Drive** for an application, you must first create a new tenant where that application is installed. Once you have created a new tenant and installed the application successfully, edit the application definition and set permissions for the **Portal User** user role to access tabs and objects. Make sure the application contains some meaningful sample data.

When the application is ready, if you are using hosted Rollbase, notify Rollbase Support of the customer tenant that contains the **Test Drive**. If you are using your own Private Cloud installation, edit the **Published Application** record and assign that customer as the **Test Drive** tenant.

## Testing and Verifying Application Correctness

Before publishing your application, you can generate an [Application Tree](#) to verify that your application does not contain any errors. Due to the mechanics of the publishing and installation process, little information other than the component's internal ID is available to report errors when they occur. For this reason, the **Application Tree** Component allows you to more easily pinpoint and fix problems before they occur by showing components with errors in red.

Application errors typically occur when a certain component has been deleted, but other components still have a reference to it. For example, a trigger may have a reference to a deleted conversion map. The application should not be published until these issues are fixed. Error messages may contain numeric IDs of missing application elements. See [Troubleshooting Published Applications](#) on page 417 for suggestions on fixing common errors.

# Administrative Management of Published Applications

Rollbase Administrators can manage published applications from the Master Server (see [Administration](#) on page 465). The **Published Apps** tab in the **System Console** provides controls for updating, deleting, and performing other actions on published applications.

Because the published application object is itself a Rollbase object, Rollbase Master Server administrators can define a custom approval process for allowing published applications to appear in the **Application Directory**. By default, this is simply done by checking the **Approved** checkbox field.

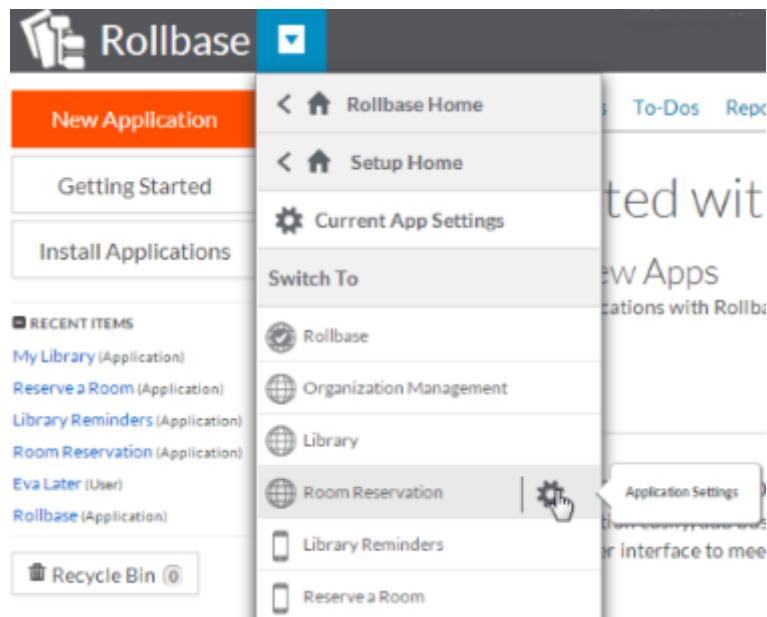
Rollbase Master Server administrators can also push new versions of an application to customers who have previously installed that application. See [Pushing Application Updates to Other Tenants](#) on page 555.

## Generating Application XML

Those who develop Rollbase applications to sell or for use by other departments in their organization can distribute the applications as XML files that can be installed in any Rollbase tenant.

To generate an application XML file, follow these steps:

1. From the header menu:
  - To navigate to settings for the current application, click **Current App Settings**.
  - To navigate to settings for a different application, select the application and click its **Application Settings** link.



2. From the **More Actions** menu, select **Generate XML**.
3. Verify that the version number is correct.
4. Select a **Lock Status**. If you select **Partially Locked**, you need to specify which application components are locked by checking the boxes next to those components. Expand the tree to specify locking at a fine-grained level. In the following example, the **Account** object is locked, the **Case** object is not.

**Application Tree**

- CRM | Select All | Select None ↴
  - Dependent Objects ↴
    - User
    - Objects ↴
      - Account ↴
        - + Fields ↴
        - + Relationships ↴
        - + Views ↴
        - + Reports ↴
        - + Charts ↴
      - Case ↴
        - + Fields ↴
        - + Statuses ↴
        - + Relationships ↴
        - + Views ↴
        - + Reports ↴
        - + Charts ↴
        - + Actions ↴
        - + Processes ↴

5. Click **Generate XML**.
6. Store the XML file locally and verify that it has an **.xml** extension. Some versions of Internet Explorer ignore the XML content type directive, so you may see garbled XML in a popup window. This does not indicate a problem. If this happens choose the **Page > Save As...** menu option and save the XML file to disk.

The file is ready to distribute.

## Publishing to the Rollbase Application Directory

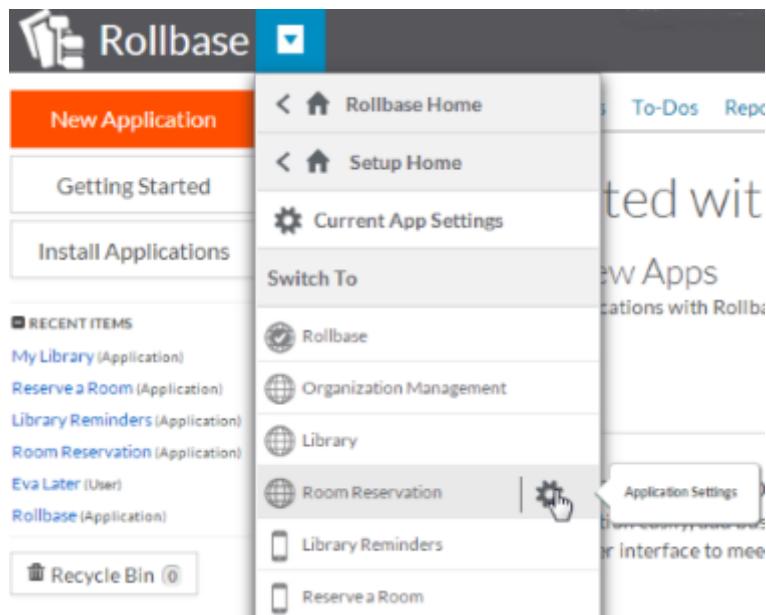
Publishing an application to the **Application Directory** makes it freely available for installation in any Rollbase tenant. By default, Rollbase personnel must approve submitted applications before they appear in the **Application Directory**. Private Cloud administrators can change this behavior as needed by modifying the **Published Application** object in the master tenant.

If you have a Rollbase hosted tenant and want your application to be made available to selected customers instead of all Rollbase users, you can become an ISV partner (see [Setup and Administration for ISVs](#) on page 545).

If you continue development of an application that has already been published, you can publish updates to that application. Each updated application will have a higher version number than the last and customers who installed previous versions will be able to upgrade to the latest version. Private Cloud Master system administrators can also push updates of an application to all tenants who have it installed. See [Pushing Application Updates to Other Tenants](#) on page 555.

1. From the header menu:

- To navigate to settings for the current application, click **Current App Settings**.
- To navigate to settings for a different application, select the application and click its **Application Settings** link.



2. Click **Publish** or **Publish Updates**. The button name depends on whether the application has been published previously.

The **Publish Application** dialog displays.

Application Version Number

If you would like users who have already installed this application to be able to install any updates you publish, you must increase the version number here.

Version  Previous value: 1

Update Application

Application Name

Category

Category (2nd)

Category (3rd)

Tags

Application Description

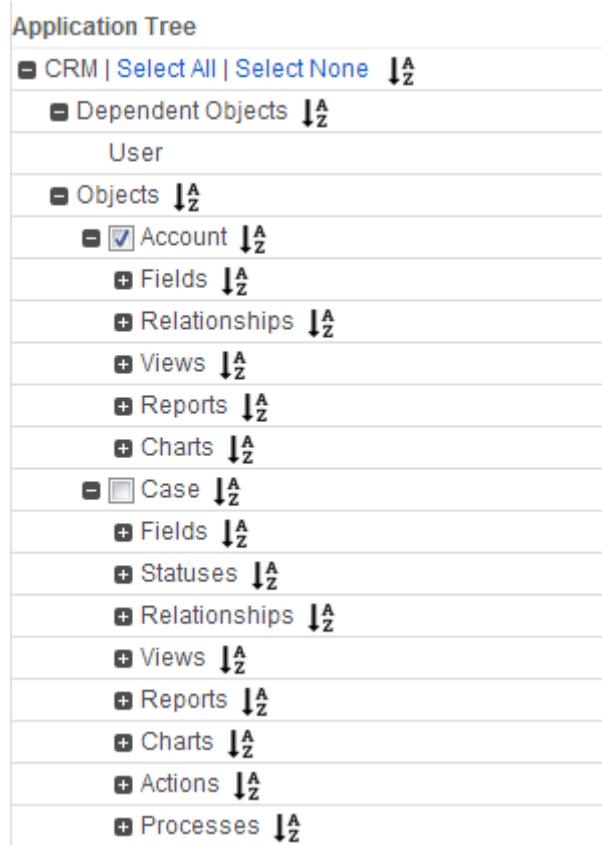
The CRM application includes all the core functionality required of a basic customer relationship management system by providing a central place for all of your sales, marketing, and support representatives to access and manage customer Accounts, Contacts, Leads, Cases and Opportunities. Also included in the CRM application is a Leads portal which you can embed into your website to accept lead submissions from visitors to your website. Configure as many different leads portals and pages as you want to manage all of your marketing campaigns. Lead records can easily be converted into Account records as needed by using standard conversion mapping functionality. Feel free to add your own Case portal to allow web submission of Cases. As with any other Progress Rollbase application, CRM can be fully customized and extended to meet the needs of your organization. For

Path: p

Company Logo (60x60)  No file chosen  
(512 KB max)

Lock Status  Unlocked (users can modify all application's components)  
 Partially Locked (users can modify selected application's components). Check boxes below to lock components.  
 Fully Locked (users cannot modify any application's components)

3. Verify that the version number is correct.
4. Select a category for the application, this will be displayed in the **Application Directory**.
5. Enter a description. Those browsing for applications will read this, so make it as descriptive as possible.
6. Optionally, upload a logo graphic.
7. Select a **Lock Status**. If you select **Partially Locked**, you need to specify which application components are locked by clicking the check boxes next to those components. Expand the tree to specify locking at a fine-grained level. In the following example the **Account** object will be locked in the published application; the **Case** object will not be locked.



8. Click **Submit**.

If you submitted to the Rollbase hosted **Application Directory**, you will receive notification on the status of your application.

## Troubleshooting Published Applications

The following table lists some application errors and suggestions on how to fix them. If an application is locked, the original designer must make the fixes and regenerate the application. If an application is unlocked, those who installed the application into their own tenant can take corrective action. If you encounter any of the errors listed below, please ask the publisher to fix them or attempt to fix them yourself before contacting Rollbase Support.

Component	Error	Fix
Object with Workflow Attribute	Default status <i>NNN</i> not found	Edit the object and set a default status
Relationship	Related object definition <i>NNN</i> not found	Delete the relationship or add the missing object definition to the application
List View	List view <i>NNN</i> not found View column <i>NNN</i> not found	Delete, re-create or edit and save the list view

Component	Error	Fix
Portal	Web page <i>NNN</i> not found	Edit the portal and set its main page
Menu	Object definition <i>NNN</i> not found Parent menu <i>NNN</i> not found Page definition <i>NNN</i> not found	Delete or re-create the object, menu, or page
Report	Object definition <i>NNN</i> not found Relationship definition <i>NNN</i> not found Report column <i>NNN</i> not found	Edit and save or delete this report
Chart	Object definition <i>NNN</i> not found  x-axis field <i>NNN</i> not found y-axis field <i>NNN</i> not found	Edit and save or delete this chart
Conversion Map	Object definition <i>NNN</i> not found	Delete this conversion map or add the destination object definition to the application
Import Map	Object definition <i>NNN</i> not found	Delete this import map
Page Cell	Field definition <i>NNN</i> not found	Delete this page cell by editing the page and removing it (this often requires just opening the page in the page editor and re-saving the page)
List Component	View <i>NNN</i> not found	Use the <b>Page Editor</b> to select the list component and set its default view
List Component	Page <i>NNN</i> not found	Use the <b>Page Editor</b> to select the list component and set its destination view and edit page
Related List	Relationship Definition <i>NNN</i> not found	Delete this related list
Chart Cell	Chart <i>NNN</i> not found	Use the <b>Page Editor</b> to select the chart component and set its default chart
Portal Link Cell	Page <i>NNN</i> not found	Use the <b>Page Editor</b> to remove this portal cell link and replace it with a valid one

Component	Error	Fix
Report Link Cell	Report <i>NNN</i> not found	Use the <b>Page Editor</b> to delete this report link from the page
Portal Submission Log in Form	Page <i>NNN</i> not found	Use the <b>Page Editor</b> to select the form component and choose a valid destination Page
Grid Control	Relationship Definition <i>NNN</i> not found	Configure or delete the grid
Related Field	Relationship Definition <i>NNN</i> not found	Delete this related field
Lookup Field	Relationship Definition <i>NNN</i> not found	Edit and save the lookup field
Template File Field	Template <i>NNN</i> not found	Edit or delete this template file field
Dependent Picklist Field	Field Definition <i>NNN</i> not found	Edit or delete this dependent picklist field
Trigger	Field Definition <i>NNN</i> Not Found Data Map <i>NNN</i> not found Status <i>NNN</i> not found Template <i>NNN</i> not found Workflow Status <i>NNN</i> not found Relationship Definition <i>NNN</i> not found	Edit or delete this trigger - some elements it relies upon are missing.
New Record Trigger	Target Object with id <i>NNN</i> is not included into this application	This trigger creates an object which is not included into this application either as a core or a dependent object
Workflow Process	Workflow Status <i>NNN</i> Not Defined Workflow action <i>NNN</i> not defined	Edit this process and make sure that the default is set, or delete the process and re-create it
Workflow Action	Workflow Status <i>NNN</i> Not Defined	Edit this action and fix the <b>Change Status To</b> setting

Component	Error	Fix
Workflow Action	Page <i>NNN</i> not found Status <i>NNN</i> not found Relationship Definition <i>NNN</i> not found Conversion Map <i>NNN</i> not found Template <i>NNN</i> not found	Edit this action and choose the correct parameters or delete and re-create it
Batch Job	Report <i>NNN</i> Not found Object Definition <i>NNN</i> not found	Attach the required object definition to the application or remove the batch job

## **Advanced Setup and Administration**

---

This topics in this section explain setup and administrative features not covered in other topics. These include personal setup, administration setup, localization and support services. It also covers the following critical areas in detail: transfer ownership of records, company-wide settings, localization, and batch jobs.

For details, see the following topics:

- [Personal Setup](#)
- [Administration Setup](#)
- [Backup and Restore](#)
- [Batch Jobs](#)
- [Date Formats](#)
- [Currency Formats](#)
- [Billing and Support Settings](#)
- [Language Support](#)
- [Global Text Search](#)
- [Support](#)

## Personal Setup

Personal Setup is the only setup feature available for non-administrator users, allowing them to change their personal settings and preferences, accessible through the Personal Setup Page (Setup > Personal Setup).

An administrator user can customize the Personal Settings Page that may (and should) look different than the User Edit Page.

The second Personal Setup option available to all users is the "Change My Password" Page.

To change their passwords, users must type in the old password and new password (twice). The new password must conform to password requirements selected for the current Customer. For information about security and access control, see [Security and Access Control](#) on page 375.

The following Personal Setup Fields are available:

- **Role:** User Role. Only an administrator user can change this field.
- **Rows Per Page:** Default number of rows per page in list views
- **Location, Department, Function:** These values will be copied by default to new records with the LDF attribute created by this user.
- **Reports To, Direct Reports:** These fields define the user's hierarchy that may be used for Access Control. For information about security and access control, see [Security and Access Control](#) on page 375.
- **Approver:** Check if this user is allowed to be an approver (see [Approvals](#) on page 200).
- **Language:** User's selection for one of the languages available in this Tenant.
- **Date Format:** User's selection for date format.
- **Time Zone:** User's selection for time zone. If different users use different time zones, Rollbase will re-calculate date/time field values to match the user's time zone before they are displayed.
- **Email settings:** Allows personalization of email messages sent by this user, such as a header, footer, etc.
- **Google account:** Users can store credentials to their Google email account which enable them to use Gmail, Google Calendar and Google Spreadsheets seamlessly with Rollbase apps. For information about integrating Google accounts, see [Integrating with Google Applications](#) on page 367.

## Administration Setup

The **Administration Setup** Page allows administrator users to manage run-time activities not related to Application development (Setup > Administration Setup).

Settings on the **Administration Setup** page apply per tenant. This Page contains the following sections:

- User Administration: Allows you to manage System-Wide Settings (see 3.1 below) as well as Users and their Roles (also available from the "Users" tab in the default Rollbase Application).

This section also contains a link to the Transfer Owners feature that allows you to move records from one user or users to another.

- Account Administration: Allows you to Account Settings and Company-wide settings. Also manages currency codes and exchange rates for your company.
- Security: Allows you to manage whitelist of IP addresses, users' Authentication method, and the ability to allow support personnel to access a customer tenant.
- Backup: Allows you to manage system backup files.
- Batch Jobs: Allows you to manage scheduled jobs for email reports, automating backups via FTP and automating export of a specific subset of data via FTP.
- Runtime Statuses: System Jobs: Displays run-time information about large jobs (such as a spreadsheet import) running in asynchronous mode.
- Portal Users Online: Displays a full listing of all currently active portal user sessions.
- System Events Log: Shows important customer-related logs, including information about application installation and deletion, large data imports, etc.
- System Errors Log: Shows all the Java exceptions (except those caused by a formula error) in the database encountered by the customer tenant. These records can be viewed for troubleshooting.

---

**Note:** Additionally, you can also create reports based on your system errors by selecting **System Error Log** as the **Object Type** to report on.

---

The Managing Users link allows you to create user records and manage their access, see [Security and Access Control](#) on page 375.

## Transfer Owners

Use the Transfer Owners feature to change relationships between one selected user and records of another type to a different user. Click the Transfer Owners link and select:

Warning: There is no easy way to roll back these changes, other than editing each record one-by-one. Do not use this feature for temporary reassessments.

- Relationship: Select one or more relationships between the User Object and other Objects (ownership of accounts, cases, etc.) that you'd like to change
- Current Owner: Select the user you'd like to reassign records from
- New Owner: Select the user to assign the records to

Rollbase will replace the current owner with the new owner for the selected relationships.

## Using Company-wide Settings

Use this feature if you want to manage a set of custom fields that represent system or customer-wide properties (rather than object-specific properties) that can be used in a number of areas throughout all your applications such as Templates and Formula fields and can be conveniently changed from one central location. Examples of such fields may be tax rate, your marketing slogan, etc.

If you use settings fields in an application, don't forget to include the **Settings** object in the application before publishing it.

To use Company-Wide settings, follow the steps below:

1. Use the Configure button (Setup > Administration Setup > Settings > Configure) to define these Fields.
2. Use the Edit button (Setup > Administration Setup > Settings > Edit) to set/modify values for these Fields.
3. In templates and formulas, use the Helper group of tokens and then select the Settings field you want to use.

**Tip:** Use the Formula editor below to define a value to be displayed for this Formula field. You can use all features of the JavaScript language to define your formula, with only a few exceptions. [Click here](#) for more information about formulas.

Template Helper	String Tokens	
Settings	Regular Price [value]	Server-Side API
<code>{ !#SETTINGS.regular_price#value}</code>		<code>regular_price</code>
<input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Save"/>		

## Calculating Sales Tax Example

Consider example: you want to calculate sales tax rate that is different in different states. To do so you can create company-wide settings:

Setting Name	Integration Code	Value
California Tax	ca_tax	7.25
Arizona Tax	az_tax	6.60

Than you can use formula:

```
switch ("{{!state#code}}") {  
    case "CA": return {{!#SETTINGS.ca_rate}};  
    case "AZ": return {{!#SETTINGS.az_rate}};  
    default: return 0;  
}
```

Advantage of using settings rather than substitute values into formula directly:

- You can change settings in a single convenient place with controlled access.
- You can publish your formula without referring to specific values.

## Using Scripts to Change Setting Fields

You can change Settings fields using server-side script the same way you would change data records. Use {{!#SETTINGS.id}} token for record's ID:

```
rbv_api.setFieldValue("$SETTINGS", {{!#SETTINGS.id}}, "ca_rate", 8.25);
```

## Account Settings

This link will take you to the Rollbase Support Portal, where you can change some settings for your Customer (Account) and customize some aspects of your Rollbase appearance (colors, sidebar components, etc). The following table summarizes the meaning of Fields that can be modified.

- **Company Name:** Name of your company, used for display purposes
- **Company Logo:** Image to be rendered in upper-left corner of Rollbase pages. If Company Logo is uploaded it will be used instead of Application logo.
- **Address info, Administration contact:** Can be used in your template and formula fields.
- **CSS Stylesheet:** Allows you to choose between standard Rollbase CSS and custom CSS which you may upload as Hosted File (see [Hosted Files](#) on page 267)
- **Page Components:** Allows you to customize which components appear in the sidebar and header on all Rollbase pages. For example, if you un-check the Create, box you will not see the Create section on the sidebar. You can also hide the Full Text Search box on the top of the page and the Support menu in the right corner of the page.
- **Date format:** Used as the default date when no user settings are available.
- **Time Zone:** Used as the default time zone when no user settings are available.
- **Email Settings:** These settings will be used as the "Reply To" address in cases when Rollbase sends an email that was not invoked by a specific user.
- **Default Email:** Footer Used as the email footer in cases when Rollbase sends an email that was not invoked by a specific user, or when a user does not have an email footer defined.
- **Email Encoding:** Used as the email encoding in cases when Rollbase sends an email that was not invoked by a specific user.
- **Base Language, Additional Languages:** see [Localization](#) on page 171.
- **Security Level:** see [Security and Access Control](#) on page 375.

- **Expiration Policy:** Number of days before user passwords expire (see [Security and Access Control](#) on page 375).
- **Detailed Log:** Check this box if you wish to have detailed system logs of users' activities and API calls.

Using Stylesheet and Page Component settings, you can customize the appearance of Rollbase Pages to a certain extent for all users in your organization.

## Administrative Settings

The following settings are available:

- **Multi-Currency Support:**

This Page allows you to manage currency names and codes. See [Multi-Currency Support](#) on page 214.

- **Email Server Settings:**

This Page is only available for Master Server administrators. This page assists in editing and testing Email Server settings (see [Administration](#) on page 465). Use this page to review and edit Email Server settings: host name, port, user name, password, default auto-reply address. To test your settings press "Test Settings" button to send sample email message to specified address. When message is sent and successfully received in mailbox, save settings into shared.properties configuration file. Your updated settings will take effect after server's restart.

Important: if you're using multi-server configuration, copy updated shared.properties configuration file to all servers.

- **Authentication:**

You can configure Rollbase to perform external call to verify user's credentials. For information about user authentication, see [Rollbase User Authentication](#) on page 386.

- **Whitelist of IP Addresses:**

As an additional security measure you can specify whitelist of IP addresses to be checked when user logs in. For information on security and access control, see [Security and Access Control](#) on page 375.

## Backup and Restore

As Rollbase Hosted cloud is a fully managed Software as a Service (SaaS) solution, Rollbase handles data backup and other disaster recovery solutions. However, you can protect yourself from inadvertent data losses, such as unintentionally deleted records, objects, or applications, by scheduling backups of application data.

Note that, backups can only be performed by users who have administrative access. Rollbase keeps up to seven copies of the most recent backup files. Only one backup file per tenant can be created per 24-hour period for performance reasons.

The process of creating a backup file runs asynchronously and uses a queue. You receive an email when the backup is completed and ready to be downloaded. A full backup is in a compressed (ZIP) file that contains:

- Relational data, all system tables with data as CSV files (one CSV per object definition).

- Binary data, all hosted files, file templates, and images.

Rollbase hosted cloud enforces a 1 GB limit for your backup data. Note that your backup file might not include the uploaded file templates, images, and hosted files if your storage exceeds the specified limit. For example, if you have 500 MB of relational data and 800 MB of hosted files, then Rollbase creates a backup file with only 500 MB of relational data. It does not back up the hosted files.

To back up more than 1 GB of data (relational data and hosted files), you can configure and manage data backup using your own Amazon S3 cloud storage account or you can contact Rollbase customer support to resize your 1 GB data backup limit.

Rollbase private cloud sets the backup data limit to 20 MB per customer tenant. As an administrator of the Rollbase master tenant, you can reset the 20 MB limit using the `StorageUsageLimitForBkp` property value in the `shared.properties` file. The `StorageUsageLimitForBkp` property value specifies the backup data limit for all the customer tenants. If you want to specify a different backup data limit for a particular customer tenant, you must **Edit** the customer record and update the **Max Backup File Storage (MB)** field with the different backup data limit. This backup data limit specified in the customer record overrides the backup limit set in the `shared.properties` file. For more information about `shared.properties` and working with customer records, see [shared.properties](#) on page 505 and [Working with Customer Records](#) on page 471.

To create and download a backup file containing all the record data in your account, go to **Administration Setup** and click **Backup**. You can either **Create a new system backup** file or **Download** an existing backup file. Click **Download** to download a copy of your full backup.

Additionally, you can also click **View Backup Log** to view all the data backup activities performed.

## Restore

If you are using Rollbase hosted cloud, you must contact Rollbase customer support to restore your data from a particular backup file.

If you are using Rollbase private cloud, Administrators of the master tenant must do the following to restore data from a backup file:

1. Acquire a backup file from your customer.
2. In the **Administration Setup > Backup** page, select **Upload System Backup** and upload your backup file. After the upload process is complete, your backup file will be listed in the **Backup** page.

---

**Note:** Alternatively, you can make a backup file available in the **Backup** page by manually copying the backup file in the storage location of your Rollbase instance. The storage location is specified in the **Backup** page. Typically,  
`<drive:>\Progress\Rollbase\Pas_Instance\rollbase\storage\<customer-ID>\backup`.

---

3. Click **Restore** to restore data from their backup file.

---

**Note:** The restore operation erases any changes made since the backup file was created.

---

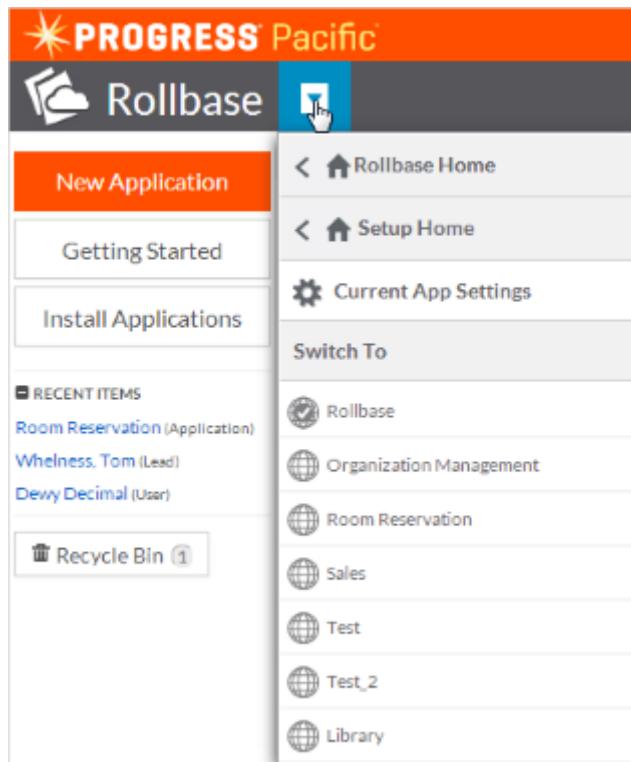
If you are an ISV and/or are using Rollbase private cloud and you want to use third-party cloud services for storage, see [Using a Third-Party Cloud Service for Storage](#) on page 551.

# Batch Jobs

Similar to Oracle Cron jobs, Rollbase batch jobs can be used to schedule periodic system maintenance, notification, and integration tasks. Batch jobs use the permissions set on each object for the **Query API**.

To create a new batch job:

1. Navigate to **Setup**. For example, From the applications menu, select **Setup Home**:



2. Under **Administrative Setup**, click **Batch Jobs**.

3. Click **New Batch Job**.

4. Select one of the following options:

- **System Backup:**

This job generates a system backup file. Optionally, it can upload the backup file to a remote FTP server.

- **Data Maintenance:**

This job runs specified the specified object script from the Rollbase client-side API on each record of the selected type.

- **Generate Report:**

This job runs the selected report and sends it as an email attachment to the specified recipient.

- **Upload Report:**

This job takes a snapshot of report data and uploads it via FTP to a remote server in CSV, XLS, or XML format.

- **Re-index Search Data:**

This job re-creates the index used for global text search.

- **Scheduled FTP Import:**

This job uploads spreadsheet file from a remote FTP server and starts the import process defined in the selected import map.

5. Enter the job-specific settings. Settings common to all job types include:

- **Job Name**

- **This batch job is deployed.** Uncheck to save the batch job settings but disable the batch job from running.

- **Start At**

- **Repeat Every** (the period of time between runs, with a one day minimum)

- **Notify Address**

6. Click **Save**.

## Managing Batch Jobs

Navigate to **Setup > Administrative Setup > Batch Jobs** to see the list of existing **Batch Jobs**. From the list of current **Batch Jobs** you can perform the following:

- Click **New Batch Job** to create a new job.
- Click **Queue** to see all scheduled jobs.
- Click **API Permissions** to view **Server API** role permissions. Batch jobs use the permissions set on each object for the **Query API**. To change **Server API** permissions, click **Edit Permissions**.
- Click **Jobs Log File** to view the job log.
- Click **Restart All** to immediately restart all jobs.
- Click a link in the **Action** column of the batch job row to edit, delete, run, or restart that particular batch job.
- Click the job name to see job properties.

Batch Jobs	New Batch Job	Queue	API Permissions	Jobs Log File	Restart All	
Action	Job Name	Job Type	Deployed	Will Run At	Last Updated	Last Updated By
Edit   Del   Run Now   Restart	System Backup	System Backup	<input checked="" type="checkbox"/>	04/26/2014 11:31 AM	04/25/2014 12:31 PM	Adam Ministrator
					Total	1

## Date Formats

Every user can choose a format for Date and Date/Time fields using the My Settings Page.

A date format can also be set:

- Per Customer (to be used as the default date format when no user selection was made or not available, such as when a system-generated email is sent)
- Per Portal (used in portal pages when displaying date and date/time values)

Note: In addition to Date Format, a user (customer or portal) can select a Time Zone. Rollbase re-calculates all date/time values to match the user's selected time zone before they are displayed.

The available date formats are listed in the table below:

Format	Date Field	Date/Time Field
mm/dd/yyyy (12 hrs)	05/22/2011	05/22/2011 03:49 PM
mm/dd/yyyy (24 hrs)	05/22/2011	05/22/2011 15:49
dd/mm/yyyy	22/05/2011	22/05/2011 15:49
dd.mm.yyyy	22.05.2011	22.05.2011 15:49
yyyy-mm-dd	2011-05-22	2011-05-22 15:49
dd-MM-yyyy	22-05-2011	22-05-2011 15:49
yyyy/MM/dd	2011/05/22	2011/05/22 15:49
yyyy-MMM-dd (uses short English name for Month)	2011-May-22	2011-May-22 15:49

## Currency Formats

Every Currency field and every Formula field that returns a value of type "Currency" allows selection of a currency format to use (this field-specific format is used for all users).

Note: If your Object supports the Multi-Currency attribute you can create a Base Currency Field that corresponds to each Currency Field to automatically calculate that field's value in the base foreign currency. You can also select the Base Currency format. See [Multi-Currency Support](#) on page 214.

The available Currency formats are listed in the table below:

Format	Output
\$###,###,###.##	\$123,000.50

Format	Output
£###,###,###.##	£123,000.50
€### ### ###.##	€ 123 000.50
### ### ###.##	123 000.50
### ### ###	(no decimals) 123 000
\$###,###,###.##	(no decimals) \$123,000
###.###.###,##	(comma for decimals) 123.000,50
¥###,###,###	¥123,000

## Billing and Support Settings

Starting Rollbase 3.1, the administrator of the Rollbase tenant can assign its users the following two roles:

- **Billing Administrator:** A user who can access and upgrade the Rollbase account type, and administer the invoice history of the Rollbase instance. This role can only be assigned to a Administrative user. By default, the administrator of the Rollbase tenant is the Billing administrator. For example, only the Billing Administrator can upgrade Rollbase subscription from Evaluation to Professional.

The **Buy Now** button at the top-center of a Rollbase instance of users with an Evaluation license and **Upgrade Subscription** button in the **Subscription Details** page is only available to the Billing Administrator. Additionally, for paid customers, the **Invoice History** page is only available to the Billing Administrator.

- **Support Contacts:** Users who are the designated support staff of the Rollbase instance. This option is only available for paid customers. The users assigned as Support contacts handle the product support related jobs that include:

- Accessing product knowledge base and product lifecycle related information
- Opening and managing support cases

For Basic and Developer license, you can only add one user as a support contact. For Professional license, you can add a maximum of five support contacts.

The administrator of the Rollbase tenant can access and assign the aforementioned roles from **Setup Home>Administration Setup>Billing and Support Settings**:

Billing And Support Settings

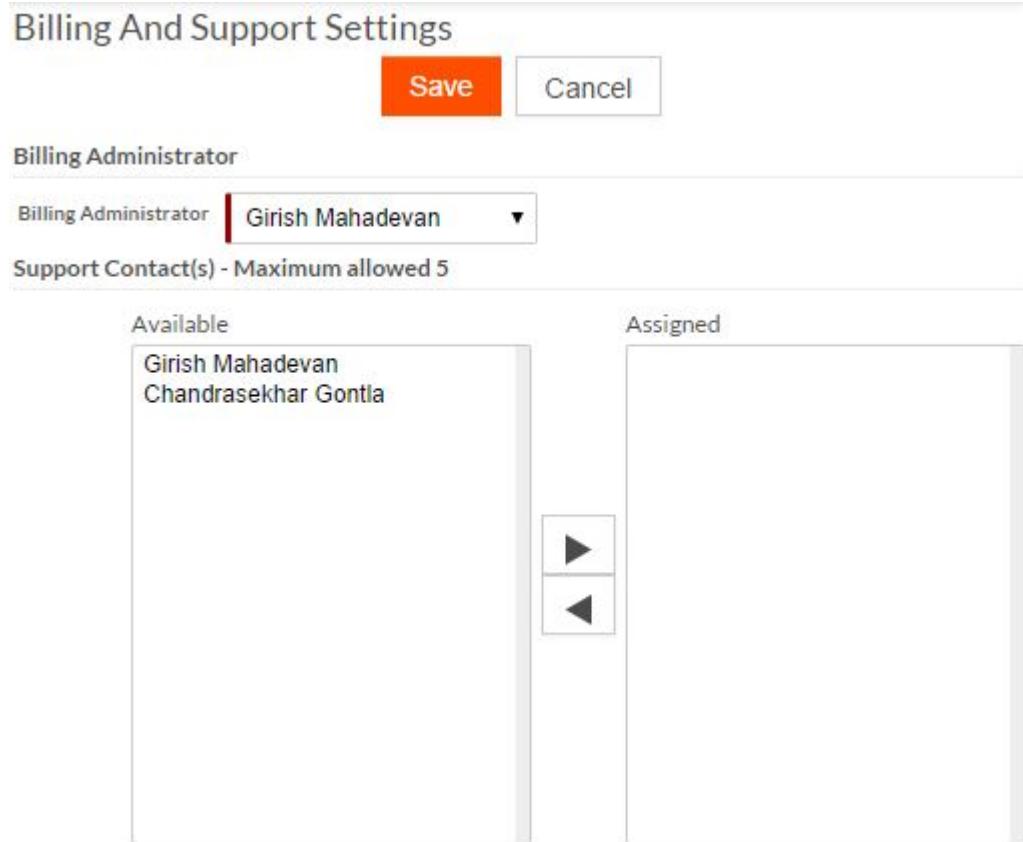
**Billing Administrator**

Billing Administrator Girish Mahadevan ▾

Support Contact(s) - Maximum allowed 5

Available	Assigned
Girish Mahadevan Chandrasekhar Gontia	

Save Cancel



## Language Support

Rollbase customer tenants have a base language and administrators can optionally select additional languages. If you publish a translated application, your translation will be preserved and installed in the destination Customer. Progress recommends choosing the base language carefully because it should not be changed. When multiple languages are enabled:

- Each user can select their preferred language on the **My Settings** page that affects the language used to render all text on each page. Users can select either the base language of the company or any of the additional languages supported by it. The base language is selected by default.

## User: Adam Ministrator

**Save**

Save &amp; New

Cancel

## Contact Information

First Name

Adam

Email Address

myemail@gmail.com

Middle Name

Phone

Last Name

Ministrator

Mobile Phone

Job Title

Rollbase Expert

Fax

## Login Name

Login Name

myprogressid

## User Preferences

Rows Per Page

Do Not Animate Collapse



## User Settings

Language

Portuguese

Date Format

07/15/2014 01:45 PM

Time Zone

(GMT-05:00) Eastern Standard Time (EDT) America/New\_York

Email Footer

Email Encoding

ISO-8859-1

- All application component names can be stored in the base language and additional languages. The first entry is always in the base language of the tenant regardless of the current user's language setting.

## Object Properties

Define a singular and plural name for this object definition. These names will be used throughout your account to refer to one or more records of this type.

Singular Name	Account	
	Compte	[French]
	Conta	[Portuguese] Example: Project
Plural Name	Accounts	
	Comptes	[French]
	Contas	[Portuguese] Example: Projects

- To translate values for picklists, radio buttons, and groups of check boxes, save the picklist first in the base language. Edit it and enter your translated values in the translation box below the base language box. Do not change or override base language values (which are included by default) in the top box. The translation boxes will not appear unless the field definition has been saved. In the bottom box, enter the base language value, a colon, and the additional language value. For example:

North: Nord

Enter the list of values for the picklist field below. Each value should appear on its own line.

Values
Fall Spring Summer Winter

Fall: outono  
Spring: primavera  
Summer: verão  
Winter: inverno

[Portuguese]

- Selected text fields can be translated to all these languages. To enable this feature check the "This field allows storing multiple values for supported languages" box in the field definition:

### Meeting: Edit Field

**Save** **Cancel**

#### Field Properties

Field properties are global settings that apply to this field wherever it is presented for input or display.

Field Type **Text (100)**

Field Label **Location**

[Portuguese]

[French]

You can specify text label to be used in the headers in Views and Reports. This label is optional. If not specified, Display Label will be used.

View Header

[Portuguese]

[French]

View Width

pixels or %

Default Size

Determines size of HTML input box

Length

**100**

Maximum number of text characters in input

This field allows storing multiple values for supported languages

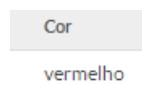
When users create records they will see boxes to enter values for each language:

Color

[Portuguese]

[French]

The value shown when viewing a record is selected based on the user's selected language preference (if multiple values are stored).



Rollbase supports the following languages:

- English
- German
- Spanish
- French
- Dutch
- Portuguese
- Chinese (simplified)
- Japanese
- Korean
- Norwegian

## String Tokens

Templates can include String Tokens which will be replaced by localized text during parsing. Selected String Tokens will be published and installed as part of a Rollbase application.

To select/create String Token use link in Token Helper component:



Next select existing String Token (it will be copied into "Select Merge Token" box) or create a new token and specify its text in available languages.

A screenshot of the 'Edit String Token' dialog box. The title bar says 'Edit String Token' with 'Save', 'Save & Close', and 'Cancel' buttons. The main area is titled 'Define String Token' and shows a 'Red = Required Information' note. It lists two entries: 'English' with 'Kyoto' and 'Japanese' with '京都'. Below this is a 'Token Name' field containing 'kyoto'. At the bottom are 'Save', 'Save & Close', and 'Cancel' buttons.

## Improving Translation Resources

Each language supported by Progress Rollbase is managed in a resource file for that specific language allowing easy improvement of translations throughout the platform. The default translation of each Rollbase string resource is machine-assisted and far from perfect. If you are proficient in a language supported by Rollbase you are invited to volunteer to improve any of our default translations. Upon request, we will email you the resource file in one of the supported languages listed above.

When working with Rollbase resource files, below are some important rules to follow:

- Do not try to deal with entire file at once - it is too big (about 3000 lines). Try to fix only strings you really care about first, or take a phased approach. Do not use automated translation tools.
- The file has the format token = value and each line contains exactly one token-value pair. This structure must be preserved:

`__Please_select__ = -- Please select -`

- Do not alter the token, as this will render the translation useless. Only translate the value portion as follows:

`__Please_select__ = - S'il vous plaît sélectionnez -`

- Many string values contain special placeholders such as {0}, {1} etc. These placeholders should not be altered, though they can be moved to a different location within the value. At runtime these placeholders will be replaced by actual pieces of data:

`_related_child_0 = (related child {0})  
_related_child_0 = (enfant lié {0})`

- Some string values contain HTML formatting tags such as <b> or <div>. These tags should not be altered; only the text inside should be changed:

`_0_equals_b_1_b_ = {0} égale <b> {1} </ b>`

## Adding Support for Other Languages in Private Cloud

In Rollbase Private Cloud, you can add support for new languages by translating the English language resource file `lang_en.properties` and saving the translation as a new `lang_NN.properties` file, where NN corresponds to the 2-letter ISO 639-1 language code. The language files are stored and accessed from the rollbase installation directory, `<ROLLBASE_HOME>\pas\rollbase\res`.

---

**Note:** You must restart your server for the new language to be made available in **Administration Setup>Account Settings>Language Settings**.

---

For instance, to add support for Vietnamese language:

1. Go to <ROLLBASE\_HOME>\pas\rollbase\res, create and save your translations in a lang\_vi.properties file.
2. Restart your server and open your Rollbase instance.
3. Select Vietnamese language from the language options available in the **Administration Setup>Account Settings>Language Settings** page.

## Translating Applications

Application component labels, such as those for objects, tabs, and menus, can be translated on their respective edit pages. You can translate an entire application by mapping the translated terms to base language terms in an XLS spreadsheet. Translation is only available if one or more foreign languages are assigned in the customer tenant **Account Settings**. And, translation is limited to languages available in those account settings. Use the **Translate** option from the **More Actions** menu in **Application Settings**.

The screenshot shows the 'Room Reservation: Translation' page. At the top, there is a navigation bar with 'Personal Setup', 'Applications Setup', and 'Administration Setup'. To the right, there is a link to 'return to Room Reservation'. Below the navigation, the page title is 'Room Reservation: Translation'. There is a tip box with an icon of a lightbulb. The tip text says: 'Select the language you want to manage translations for and then click the link to download the spreadsheet file for that language below. This file will contain a complete list of string resources used by your application in the selected language. Modify the last column of this spreadsheet to include your translations for the selected language and leave other columns unchanged. Finally, upload the translated spreadsheet here to update all translations at once.' A 'Learn more' link is also present. Below the tip box, there is a 'Select Language' dropdown set to 'Portuguese'. Below the dropdown is an 'Upload Translation' section with a 'Choose File' button and a message 'No file chosen'. Below that is a link 'Download current translation as XLS spreadsheet'. At the bottom of the page are 'Save' and 'Cancel' buttons.

Select one of the available languages and click the link to download the current translation as a single XLS file. That file has the following format:

Column	Content
A	Original ID of item
B	Component's name
C	Component's type
D	Field's name (inside component)
E	Text in base language
F	Text in foreign language

Do not modify content of first 5 columns (that may render spreadsheet useless) and only change translations in the last column using text in column E as a reference. The same rules as for Rollbase resources apply. The following screen shows an example translation spreadsheet with a base language of English and a translation in Portuguese:

	A	B	C	D	E	F
1	ID	Component	Type	Field	en	pt
2	7549	Room Reservation	Application	name	Room Reservation	Reserva de Quarto
3	207510	Device	DataObjectDef	singularName	Device	Dispositivo
4	207510	Device	DataObjectDef	pluralName	Devices	Dispositivos
5	823362	Rooms	Field	displayLabel	Rooms	Quartos
6	823371	Description	Field	displayLabel	Description	Descrição
7	850495	Manufacturer	Field	displayLabel	Manufacturer	Fabricante

When you finish the translation, save the XLS file on your local disk, navigate to the **Translation** dialog, verify that the correct language is selected and upload the XLS file. Uploaded resources will be stored in your application. If you later modify a translation, upload the new translation and re-publish the application or serialize it to XML.

## Global Text Search

This page displays information related to Global Text Search (see [Search](#) on page 43):

- Lists Objects and their "Text Index" attribute
- Lists Fields with "Text Index" attribute
- Allows start re-indexing for selected Object
- Allows start re-indexing for all Objects
- Allows browsing search-related log file

## Support

You can access Progress Rollbase support options from the **Forums** link of the drop-down menu near your profile in the upper right hand corner of the main Rollbase UI screen. The following resources are available from within the interface:

- Online Help

Access the online help from the **Help Me** link in the Rollbase footer. The online help contains the latest documentation and many useful features such as responsive design, the ability to translate pages using Google translate (click the globe icon on the toolbar and give it time to load), the ability to print pages, and search content.

- PDF

The Rollbase environment contains links to individual chapters of the "Rollbase in Action" book in PDF format. We recommend only using the PDF versions for printing, since the online help contains the latest information and better search capability.

- Forums

You are encouraged to participate in the Rollbase forum. We generally answer most of questions within 24 to 48 hours.

## **Forums**

This menu will take you to the Progress Community, where you can participate in forums submit a support request along with a bug report if applicable. You can also monitor previously posted requests and add comments to them for further followup by Progress staff. We generally respond to support tickets within 24 to 48 hours.

## **Subscription Details**

This menu will take you to the "About Your Account" page, where you can see important information about your tenant including:

- Type of service
- Security settings
- Presentation settings
- Resources available used and limitations on resources
- Current release information



---

# 12

## Installing and Administering Private Cloud

---

The topics in this section describe how to install, configure, and administer Rollbase Private Cloud. For details, see the following topics:

- [Introduction](#)
- [Installation](#)
- [Administration](#)
- [Multi-server Environments](#)
- [Configuration File Reference](#)
- [PAS Command Line Reference](#)

## Introduction

Progress® Rollbase® Private Cloud is a fully functional version of the Rollbase platform that you can download, install and host on your own servers or on another cloud platform. A Private Cloud system includes all of the design and runtime functionality of hosted Rollbase. Rollbase Private Cloud supports single server and multi-server deployments, as defined by your license, see <http://www.progress.com/products/rollbase/pricing/rollbase-private-cloud> for pricing details.

Private cloud end-users can be individuals or departments within your organization or others to whom you sell Rollbase applications and services. You can create separate tenants to keep sets of applications separate and secure for different groups of users. Each customer tenant will have its own login account.

## Evaluating Rollbase

You can evaluate Rollbase Private Cloud as long as you want at no cost. You simply register, download and install the software with no license. Without a license, Rollbase enforces the following limitations:

- Every page displays the notification, **Free Rollbase Evaluation**.
- A maximum of one database instance (MySQL, OpenEdge, DataDirect Cloud, MS SQL Server, or Oracle) and one Rollbase instance
- A maximum of two **Customers** tenants, with two **User** accounts and 5000 object records (total for all objects) in each tenant.
- Progress Technical Support only provides answers to installation-related questions. However, you can join Progress Community [Rollbase Technical Users](#) forums, which provide answers to a variety of questions.

## Runtime Architecture

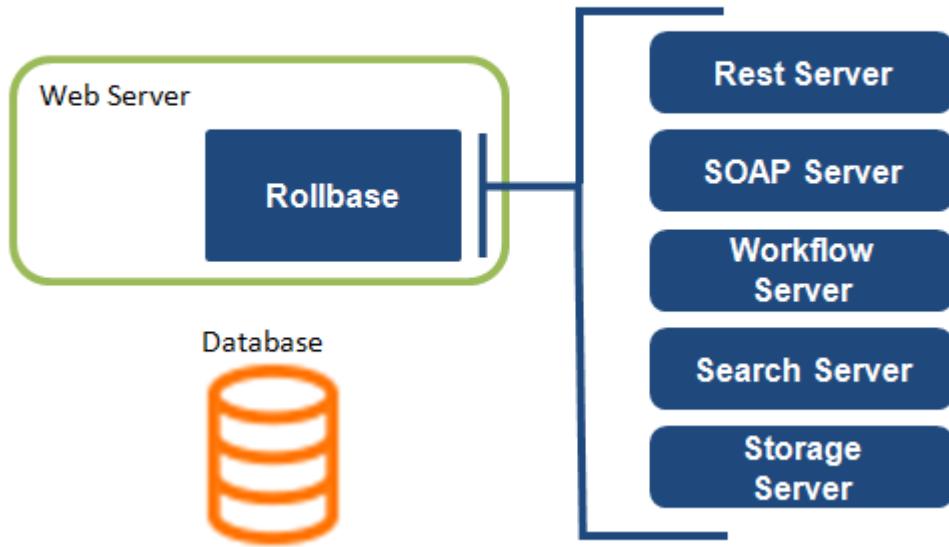
Rollbase Private Cloud requires the following runtime components:

- One or more Java Web application server instances to which Rollbase components are deployed (The installer includes the Pacific Application Server (PAS)). PAS is based on Apache Tomcat® but is tailored to be secure for use in production systems. PAS also simplifies the process required to create and run multiple Rollbase components on different hosts. Each PAS shares the executables and libraries of a common Tomcat server, but each instance is a separate process, running in a separate JVM, with its own configuration, such as for ports, security framework, and Web applications.
- One or more database instances. The database can be on its own host or be collocated with Rollbase components. The following image illustrates a single server Rollbase Private Cloud architecture with Rollbase auxiliary components identified in the call-out.

If you choose not to use PAS, you will need to install and configure your Web server to work optimally with Rollbase. Since Progress certifies use of Apache TomcatX with Rollbase, the documentation provides PAS and Tomcat-specific information. If you are using another Web server, refer to that documentation to find the equivalent functionality.

The multi-server architecture supports multiple instances of all components, which can be distributed for performance and scalability. See [Planning Your Multi-server Architecture](#) on page 481

## Rollbase Private Cloud Architecture



The first Rollbase instance you install is a Master Server, from which you can:

- Create and manage customers (tenants).
- Monitor system components.
- Setup ISV partners.
- Manage a shared applications directory and a support portal.
- Run applications for your own business, such as CRM, bug tracking, and customer support.

See [Administration](#) for more information.

Progress offers the following installation package options:

- The Rollbase Private Cloud installer for Microsoft® Windows and Linux operating systems. In addition to Rollbase, you can optionally install an OpenEdge database, and the Pacific App Server, a pre-configured instance of Tomcat. See [Using the Rollbase Installer](#) on page 450.
- Zipped packages allow advanced users to install Rollbase components manually. This method of installation works best for multi-server environments or for operating systems not supported by the Private Cloud Installer. See [Setting Up Rollbase Manually](#) on page 453.

## Supported Platforms

Progress Software certifies the versions of software listed below to work with the current version of Rollbase Private Cloud. Other versions might work with no problems, but are not certified.

### Operating Systems

Rollbase is a 100% Java application and as such is operating system-agnostic. However, the Rollbase Private Cloud installer only runs on the following operating systems:

- Microsoft® Windows® 7 (64-bit)
- Red Hat® Enterprise Linux® 6.1 (64-bit)
- SUSE® Linux 11 (64-bit)

## Java

Rollbase requires an Oracle Java 7 Runtime environment and will not run with older Java versions.

## Application Web Server

Rollbase requires a Java-based application server. The following are certified:

- Progress® Pacific Application Server 1.0.1 (based on Apache Tomcat®)
- Apache Tomcat® 7.0.42

## Databases

Rollbase Private Cloud requires a database and is certified with the following:

- Microsoft® SQL Server® 2008
- MySQL™ 5.6.x
- Oracle® 11gr2
- Progress® OpenEdge® 11.3.1 and 11.3.2 (11.3.1 is included in the Rollbase Private Cloud installer)

## Browsers

To use Rollbase, you need an up-to-date web browser with JavaScript and Cookies enabled. Progress recommends use of the following browsers:

- Internet Explorer® 9.0 +
- Firefox® 21 +
- Chrome™ 27.0 +
- Safari® 6.0 +

# Licensing

Under the terms of the Rollbase license you cannot reverse-engineer or modify the Rollbase software. You can, however, customize the appearance of your web pages by replacing standard Rollbase resource files (images, icons and CSS) with your own files.

Each Progress Rollbase Private Cloud license is associated with a particular domain name, such as [www.mycompany.com](http://www.mycompany.com). You must purchase a new license to use a different domain name. Each Private Cloud license has an expiration date, after which Rollbase will no longer run. The expiration date includes a window of time to renew the license.

You can delete the license file to switch to the free evaluation edition. However, if you have more than two tenants, you will get a license error because a free evaluation only allows two tenants.

See [Private Cloud pricing options](#) For enterprise and ISV pricing or to purchase, see the [contact information](#) on our website.

## Packaged OpenEdge License Restrictions

If you install the Progress OpenEdge database that is packaged with the installer, that instance of OpenEdge can only be used as the embedded database for the Rollbase application and the standard Rollbase tables. Specifically, the following restrictions apply:

- The database cannot be used for OpenEdge (PUB schema) ABL-accessible tables.
- You cannot add SQL tables to this database.

## Private Cloud Updates

Progress Rollbase periodically provides Private Cloud software updates for download. To check on updates, go to this page: [Rollbase Private Cloud Downloads Portal](#), and log in using your Private Cloud download account, which is separate from your Progress ID. Some updates require you to also run update scripts on your database(s).

If you customize the look and feel of application web pages, updates might replace your resource files with Rollbase resources. It is your responsibility to maintain your resource customizations.

## Included Rollbase Applications

The applications listed in the following table are installed in the Web Sever deployment directory. For example, for PAS, in the webapps folder. If you distribute applications, make sure to include the required applications called out in the table.

Application Name	Description	
Rollbase	The main Rollbase system application that defines the <code>USER</code> object to manage user accounts and will be installed by default for each customer tenant.	Must be installed in every customer tenant at creation time. See <a href="#">Managing Customers and Users</a> for more information
Organization Management	Defines <b>Location</b> , <b>Department</b> , <b>Function</b> and <b>Group</b> objects.	Recommended for installation in every tenant at creation time.
System Console	Used to monitor and manage all Rollbase components, such as the creation of a new <code>Customer</code> , and includes the <b>Application Directory</b> portal.	To be installed in the Master Server only.
ISV Partner	Gives your ISV partners limited access to your Master Sever.	To be installed in the Master Server only.
Support Center	Includes a Support Portal to facilitate submission and processing of support requests from your users.	To be installed in the Master Server only.

Application Name	Description	
Approvals	Defines a default approval process (can be sequential or parallel).	Recommended for installation in every Tenant.
CRM	Simple CRM application that can be further customized and used as a starting point.	Can be installed as desired from your <b>Application Directory</b> portal.

## Third Party Software You Can Install

The software described in this section is used by Rollbase Public Cloud. You must download and install it separately, and purchase any required licenses.

### PDF Converter

Rollbase uses the PD4ML PDF Converter to convert HTML documents into PDF. PDF rendering functionality on all levels is disabled unless you download and install a PDF Converter. PD4ML Pro supports use of multi-byte Asian characters.

To install after purchase for use with Rollbase Private Cloud, follow these steps:

1. Download a distribution of PDF4ML or PDF4ML Pro for Java from: <http://pd4ml.com/index.htm>.
2. Copy `pd4ml.jar` and `ss_css2.jar` into the Web server `lib` directory. For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\common\lib`.
3. Restart PAS or Tomcat.

If you wish to use multi-byte Asian characters in generated PDF documents, perform these steps:

1. Make sure you've purchased the PD4ML Pro edition.
2. Locate the `shared.properties` file located in the `config` folder of your Rollbase installation. For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\rollbase\config`.
3. Open `shared.properties` with a text editor and edit the `FontDirectory` entry so that it points to the system fonts directory on your server. For example:

```
FontDirectory=c:\\Windows\\Fonts
```

4. Save and close `shared.properties`.
5. Using a command prompt, run the following command (substituting your font directory): :

```
java -jar pd4ml.jar -configure.fonts c:\\Windows\\Fonts
```

This will create a `pd4ml.properties` configuration file.

6. Restart PAS or Tomcat.

## JExcel API

Progress Rollbase uses the JExcel API to read, write, and manipulate Excel 2003 spreadsheets in XLS format. Progress Rollbase only supports JExcel API 2.6.9 (Java 1.6) and earlier.

If you want to use XLS format documents on your Private Cloud instance, do the following:

1. Download a supported version of JExcel API from <http://sourceforge.net/projects/jexcelapi/files/jexcelapi/>.
2. Copy `jxl.jar` into the Web server `lib` directory.

For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\common\lib`.

3. Restart PAS or Tomcat.

## Aspose.Words for Java

Rollbase uses Aspose.Words for Java to process Word-based document templates. Starting with Progress Rollbase 3.1, Rollbase supports Aspose.Words for Java 14.8.0.

To use Aspose.Words for Java with Rollbase Private Cloud, do the following:

1. Download and purchase Aspose.Words for Java 14.8.0 from: <http://www.aspose.com/community/files/72/java-components/aspose.words-for-java/default.aspx>.
2. Optionally, if you have a previous version of Aspose.Words for Java, you must remove the related `.jar` and `.lic` files from your Rollbase installation.
3. Copy `aspose-words-14.8.0-jdk16.jar` into the Web server `lib` directory.

For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\common\lib`.

4. Copy the `Aspose.Words.lic` license file into the `config` directory of your Rollbase installation.

For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\rollbase\config`.

5. Restart PAS or Tomcat.

## Aspose.Pdf for Java

Rollbase uses Aspose.Pdf for Java to process PDF writable forms and to extract plain text from PDF documents for search indexing. Starting with Progress Rollbase 3.1, Rollbase supports Aspose.Pdf for Java 9.3.1.

To use Aspose.PDF for Java with Rollbase Private Cloud, do the following:

1. Download and purchase the Aspose.Pdf for Java 9.3.1 from: <http://www.aspose.com/community/files/72/java-components/aspose.pdf-for-java/default.aspx>.
2. Optionally, if you have a previous version of Aspose.Pdf for Java, you must remove the related `.jar` and `.lic` files from the Rollbase directory.
3. Copy `aspose-pdf-9.3.1-jdk16.jar` into the Web server `lib` directory.

For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\common\lib`.

4. Copy the `Aspose.Pdf.lic` license file into the `config` directory of your Rollbase installation.  
For example, if you used the installer, installed PAS, and accepted the default folder, this location would be `Progress\Rollbase\Pas_Instance\rollbase\config`.
5. Restart PAS or Tomcat.

## StelsMDB Access JDBC Driver

Rollbase uses the StelsMDB Access JDBC Driver to convert Access databases into Rollbase applications. Conversion of Access databases into Rollbase Private Cloud applications is disabled unless you purchase download and install StelsMDB. To use the Access JDBC Driver with Rollbase Private Cloud, follow these steps:

1. Download and purchase the version 2.5 software from:  
[http://www.csv-jdbc.com/stels\\_mdb\\_jdbc.htm](http://www.csv-jdbc.com/stels_mdb_jdbc.htm).
2. Copy the `mdbdriver.jar` and `commons-lang.jar` files into the `TOMCAT_HOME/lib` directory.
3. Restart Tomcat.

## FusionCharts

Rollbase uses FusionCharts for visual representation of its application data in the form of charts. Starting Progress Rollbase 3.1, Rollbase only supports FusionCharts 3.4.0 and above. Install FusionCharts using the following steps:

1. Download the FusionCharts software from <http://www.fusioncharts.com>.
2. Extract and open the software folder in your computer directory.
3. Open the `js` folder and copy the following files:
  - `fusioncharts.js`
  - `fusioncharts.charts.js`
  - `fusioncharts.widgets.js`
  - `fusioncharts.gantt.js`
4. Paste the copied files in the `js` folder of your `master` and `prod` server instances.  
For example, if you installed PAS and accepted the default folder location during installation, this location would be `Progress\Rollbase\Pas_Instance\webapps\master\js\` and `Progress\Rollbase\Pas_Instance\webapps\prod1\js\`.
5. Restart your Web server.

## FusionWidgets

Rollbase uses FusionWidgets to render HTML-based gauge components. This software can be downloaded and purchased from <http://www.fusioncharts.com>. Follow these steps to use FusionWidgets with Rollbase Private Cloud:

1. Download the FusionCharts software from <http://www.fusioncharts.com>.
2. Extract and open the software folder in your computer directory.
3. Open the `js` folder and copy the following files:

- fusioncharts.js
- fusioncharts.charts.js
- fusioncharts.widgets.js
- fusioncharts.gantt.js

4. Paste the copied files in the `js` folder of your `master` and `prod` server instances.

For example, if you installed PAS and accepted the default folder location during installation, this location would be `Progress\Rollbase\Pas_Instance\webapps\master\js\` and `Progress\Rollbase\Pas_Instance\webapps\prod1\js\`.

5. Restart your Web server.

## Installation

Where and how you install Rollbase depends on the architecture you have selected, whether you are installing [Single Server Edition or Multi-server Edition](#), and which database(s) you plan to use. There are two ways to set up Rollbase:

- Use the Rollbase Private Cloud installer (for Windows or Linux operating systems)

The installer gives you the option to install a Progress OpenEdge Database and PAS, a pre-configured version of Tomcat. If you don't use these, you must install a Web server and a database separately and configure them to work with Rollbase. The OpenEdge and PAS instances installed by the Rollbase installer are not set up as Windows services. To configure the installed PAS instance as a windows service or Linux daemon, see [Installing and running an instance as a Windows service](#) on page 485 or [Installing and running an instance as a UNIX daemon](#)

- Install Rollbase manually by extracting zip files

The zip files provide flexibility for multi-server configurations and on operating systems on which the installer will not run. Manual installation with the zip files requires you to configure a web server and database separately and edit Rollbase configuration files.

The first Rollbase instance you install and configure becomes the Master Server. To use multiple Rollbase instances or auxiliary components, see [Multi-server Edition](#). In a multi-server configuration, you will log in to the Master Server for administering your Private Cloud deployment.

---

**Note:** Default installations of third party software often are configured to use minimum security. Please review vendor websites and documentation for recommended security configuration, such as changing default administrator passwords.

---

## Pre-Requisites

Before you can download and install Rollbase Private Cloud:

- You will need a [Rollbase Private Cloud download account](#).

- All hosts on which you plan to install Rollbase must have Java installed. The Private Cloud installer will install a JRE for you. If you are doing a manual install, download and install the latest version of the Oracle Java 7 Runtime Environment from [www.oracle.com](http://www.oracle.com).
- If you are going to use Rollbase with an existing Web server, make sure that it is stopped and not restarted until you finish the entire installation process.  
If you've installed Tomcat as a Windows service, run `tomcat<version>.exe` to stop the service (or navigate to the **Services** panel to stop it manually).
- You need a database and a compatible JDBC driver. The Rollbase Private Cloud installer allows you to install Progress OpenEdge, which includes a JDBC driver. To use a different database, that database must be configured with an account that Rollbase can access. The account should have all permissions. [Configuring a Supported Database](#) on page 455 describes the scripts Rollbase provides to create the necessary tables.
- On Windows, you need to run all installation and startup scripts as an administrator. On Linux, you need to run all installation and startup scripts as root.

The following topics describe the detailed installation procedures:

- [Using the Rollbase Private Cloud Installer](#)
- [Installing Rollbase Manually from Zip Files](#)

## Using the Rollbase Installer

If you will use Rollbase with an existing Web server, stop it before installing Rollbase.

---

**Note:** If you are upgrading from a previous release, please read [Upgrading to Version 3.0.x](#) before running the installer.

---

To use the Rollbase Private Cloud Installer, follow these steps:

1. From the [Rollbase Private Cloud Downloads Portal](#), download the zip file for your operating system:

- For Windows,  
`PROGRESS_ROLLBASE_FULL_INSTALLER_<version_number>_WIN_64.zip`.
- For Linux,  
`PROGRESS_ROLLBASE_FULL_INSTALLER_<version_number>_LINUX_64.zip`.

2. Unzip the downloaded file.

The resulting files include an installer executable and a zip file for the OpenEdge database.

3. Run the installer and click **Next** to start the installation process.
4. Accept the Progress Rollbase License agreement and click **Next**.
5. Optionally, specify the location of your license file.

Without a license, Rollbase will run in evaluation mode. You can add a license at any time.

6. Click **Next**.
7. Specify a destination directory for the installation and a working directory for the OpenEdge database. If you are using a different database, the working directory is not important.

**Note:** Do not use paths that contain spaces, such as `Program Files`.

8. Click **Next**.

9. Choose the database type you want to use for Rollbase.
  - **Progress Database** - Allows you to use an existing Progress OpenEdge Database or have the installer install one.
  - **Other Database** - Select this to use one of the supported databases, you will need to configure it yourself, as described in [Configuring a Supported Database](#) on page 455.
10. Click **Next**.
11. For **Server Details**, enter the host name and port number for Rollbase. If you have a license, the host name should match that in your license. If you elected to use an OpenEdge database, for **Database Details**, enter details for an existing OpenEdge database or a new one. If you want the installer to create the database, check the box.
12. Click **Next**.
13. Choose **Install New Pacific Instance** to install the Pacific Application Server or select the other option to use an existing Tomcat installation. If the latter, make sure the Tomcat instance is a supported version and is not running.
14. Click **Next**.
15. If you chose to install PAS, specify port numbers. You only need to change the port numbers if they conflict with ports already in use on your network.
16. Specify **Mail Server Details**. Rollbase will use this mail server to send emails. It can be an organizational mail server or email service such as Google (`smtp.gmail.com`, port 465) or Yahoo (`smtp.mail.yahoo.com`, port 465). Values must be entered but can be changed later in the `shared.properties` configuration file or in administrative setup. The [shared.properties](#) on page 505 file will contain the values you enter here.
  - **Host Name** - Host name of the desired email server.
  - **Port Number** - Desired email server port number.
17. Specify the **Administrator Details** email. This email address will identify the first administrative user on the system and must be a valid email address. This would typically be your e-mail address unless someone else will be administering Rollbase.
18. Specify **Email Account** information. Rollbase will send system emails from this account. Values must be entered but can be changed later in the `shared.properties` configuration file. For example, you might want to set up an account named something like `noreply@domain.com`. The [shared.properties](#) file will contain the values you enter here.
  - **User Name** - Email server user name used to send system messages.
  - **Password** - Email account password.
19. Click **Next**.
20. Review the **Pre-Installation Summary**. If acceptable, click **Next**.

The installation process starts.
21. Click **Done** when complete.
22. If you did not choose to install PAS, verify that your Tomcat installation is configured as described in [Using Your Own Instance of Tomcat](#) on page 452, including:
  - Specify as much memory as possible for initial and Max memory pools
  - Disable system persistence
  - Ensure proper UTF-8 support

- Specify <session-timeout> node value
23. If you are using a new Tomcat instance that you installed yourself, set `JRE_HOME` as described in [Set Environment Variables](#). Note that the installer will have already set `ROLLBASE_HOME`.

After installing Rollbase private cloud, proceed to [Post-Requisites](#) on page 452.

## Post-Requisites

After installing Rollbase Private Cloud:

- If you are using a database other than OpenEdge, you need to run the scripts for that database and enter information about it in the `databases.xml` file. Then, you can start the runtime components as described in [Starting Components and Logging In](#) on page 460.
- The default Web server configuration may not completely secure your Web server. For example, depending on your environment, unauthorized users could access your data on the servers. Progress recommends that you optimally secure your Web servers by conforming to the benchmarks given by Center for Internet Security (CIS). For information on CIS security benchmarks, see <http://benchmarks.cisecurity.org/downloads/browse/?category=benchmarks.servers.web>.

## Using Your Own Instance of Tomcat

If you are using your own Tomcat instance instead of PAS, Progress recommends the following:

- Use an instance of Tomcat dedicated to Rollbase. Download a version of Tomcat that Progress certifies.
- Start by using the default port 8080 for Tomcat. Later you can add Apache as a gateway to your Tomcat instance (Apache typically runs on port 80).
- Follow Apache's installation instructions. Once installed, start Tomcat and point your browser to <http://localhost:8080> to make sure that you see the Tomcat Welcome page. Stop Tomcat once confirmed.
- Run Tomcat from the command line during installation. To do this open a command prompt window and go to the bin directory within your Tomcat folder and run `tomcatX.exe` (where X is the main version number of your Tomcat installation).
- The default Tomcat installation is optimal for development purposes. For deployment, you will likely want to increase security. See the Tomcat documentation for details.
- Set Tomcat up to use as much memory as you can spare for its initial and maximum memory pools: for example, 1500MB in production on a 32-bit machine (you can more than double this for a 64-bit OS). However, if you set memory requirements too high Tomcat will fail to start.
- Disable session persistence: un-comment the section of `conf/context.xml` related to session persistence.
- For proper UTF-8 support, add `server.xml` include a `URIEncoding` attribute with value of `UTF-8` to all Connector nodes:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
URIEncoding="UTF-8"/>
```

- To ensure that Portal User log in sessions last long enough, update the `<session-timeout>` node value in the `web.xml` configuration file in the `conf` directory. The default timeout is 30 minutes. The appropriate value depends on your application and end-users usage patterns.
- Periodically delete files from Tomcat's log directory.

## Setting Up Rollbase Manually

The topics in this section explain how to manually install and configure Rollbase Private Cloud on a single server, such as: a dedicated in-house server, a third party server, a cloud infrastructure such as Amazon or Rackspace, or a laptop for testing. To use multiple Rollbase instances or auxilliary components, see [Multi-server Edition](#).

If the host on which you want to install Rollbase Private Cloud has a Windows or Linux operating system, using the [installer](#) instead of installing manually allows you to avoid a number of manual configuration tasks.

The high-level steps required to install manually include the following:

1. Verify that Tomcat and the database you plan to use with Rollbase Private cloud are installed and configured as described in [Using Your Own Instance of Tomcat](#) on page 452.
2. Verify that Tomcat is stopped and not restarted until you finish all necessary steps for installing and configuring Rollbase. If you've installed Tomcat as a Windows service, run `tomcat6w.exe` to stop the service (or navigate to Services to stop it manually). For Ubuntu Linux platform run the `shutdown.sh` script.
3. [Download and unzip Rollbase components](#).
4. [Set environment variables](#).
5. [Edit the `shared.properties` file](#)
6. [Run the Rollbase script for your database type](#) and [edit `databases.xml`](#).
7. [Start the runtime components and log into Rollbase](#).

## Download and Unzip Rollbase Components

The following steps assume that you have a Rollbase Private Cloud Download account and that you have a Web Server installed, such as Apache Tomcat. Make sure that the Web Server is stopped and not restarted until you finish all necessary steps below.

Follow these steps to download and unzip Rollbase:

1. From the [Rollbase Private Cloud Downloads Portal](#), download the following files to your server:
  - `rollbase.zip`: Configuration and resource files
  - `lib.zip`: Shared libraries
  - `webapps.zip`: Web archives for Rollbase and auxillary components
2. Create a directory to hold the Rollbase files, such as `\Progress`.  
The person who will start Rollbase must have permissions to write to this directory.
3. Preserving the folder structure, unzip `rollbase.zip` into the directory you created.

You should see a `rollbase` directory with the following sub-directories:

- `apps`: contains XML files for applications to be installed.
  - `config`: contains configuration files.
  - `res`: contains localized resource strings.
  - `docs`: contains documentation.
  - `sql`: contains SQL script needed to create Rollbase database.
4. Unzip `webapps.zip` into your web server deployment folder. For Tomcat, the `webapps` folder in its installation directory.
  5. Unzip `lib.zip` in the web server library directory. For Tomcat, the `lib` folder in the Tomcat installation directory.

Next, you will need to [set environment variables](#).

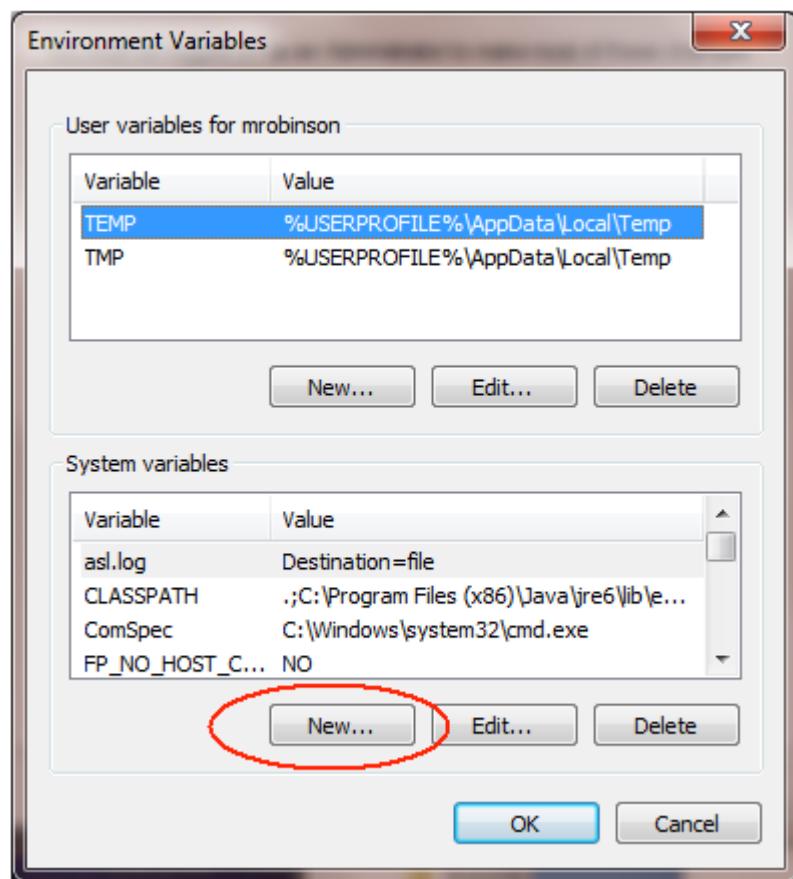
## Set Environment Variables

The `ROLLBASE_HOME` environment variable should point at your Rollbase installation, the directory containing Rollbase `config` and `res` folders. The `JRE_HOME` variable should point at the `jre` folder of your Java installation. If you used the Rollbase installer, it creates and sets these variables for you. This topic describes how to set these variables on hosts with Windows and Linux operating systems.

### On Windows Machines

To set the `ROLLBASE_HOME` environment variable on a host with a Windows OS:

1. In **Windows Explorer**, right-click **Computer** or **My Computer** and select **Properties**.
2. Click **Advanced System Settings**.
3. Click **Environment Variables**.
4. Under the list of **System variables**, click **New**
5. Enter `ROLLBASE_HOME` for the **Variable name** and the full path to the root directory of your Rollbase installation for the **Variable value**. For example, if you unzipped `rollbase.zip` in a `C:\Progress` directory, the path would be `C:\Progress\rollbase`.
6. Click **OK**.
7. Under the list of **System variables**, click **New**
8. Enter `JRE_HOME` for the **Variable name** and the path to the `jre` directory of your Java installation for **Variable value**. For example, if the Java installation is in `C:\Java`, the path would be `C:\Java\jre<version>`.



## On Linux Machines

On Linux or UNIX based machines use the commands `ROLLBASE_HOME=` and `JRE_HOME=`. For example, using the bash shell and assuming a location of `usr/share/rollbase`, the command would be as follows:

```
export ROLLBASE_HOME=/usr/share/rollbase
```

## Configuring a Supported Database

Rollbase certifies use of the following databases (see [Supported Platforms](#) on page 443):

- [MySQL](#)
- [OpenEdge Database](#)
- [Oracle Database](#)
- [SQL Server Database](#)

You should create a separate database user account for Rollbase. To improve database security Progress recommends that you only give the following access privileges to this account: `SELECT`, `INSERT`, `UPDATE` and `DELETE`.

The following topics describe how to configure the supported databases:

- [MySQL](#) on page 456

- [OpenEdge](#) on page 457
- [Oracle](#) on page 458
- [SQL Server](#) on page 458
- [Edit databases.xml](#) on page 458

## MySQL

If MySQL is on the same host as Rollbase, or is on another server reachable over the local network, run the `create_mysql.sql` script from the `sql` folder in the `Rollbase` directory of your installation as described in [Create Tables](#) on page 456.

If you do not have MySQL installed, download and install it, using the following options:

- Install for a developer machine
- Choose a transaction database (Rollbase does not use OLAP)
- Choose online transaction processing
- Choose `utf8` as the default character set

Verify that either the MySQL JDBC driver `mysql-connector.jar`, or the Progress DataDirect JDBC driver file, `mysql.jar`, is in the `lib` directory of your Web server.

### Create Tables

In the `ProgressRollbase\sql` folder of your Rollbase installation, locate the `create_mysql.sql` file and comment out the sections that do not apply to MySQL. If you installed PAS, this folder will be `Pas_Instance\Rollbase\sql`.

You can run the script as follows:

- On Linux or Unix machines use a Terminal service to run the `mysql` command, then use the `source` command to run the `create_mysql.sql` script.
- On Windows machines, use the `mysql` command line tool or install MySQL Workbench. to use the Workbench:
  1. Start from the **Workbench Central** screen and click **New Connection**.
  2. In a dialog enter a name for the new connection and specify a password for the root user.
  3. Open the newly created connection by double-clicking it.
  4. From the menu select **File > Open SQL Script**. Locate and open the `create_mysql.sql` file in the location where you unpacked `rollbase.zip`.
  5. Use the menu **Query > Execute (All or Selection)** to run the script from the opened SQL file and create the Rollbase database `rb_dbo` and all tables.
  6. Click **Refresh** in the **SCHEMAS** sidebar area. You should see an `rb_dbo` database and a list of tables inside, such as `rb_act_trail`, etc.

Next, modify the Rollbase configuration file as described in [Edit databases.xml](#).

## OpenEdge

You can install a dedicated OpenEdge database instance [Using the Rollbase Installer](#). The Rollbase installer sets up and configures the dedicated instance for you. To use an existing OpenEdge database, it must be the OpenEdge Enterprise RDBMS package.

When using OpenEdge database with Rollbase, Progress recommends that you become familiar with the physical storage structures of your Rollbase OpenEdge RDBMS instance and the many configuration parameters for the database. This helps you to ensure an optimal performance of your Rollbase applications and reliability of its data. To learn about OpenEdge database, see the *OpenEdge Getting Started: Database Essentials* and *OpenEdge Data Management: Database Administration* manuals in your OpenEdge documentation set. Additionally, if you need assistance with OpenEdge database management or administration tasks, you can contact Progress professional services or one of the third-party Progress consulting partners.

Use the OpenEdge `proenv` utility to execute a script provided by Rollbase. This creates the following:

- A database and the following related artifacts in a specified folder inside your OpenEdge Work directory
  - A structure definition file
  - A configuration parameter file
  - A database with a transaction log and data extents for four areas
- A `startdb.bat` or `startdb.sh` script to start the database
- A `stopdb.bat` or `stopdb.sh` script to stop the database
- An account for the specified user with the specified password

---

**Note:** OpenEdge has a variable length `VARCHAR` data type but SQL does not. Therefore, the OpenEdge Data Dictionary definition of a `VARCHAR(10)` field can hold 1, 5, 10, 100, 1000, etc. characters. OpenEdge's ABL can read or write any sized string, but an attempt to access this with SQL produces an error if the data is greater than 10 characters. Therefore, you should set the character value of `(N)` large enough to prevent such errors.

---

## Preparing an Existing OpenEdge Instance to Work with Rollbase

To configure the OpenEdge database for Rollbase:

1. Start up the Progress environment using the `proenv` utility, which is available from the OpenEdge `bin` directory or on Windows systems, from the Start Menu under **All Programs > Progress > OpenEdge <version>**.
2. Navigate to the Rollbase `sql` directory that contains the `create_oedb` script. This directory also contains the `create_oe.sql` script.
3. Using the `proenv` utility, execute `create_oedb.bat` on Windows systems or `create_oedb.sh` on Linux systems. This will create the Rollbase schema and database. The `create_oedb` script takes the following arguments:
  - `-dbname dbname`: The database name, which defaults to `rbdb`.
  - `-port portnumber`: The database port number, which defaults to `8911`.
  - `-user username` -- The username, which defaults to `dbadmin`.

- `-pwd password`: The password, which defaults to dbadmin.
- `-script filepath`: Full path to the SQL script file that creates Rollbase schema for OpenEdge. The default value is `create_oe.sql`, which is found in the same folder as this script.
- `-dbhome homefolder`: The folder inside the `WRK` directory where the Rollbase database is created. The default value is `oe_rollbase_db`. For example:

```
proenv > create_oedb.bat -dbname oe_rbdb -port 9911  
-user rbadmin -pwd mypwd
```

Upon successful completion, you should see output similar to the following:

```
Database home is "C:\\\\OpenEdge\\\\WRK\\\\oe_rollbase_db"  
Setup for database oe_rbdb COMPLETED OK  
11-12-2013 11:34:51.38
```

You can use the command `proenv> create_oedb.bat -h` to display the usage information for the script's arguments.

4. Modify the Rollbase configuration file as described in "[Edit databases.xml](#)".

## Oracle

To use Oracle, download and install the latest version from [www.oracle.com](http://www.oracle.com) if you do not already have an Oracle instance available. The Rollbase Private Cloud download includes the Progress DataDirect JDBC driver for Oracle, `RBoracle.jar`.

Use an existing Oracle user (schema) or create a new Oracle user. Then run the `create_ora.sql` script to create all of the required Rollbase tables:

```
sqlplus> @create_ora.sql
```

Modify the Rollbase configuration file as described in "[Edit databases.xml](#)".

## SQL Server

To use Microsoft SQL Server, download and install the latest version from [www.microsoft.com](http://www.microsoft.com) if you do not already have a Microsoft instance available. The Rollbase Private Cloud download includes the Progress DataDirect JDBC driver for Microsoft, `RBsqlserver.jar`.

Use an existing SQL Server database or create a new database. Then run the `create_ms.sql` script to create all of the required Rollbase tables.

Modify the Rollbase configuration file as described in "[Edit databases.xml](#)".

## Edit databases.xml

The `database.xml` file in specifies the URL, username, password, and other properties that Rollbase will use to access the database. You need to edit this file manually in the following circumstances:

- If you did not use the Rollbase installer.
- If you used the Rollbase installer, but did not choose to use the OpenEdge database.

The `database.xml` will be in one of the following locations:

- If you used the installer and had it install PAS:

```
>InstallationDir<\Rollbase\PAS_Instance\rollbase\config
```

- If you used the installer and did not have it install PAS:
- If you are installing manually from zip files, the config folder in the location where you extracted rollbase.zip.

Edit the file as follows:

1. Open database.xml in a text editor.
2. At a minimum, enter values for the following properties. For a full list of properties, see the reference topic for [databases.xml](#)
  - Database name
  - Driver
  - URL (Make sure that no white space exists after the URL and before the </Url> tag.)
  - DbUser (Enter credentials for the account used to execute the Rollbase create\_oedb script.)
  - Password

The following examples are in alphabetic order, by database type.

This example shows how to use a DataDirect Cloud to connect to a cloud data store with the Data Source of RB\_DBO. The Data Source must have been defined and tested using the DataDirect Cloud dashboard:

```
<Database name="RB" isDefault="yes" isExternal="no"
  MinConnections="3"
  MaxConnections="10" MaxInUseConnTimeMins="30"
  MaxNotUsedConnTimeMins="1" MaxConnLifetimeMins="60"
  TxIsolation="2" useTxRecovery="yes">
  <Driver>com.ddtek.jdbc.ddcloud.DDCloudDriver</Driver>
  <Url>jdbc:datadirect:ddcloud://service.datadirectcloud.com:443;
    databaseName=RB_DBO</Url>
  <DbUser>root</DbUser>
  <Password>my_password </Password>
</Database>
```

This example shows how to specify a JDBC driver and URL to connect to a MySQL database named RB\_DBO:

```
<Database name="RB" isDefault="yes" isExternal="no"
  MinConnections="3"
  MaxConnections="10" MaxInUseConnTimeMins="30"
  MaxNotUsedConnTimeMins="1" MaxConnLifetimeMins="60"
  TxIsolation="2" useTxRecovery="yes">
  <Driver>com.mysql.jdbc.Driver</Driver>
  <Url>jdbc:mysql://localhost:3306/RB_DBO</Url>
  <DbUser>root</DbUser>
  <Password>my_password</Password>
</Database>
```

This example shows how to specify a JDBC driver and URL to connect to an OpenEdge database. Rollbase includes the Progress DataDirect JDBC driver file, opendedge.jar. The example uses variables that are described below:

```
<Database name="RB" isDefault="yes" isExternal="no"
  MinConnections="3" MaxConnections="10"
```

```
MaxInUseConnTimeMins="30" MaxNotUsedConnTimeMins="1"
MaxConnLifetimeMins="60" TxIsolation="2" useTxRecovery="yes">
<Driver>com.ddtek.jdbc.openedge.OpenEdgeDriver</Driver>
<Url>jdbc:datadirect:openedge://localhost:8911;databaseName=rbdb</Url>
<!-- -->
<DbUser>dbadmin</DbUser>
<Password>dbadmin</Password>
</Database>
```

In the JDBC URL, the example uses `localhost:8911;databaseName=rbdb` for the hostname, port number, and database name. Note the following:

- **hostname** - Use `localhost` if the database resides on the same machine as Rollbase, otherwise use the actual host name.
- **port** - Use the port number on which the Rollbase database was started before executing the `create_oedb` script.
- **databaseName** - The name of the Rollbase database created when executing the `create_oedb` script.

This example shows how to specify a JDBC driver and URL to connect to an Oracle database:

```
<Database name="RB" isDefault="yes" isExternal="no"
          MinConnections="3"
          MaxConnections="10" MaxInUseConnTimeMins="30"
          MaxNotUsedConnTimeMins="1" MaxConnLifetimeMins="60"
          TxIsolation="2" useTxRecovery="no">
<Driver>com.rb.jdbc.oracle.OracleDriver</Driver>
<Url>jdbc:rollbase:oracle://localhost:1521;ServiceName=RB_DBO;
          ConnectionRetryCount=10;ConnectionRetryDelay=10</Url>
<DbUser>root</DbUser>
<Password>my_password </Password>
</Database>
```

This example shows how to use a JDBC driver and URL to connect to a SQL Server database:

```
<Database name="RB" isDefault="yes" isExternal="no"
          MinConnections="3"
          MaxConnections="10" MaxInUseConnTimeMins="30"
          MaxNotUsedConnTimeMins="1" MaxConnLifetimeMins="60"
          TxIsolation="2" useTxRecovery="no">
<Driver>com.rb.jdbc.sqlserver.SQLServerDriver</Driver>
<Url>jdbc:rollbase:sqlserver://localhost:1433;
          databaseName=RB_DBO;
          ConnectionRetryCount=10;ConnectionRetryDelay=10</Url>
<DbUser>root</DbUser>
<Password>my_password</Password>
</Database>
```

## Starting Components and Logging In

Before logging in to Rollbase, perform the steps appropriate for your operating system:

- [Starting Components on Windows Systems](#)
- [Starting Components on Linux Systems](#)

## Starting Components on Windows Systems

Follow these steps to start the private cloud components:

1. If necessary, start the database. If you used the Rollbase installer and had it install OpenEdge, the database should be running. You can verify by doing the following:
  - a) Open a command prompt. (On Windows systems, run it as administrator)
  - b) Change directories to the `oe_rollbase_db` folder of your installation. If you used the default location, this will be `Progress\WRK\oe_rollbase_db`.
  - c) If the database is running, you will see a `rbdb.1k` file. If not, run the `startdb` script.
2. Start the Web server in one of the following ways:
  - If you used the Rollbase installer and installed PAS, open a command prompt as administrator, navigate to the `Pas_Instance\bin` folder and run `tcmam start`.

---

**Note:** The Pacific Application Server functionality described in the documentation will be available in early August in the release of Rollbase Private Cloud 3.0

---

- If you installed Tomcat yourself, open a command prompt as an administrator and run the `Tomcat startup.bat` script located in the `bin` folder of the Tomcat installation.
3. Enter the URL for your Web server in a browser. If you are using PAS, use `http://localhost:8830` or `https://localhost:8831`, which redirect you to the Rollbase login page. If you installed Tomcat using the default port, use <http://localhost:8080>.
- The Rollbase login page (or Tomcat home page if you are not using PAS) should display. If it does not, check to make sure that `JRE_HOME` is set properly and that you have administrative privileges.
4. Log in using the temporary password from the Rollbase welcome email. When Rollbase starts, this email is sent to the Administrative email user specified in the `shared.properties` file. Bookmark the URL for future use.
  5. Change the temporary password.
  6. From the **Applications** drop-down menu, select **System Console** and verify that all components are running.
  7. Verify that all applications described in [Private Cloud Applications](#) are successfully installed.

If you need to stop the OpenEdge database, run the `stopdb.bat` script in the work directory you specified during installation. If you need to stop Tomcat, run the `shutdown.bat` script for standalone instances, or stop the service.

## Starting Components on Linux Systems

To start the database and Tomcat on Linux you need to be running as root.

1. Start the database.
2. If you installed a new instance of Tomcat, set the `JAVA_HOME` environment variable to `$ROLLBASE_HOME%/jre`.
3. Start the Tomcat server. If you have installed a new instance of Tomcat, start it by running the `Tomcat startup.sh` script, which by default is found in `$ROLLBASE_HOME%/apache-tomcat-7.0.42/bin`.
4. Enter <http://localhost:8080> in your browser.

The Tomcat welcome page should display. If it does not, check to make sure that `JRE_HOME` is set properly and that you have administrative privileges.

5. Log in using the URL and temporary password from the welcome email. This email was sent to the Administrative email user specified in the `shared.properties` file. Click the **Refresh** link if no data is shown.
6. Change the temporary password.
7. From the **Applications** drop-down menu, select **System Console** and verify that all components are running.
8. Verify that all applications described in [Private Cloud Applications](#) are successfully installed.

## Activating Your License

You will receive your Rollbase license through email. Progress recommends saving a copy. If you are upgrading from an evaluation license, you need to edit configuration files to use the host name you supplied when purchasing the license. If you are simply renewing a license for the same host name, you can upgrade the license without restarting.

The following sections describe how to activate or upgrade a license.

- [Upgrading from an Evaluation License](#)
- [Upgrading a License Without Restarting](#)

## Upgrading from an Evaluation License

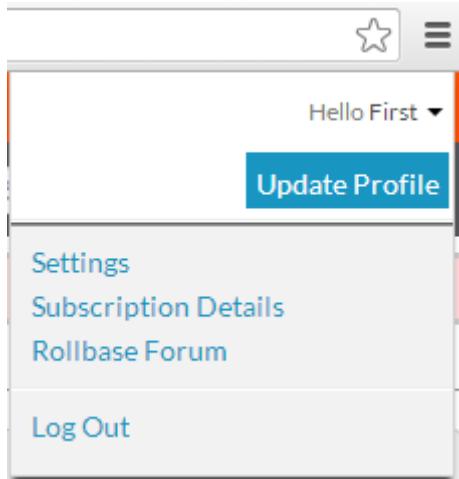
To upgrade Rollbase Private Cloud from an evaluation version, follow these steps:

1. If you evaluated Rollbase in local environment using `localhost` as the host name, follow these steps to use an external host name:
  - a) In `components.xml`, modify the URLs to point to the correct host name.
  - b) In the `shared.properties` file, modify the `HostName` entry to resolve `#HOST_NAME` tokens with the correct host name.
2. Copy the `license.xml` file into the `config` directory.
3. Stop and restart PAS or your Web server.

## Upgrading a License Without Restarting

To activate a license without restarting:

1. Log in to Rollbase.
2. In the top right of the screen, click the arrow next to your profile and select **Subscription Details**.



3. In the **Your License** section, click **Update**.
4. Click **Choose File**, navigate to the location of your `license.xml` file, select it and click **Next**.
5. Confirm that you want to update your license.

Your license will be updated without restart.

## Troubleshooting

The following topics describe some common problems and what you can do to resolve them:

- [Installation Issues](#) on page 463
- [License Errors](#)
- [Email Issues](#)

## Installation Issues

When the Rollbase installer is running, it logs messages to the standard output `stdoutXXX.log` file (or `hostname.XXX.log`) in the Tomcat `log` directory (if Tomcat is running as a service) or to the application's terminal window. For normal, error-free startup you should see output similar to the following (omitting some Tomcat-generated messages):

```
==>> Master Server is starting
ROLLBASE_HOME=c:\rollbase\shared
Host name: localhost:8080
Release: 4.07
Master Server: Initialization completed successfully

==>> PROD1 Server is starting
Production Server PROD1: Initialization completed successfully

==>> REST server is starting
==>> Router Server is starting
==>> RSS server is starting
==>> SEARCH Server is starting
==>> STORAGE Server is starting
```

```
==>> WEBAPI Server is starting
```

These log messages are important for diagnosing installation and setup issues. Please include them in any support request related to Rollbase Private Cloud installation.

If you encountered an error during installation, cannot start or login into your Rollbase server, the following issues could exist:

Issue	Resolution
The ROLLOBASE_HOME environment variable is not set or is pointing to the wrong directory.	Make sure that the ROLLOBASE_HOME environment variable is set and pointing to the correct directory. See <a href="#">Set Environment Variables</a> on page 454
The Tomcat server was not stopped while WAR files were copied by you or the installer into the Tomcat webapps directory.	Stop Tomcat, delete the files from the Web Server deployment directory, as well as temporary files which may have been created, including JSP cache files from the working folder), recopy the WAR files and restart Tomcat.
The host name specified in the shared.properties, databases.xml and components.xml configuration files does not match the actual host name you're using.	Update the configuration files with the correct host name.
The shared.properties file contains invalid email credentials	Update the configuration file with valid email credentials.
The version date of rb_util.jar in the Tomcat lib directory is inconsistent with the version and or date of the Rollbase WAR files.	If you installed manually, confirm that you unzipped the Rollbase lib.zip into the Tomcat lib directory.
Database issues	Drop the rb_dbo table by running the SQL command: drop database rb_dbo;
The WebAPI server is not running when you are logged into the Master Server and viewing Rollbase Private Cloud component status.	Delete the Tomcat webapps/webapi directory and restart Tomcat.
An error message mentions Java "PermGen space".	Restart the server.

If problems persist, feel free to use the Rollbase Community forum to ask questions and interact with other customers. However, if the problem is related to the specifics of your local environment, you will probably need to involve your IT staff.

## License Error

If you attempt to change the content of a license file or your license has expired, you will experience the following:

- You still can start and run Rollbase.

- Every page displays the notification **License Expired or Invalid**.
- You can no longer create Customers (tenants), but you can delete them.
- You can no longer create or update user accounts.
- You can no longer create or update object records.

## Email Issues

If you cannot send emails from your Rollbase Private Cloud instance please check the following:

- Make sure that all email-related global settings in the `shared.properties` file are correct.
- If you're behind firewall (corporate or local), make sure that firewall allows outgoing SMTP connections.
- If you are using Gmail, please make sure that POP/IMAP Access is enabled in your Gmail settings.

If you still cannot successfully send emails, add a `SkipEmails=true` setting temporarily. This will dump all emails to standard output (console window of Tomcat log file). This also allows you to recover system-generated passwords for new users.

## Logging In

If you are having trouble logging in to Rollbase and you are sure that the web server is running:

- Log in using the Admin email address specified in your `shared.properties` file as the user name.
- The first Administrative user's password is always set to `welcome`. You should change this password on first login.

If you run into a problem please fix your email settings, restart Tomcat and use the **Forgot Password** link (available from the `login.jsp` page). The system will reset your password and send another email to you.

In addition to creating the first administrative user account the system will create a second administrative account. Use this account if you're having problem to login as first administrative user. Delete or disable this account when you no longer need it:

- User name: `admin2`
- Password: `welcome`

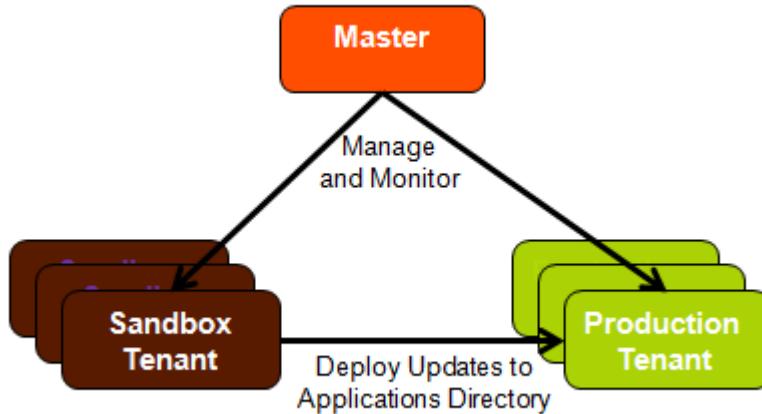
# Administration

After installing Rollbase Private Cloud and performing the steps described in [Starting Private Cloud Components](#), you can begin using the master tenant. From the master tenant, administrative users can set up and manage customer tenants and the Rollbase system. The topics in this section provide an overview of administration and describe how to perform common tasks.

The procedures described in this section are the same for single server and multi-server environments. However, to have access to full administrative capabilities, you need to log in to the master tenant.

## Development Sandboxes and Production Tenants

Progress recommends that Private Cloud installations have at least one customer tenant to use as a sandbox for development and testing. To set this up, create a **Customer** object and add developers as users. Once applications are ready, you can use the XML publishing mechanism or the **Application Directory** portal to distribute applications to other customer tenants.



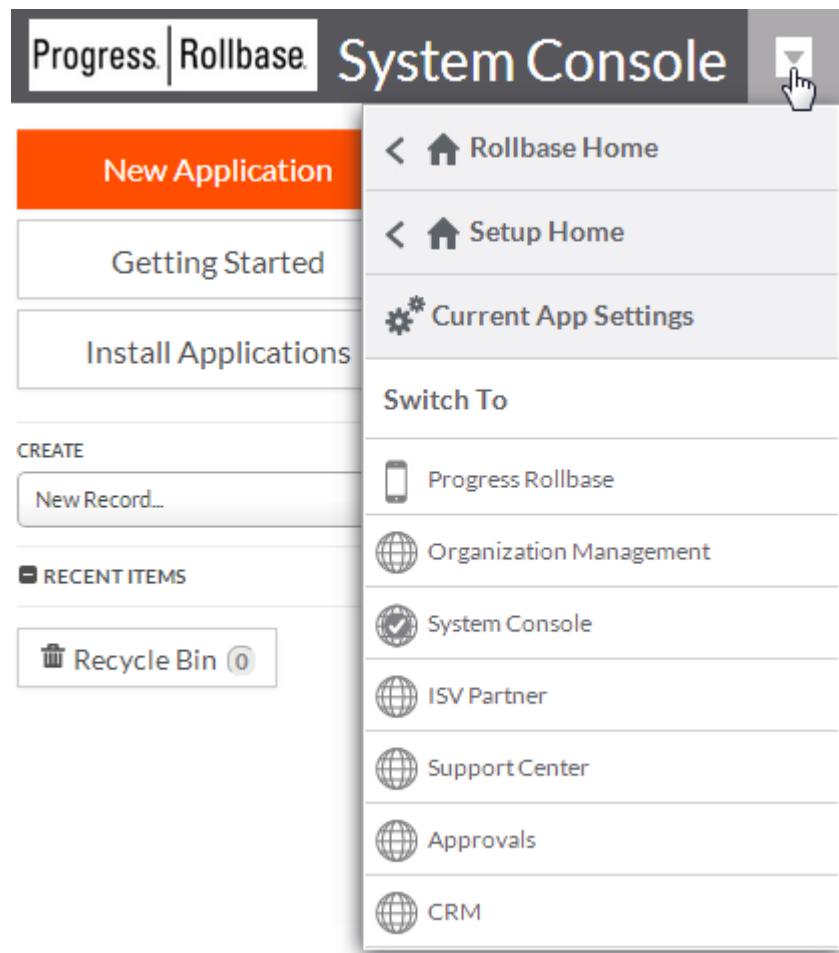
## Overview

Administrative users who set up customer tenants determine which applications are visible in each tenant. In addition, the Rollbase interface is highly customizable, allowing you to replace the Rollbase logo and use your own CSS styles.

When new users install Rollbase Private cloud, the interface appears identical to the Progress-hosted Rollbase interface, with the Pacific header and footer. However, you can remove these by editing `shared.properties` and setting the `PacificUIDisabled` property to true.

**Note:** If your Rollbase environment is highly customized, the interface might differ slightly from the screens shown in this documentation..

**Setup Home**, available from the applications drop-down menu, provides access to administrative settings for users with administrative privileges and to pre-installed applications:



In addition to any applications you install, menu items include:

- **Home** — the Rollbase application that contains the **Calendar** and **User** objects that you use to create user accounts.
- **Setup Home** — a short cut to setup.
- **Current App Settings** — a short cut to setup for the currently open application.
- **Organization Management** — contains **Location**, **Department**, **Function**, and **Group** objects, which allow you to set up complex access control. See [Location/Department/Function Permissions](#) on page 402 for more information.
- **System Console** — contains objects for managing customer tenants, subscribers (users of customer tenants), and applications in the **Application Directory**.
- **ISV Partner** — contains objects for managing ISVs and partners who will resell applications.
- **Support Center** — contains objects for managing support of your customer tenants.
- **Approvals** — contains objects for adding approval workflows to your applications.
- **CRM** — the sample CRM application that is also available in the public cloud. You can customize this application to manage your own customers.

# Monitoring System Components

From the **System** tab of the **System Console** application, you can view the current status of all Rollbase system components.

Status	Component	Threads	Connections	Customers	Sessions	Time Running	Jobs
Running	Master Server [localhost]	38	9	1	1	3h48m	0/0
Running	Production Server 1 [localhost]	38	9	0	0	3h47m	0/0
Running	REST Server [localhost]	38	9	0	0	3h47m	0/0
Running	RSS Server [localhost]	38	9	0	0	3h47m	0/0
Running	Router Server [localhost]	38	9	0	0	3h47m	0/0
Running	SOAP Server [localhost]	38	9	0	0	3h47m	0/0
Running	Search Server [localhost]	38	9	0	0	3h47m	0/0
Running	Storage Server [localhost]	38	9	0	0	3h47m	0/0
Running	Workflow Server [localhost]	38	9	0	0	3h47m	0/0

Action	Status	Type	Pool	Database	Is Master	Is Default	Connections	Customers	Records
Query   Edit	Running	OpenEdge	Rollbase	Database RB [localhost]	✓	✓	9	1	91

System | Customers | Subscribers | Published Apps

From this page you can:

- Click a production server name link (if enabled) to get more detailed information, including a list of currently loaded customers and logged in users, as well as a list of import, batch, and customer creation jobs currently running and in the queue. From the production server page you can do the following:
  - Check whether a customer tenant is loaded or unloaded.
  - Click the link to force load or unload of a particular customer from the production server cache. If you change settings such as expiration date, limitations, while the customer tenant is in use, you need to unload and load in order to pick up the new settings.

**Note:** Unloading a customer tenant from the production cache will log out all users currently logged into that tenant, including any API sessions.

- Click a link in the **Threads** column to check the current status of all system Java threads for that component.
- Click a link in the **Connections** column to check the status of all database connections.

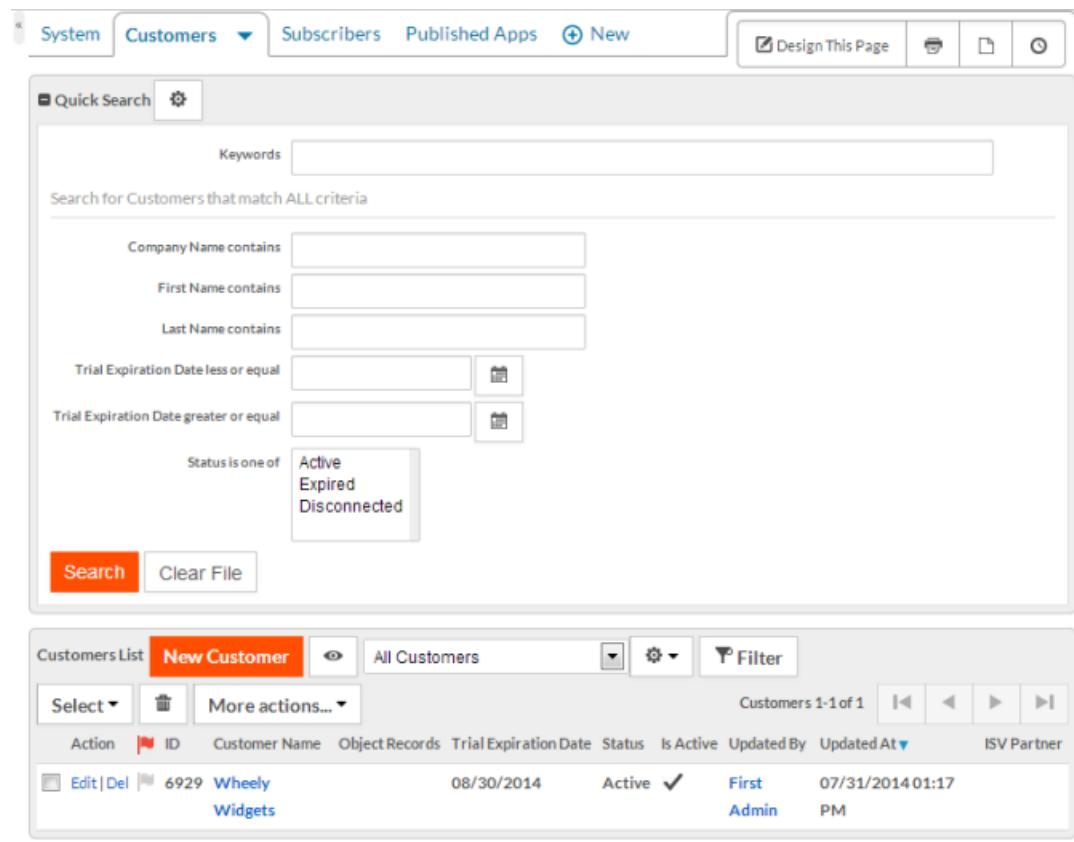
From sub-menus of the **System** tab you can:

- Monitor logs for the Master Server and All Customers (`AllJobs.log` and `AllErrors.log` are particularly important). Log operations are performed in asynchronous mode. There may be a slight delay between when an event is logged and it is written to the file and is viewable.

- Re-index the Master Server and/or a group of selected customers.
- Set the ID of an object definition for debugging purposes: All SQL queries used to list records of this definition will be logged in the `query.log` file.

## Managing Customer Tenants

From the master tenant **System Console** application, administrators can create and manage **Customer** records. Each customer record represents a tenant — an independently-managed collection of applications and services. The **Customer** record determines which applications users of that customer tenant can access, where their data will be stored, the level of security, and more. From the customer record, master tenant administrators can log into the customer tenant with super-admin privileges.



The screenshot shows the Rollbase System Console interface for managing customer tenants. The top navigation bar includes tabs for System, Customers (selected), Subscribers, Published Apps, and a New button. To the right are links for Design This Page and various system icons.

The main search interface (top half) features a "Quick Search" section with a "Keywords" input field and a "Search for Customers that match ALL criteria" section. This section includes fields for Company Name, First Name, Last Name, Trial Expiration Date (less or equal and greater or equal), and Status (Active, Expired, Disconnected). Below these are "Search" and "Clear File" buttons.

The bottom interface (bottom half) shows a "Customers List" table. The table has columns for Action, ID, Customer Name, Object Records, Trial Expiration Date, Status, Is Active, Updated By, Updated At, and ISV Partner. A single customer record is listed:

Action	ID	Customer Name	Object Records	Trial Expiration Date	Status	Is Active	Updated By	Updated At	ISV Partner
<a href="#">Edit</a>   <a href="#">Del</a>	6929	Wheely Widgets		08/30/2014	Active	✓	First Admin	07/31/2014 01:17 PM	

The **isActive** formula field determines whether users can log in to the customer tenant. The default implementation for the **IsActive** formula sets the **Status** of newly created customer records to **Active**. You can change this formula if desired.

Some fields of **Customer** records, like the address, can be edited by administrative users of the customer tenant through their **Account Setting** page.

If you delete a **Customer** record, all records and hosted files from that tenant will be deleted. Deletion cannot be reversed unless you have a backup file. You can restore customer data from a backup file created on a different Rollbase server, see [Moving and Restoring Customer Tenants](#) on page 473.

The **Subscribers** tab of the **System Console** application lists all users in all customer tenants. You cannot create these records directly. Rollbase creates them automatically when an administrator in the master or any customer tenant creates a **User** record. **Subscriber** records are useful for marketing, communication and other purposes such as sending mass emails.

## Creating a New Customer Record

A **Customer** record defines the applications, security level, storage information and usage for an organization or group that will be using a customer tenant.

Follow these steps to create a new **Customer** record.

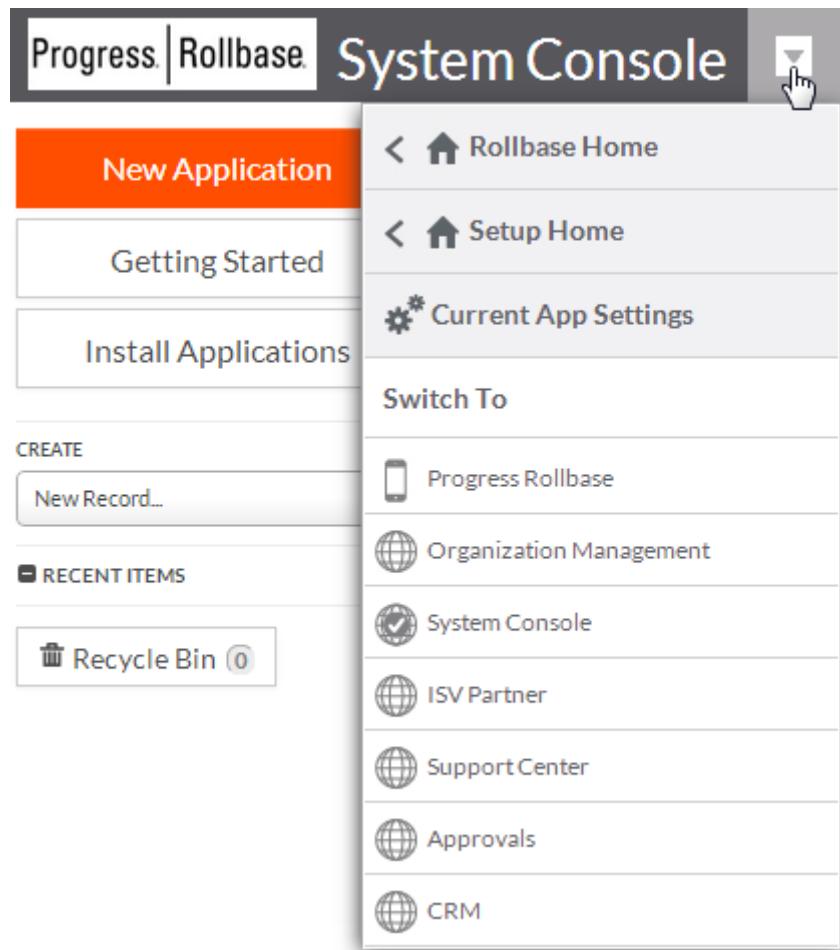
1. Log in to the Private Cloud master tenant as an administrator.
2. From the drop-down menu of applications, select **System Console**.
3. Select the **Customers** tab.
4. In the **Customers List** section, click **New Customer**.
5. Enter the following information:
  - **Database** — select the database to hold this customer's data.
  - **Plan** — select a pricing plan to determine default limits on the number of records, fields and other resources allowed for this customer. See [servicelevel.xml](#) on page 504 for information on how to customize these plans.
  - **Security Level** — select the security level for the customer tenant. The customer can later change this from their **Account Settings** page.
  - **Applications** — select the applications to be installed into the new customer tenant. If this field does not appear on the **New Customer** page, click **Design This Page** and drag **Applications Lookup** onto the page from the **Available Components** list. The **Rollbase** application will be included by default. It is required, do not remove it.
  - **Email** — specify the email address for the first administrative user for this customer tenant.
  - **Storage server** — if more than one storage server is installed, select the storage server assigned to this customer.
  - **Search server** — if more than one search server is installed, select the search server assigned to this customer.
6. Click **Save** to create the record, or click **Save and New** to save the record and create another **Customer** record.

After several seconds or minutes (depending on the speed of your server) the system will finish the tenant creation process by installing all selected applications. Until tenant creation is completed, the **Login** button on the **Customer** view page is disabled. The page will be refreshed automatically when process is completed and the button will be enabled. After this is finished, Rollbase sends a welcome email to the first administrative user, who will then be able to log in to the new tenant. When an administrator of the master tenant logs in to a customer tenant, they have **super-admin** privileges.

## Working with Customer Records

To view and work with **Customer** record details, Rollbase administrators of the master tenant can do the following:

1. Navigate to the **System Console**:



2. Select the **Customers** tab.
3. In the **Customers List** area, click the **Customer Name**.
4. Do any of the following operations:
  - **View Logs:** View log files associated with this customer, which can be useful for troubleshooting and debugging. Requires view permission
  - **Login:** Log into a customer tenant as a Super-Admin (invisible user with full access). This option requires log in permission. The button only appears after the administrative user of the customer tenant enables access for the master tenant users. For more information about enabling support access, see [Enabling an administrative user to log into a customer tenant](#) on page 406.
  - **Edit:** Modify the customer record. Requires edit permission.
  - **Delete:** Delete the customer record and all customer's data. Requires delete permission.

From the **More Actions** drop-down menu, the following are available:

- **Login As:** Log into a customer tenant as a particular user. This option requires log in permission. The button only appears after the administrative user of the customer tenant enables access for the master tenant users. For more information about enabling support access, see [Enabling an administrative user to log into a customer tenant](#) on page 406.
- **Data Maintenance:** Use this procedure to restore the integrity of relationships for this customer. This is only for fixing problems and will not be used under normal conditions. Requires edit permission.
- **System Backup:** Monitor and create backup files created in this customer. Requires view permission.
- **Restore from backup:** (from Backup page) Use the **Restore** option to delete all current customer data and replace it with data imported from the selected backup file (no users can be logged in during backup). Requires edit permission.
- **Database Move:** Moves customer data to another (selected) database. Requires edit permission. Additionally, after performing **Database Move**, you must navigate to the customer's details page, under **Runtime Information** area, select **Load** to load the customer tenant from the production server cache.

---

**Note:** You unload and load the customer tenant for the new settings to take effect. Only after loading the customer tenant from the production server cache can the users log into the customer tenant.

---

The following buttons are available:

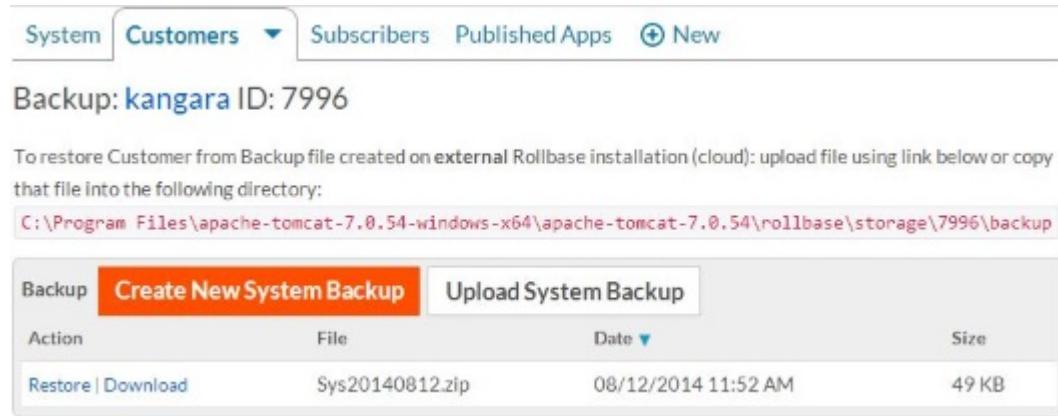
## Moving and Restoring Customer Tenants

You can move customer data for a customer tenant from one Rollbase Private Cloud installation to another for backup and restoration. The restoration process will create database records from information stored in the backup file using unique IDs created on the originating server. These IDs may potentially conflict with IDs already present in destination database on the target server. Therefore, Progress recommends that you use a fresh empty database to restore customer tenant data created on another server.

It is important to note that although moving customer data between servers is technically possible, the following procedure may be error prone and may impose additional limitations.

1. Create and download a backup of the customer on the source server (see [Working with Customer Records](#) on page 471).
2. Create and register a new empty database (see [Adding a New Database for Use with Customer Tenants](#) on page 476).
3. Create a **New Customer** on the target server assigning the newly created empty database to it (see [Creating a New Customer Record](#) on page 470).
4. Copy the backup file from the source server to the target server or upload the downloaded backup file on your target server. You will find the destination location on **System Backup** page. For information on navigating to the System Backup page and uploading the system backup, see [Working with Customer Records](#) on page 471.

After copying or uploading the customer backup, your backup will be listed in your **Backup** page:



Backup: [kangara ID: 7996](#)

To restore Customer from Backup file created on external Rollbase installation (cloud): upload file using link below or copy that file into the following directory:

C:\Program Files\apache-tomcat-7.0.54-windows-x64\apache-tomcat-7.0.54\rollbase\storage\7996\backup

Action	File	Date	Size
<a href="#">Restore</a>   <a href="#">Download</a>	Sys20140812.zip	08/12/2014 11:52 AM	49 KB

5. Select the **Restore** link to start the restoration process.

**Note:** If the original customer tenant has lots of data in uploaded files, the backup file will not include these files. In this case, you must move your uploaded files into your cloud storage or copy those files manually from the original server to the target server.

When the process is complete, you will receive a confirmation email from Progress Rollbase. Note that you can check the `backup.log` file on your customer for information about the restoration process and possible errors.

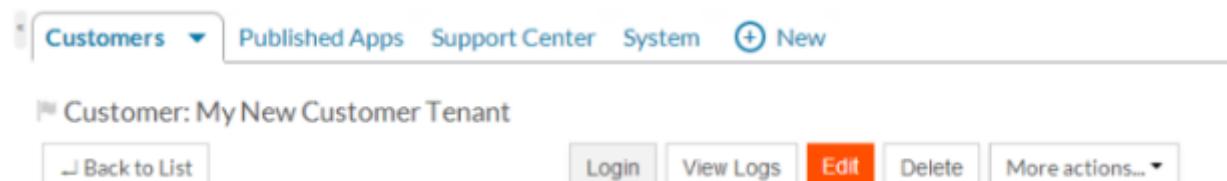
## Enabling Logging for Charts and Views

Administrators of the master tenant can enable logging to debug charts and views for a customer tenant. The log will contain the SQL statements and results to help with troubleshooting.

**Note:** Logging is limited to only one chart or view at a time. If logging is enabled already, and you enable logging on a second chart or view, the setting on the first will be automatically disabled.

To enable logging for a chart or view, follow these steps:

1. Navigate to the customer record for the tenant and click **Login**.



Customer: My New Customer Tenant

[Back to List](#) [Login](#) [View Logs](#) [Edit](#) [Delete](#) [More actions...](#)

2. In the customer tenant, navigate to the object definition that contains the chart or view.
3. Navigate to the **Charts** or **Views** section.
4. Click **Edit** for the chart or the view.

The **Edit** page opens:

Application Setup > Objects > Room > All Rooms

### Room: Edit View: All Rooms

**View Name**

Provide a name for this view.

View Name

Hide this View from View Selector

Hide count of total records for this View

Private  (Private Views are only available to their creator)

**Columns**

Select the fields to display in each column of this view.

Available Columns

Country  
Created At  
Created By  
Devices  
Furnishings  
ID  
Reservation  
Street Address 2  
Updated At  
Updated By  
ZIP/Postal Code

Selected Columns

Room  
Street Address 1  
City  
State/Province  
Phone Number

Show "Actions" column in this view

5. Scroll down to the **Debug** section:

**Permissions**

User Roles   <a href="#">Select All</a>   <a href="#">Select None</a>	
Administrator	<input checked="" type="checkbox"/> View
Portal Visitor	<input type="checkbox"/> View

Individual Users | [Select User](#)

**Debug**

Log all SQL queries created by this View in `query.log`

6. Click to check the box: **Log all SQL queries created by this <View or Chart> in `query.log`**.
7. Click **Save**.

After exercising the view or chart, for example by creating a new record, return to the edit page where you enabled logging and click `query.log` to open the log file.

## Managing Databases

The `databases.xml` file in the `config` directory of the Rollbase Master Server instance specifies configuration for the database(s) used by Rollbase. The configuration includes connection information that determines which database will store data for a particular customer tenant. If you change the file manually, you must restart the Web server. An administrator logged into the master tenant can create and update system databases from the **System Console** application without restarting the Web server.

The topics in this section cover the following procedures:

- [Adding a New Database for Use with Customer Tenants](#) on page 476
- [Creating Custom Database Indexes](#) on page 478
- [Adding Columns to a Private Cloud Database](#) on page 478

### Adding a New Database for Use with Customer Tenants

Before adding a database, you must install it and create Rollbase tables using the steps described in the Installation section under [Databases](#). You also need to have a JDBC driver and add its JAR file to the Web server library. For example, for a PAS installed in the default location, this is the `Progress\Rollbase\Pas_Instance\common\lib` folder.

To add a database to a Private Cloud system, follow these steps:

1. Log in to the master tenant as an administrator.
2. From the drop-down menu of applications, select **System Console**.
3. Locate the **Databases** list on the **System** tab and click **Add**.

The **Create Database** screen displays.

System ▾ Customers Subscribers Published Apps + New

Create Database

Save Cancel

Database Info

Database Name

Database Type -- Please select --

JDBC Driver

Database URL

User Name

Password

Test Database Connection

Use this database as default for new Customers

This database includes External tables which can be mapped as system objects

Connections Pool Options

Min Connections 1

Max Connections 100

Max Time In Use 30 Minutes

Max Idle Time 30 Minutes

Max Life Time 60 Minutes

New Connection Timeout Seconds

Transaction Isolation Default

4. Enter values for the following required fields:

- **Database Name:** must be unique and contain only alphanumeric characters
- **Database Type:** OpenEdge, MySQL, Oracle, or SQL Server. For Oracle and SQL Server, you have a choice to use a DataDirect JDBC driver.
- **Database URL:** When you select a database type, this field populates with an example URL. Substitute your host name and schema name, or if you are using a different driver, provide the URL to that driver.
- Enter the user name and password Rollbase will use to access the database.

5. Click **Test Connection**.

If the test is successful, the **Save** button will be enabled.

6. Set any of the desired options for this database and its related connections pool:

- Whether to use this database as default for newly created customer tenants.
- Whether this database contains external tables which can be mapped to Rollbase Objects.
- Minimum number of connections in the pool.
- Maximum number of connections in the pool.
- Maximum time (in minutes) allowed database connection to be in use. The connection will be forced to close after the maximum time has elapsed.
- Maximum time (in minutes) allowed database connection in a pool to be idle. The connection will be closed after the maximum time has elapsed.
- Maximum connection lifetime before closure (in minutes).

- Transaction isolation level. Consult your database manual. Use the **Default** option if you are not sure about this setting.

#### 7. Click **Save**.

After saving the information you entered, a new database will be displayed in the list. The `databases.xml` file will be properly updated on all servers in your Private Cloud without a Web server restart.

After you have added a database, you can click **Edit** to change editable parameters. The name and connection information are not editable.

## Creating Custom Database Indexes

Experienced database administrators can create or modify indexes to tune performance. Database indexes can improve application performance because they reduce the processing required for certain queries. Indexes should be used only after analyzing which SQL queries in an application are slow because of a database bottleneck.

Rollbase object records are stored in `RB_OBJ_DATA` table, which has more than 500 columns. Some of these columns are already indexed to improve performance of SQL queries. You can identify the indexed columns by searching the database creation scripts for lines including the `CREATE INDEX` command. For example:

```
CREATE INDEX RB_OBJ_DATA_I17 ON RB_OBJ_DATA(OBJ_DEF_ID, STR2);
```

It is neither practical nor possible to index all columns in the `RB_OBJ_DATA` table. It is best to concentrate on columns with large numbers of records per object. Indexes can be created for single or multiple columns. For example, if your application uses a field mapped to a `STR10` column to sort and filter large amount of records, you could create an index on that column. The following database command would create such an index:

```
CREATE INDEX RB_OBJ_DATA_CUST1 ON RB_OBJ_DATA(OBJ_DEF_ID, STR10);
```

## Adding Columns to a Private Cloud Database

There are limits on the number of fields you can create per object definition. For instance, you cannot create more than 50 fields of **Date** or **Date/Time** type. Rollbase uses the single wide database table, `RB_OBJ_DATA`, to store all object records. That table has 50 columns of **DATETIME** type (in MySQL terms) named from **DATE0** to **DATE49**. When a new field is created, one of these columns is assigned to that field to store data. If you are a Private Cloud customer and have full control over your database, you can increase limits on the number of fields.

To add columns:

- Use an SQL script to create more columns of the selected type to the following tables:
  - `RB_OBJ_DATA` (main data table)
  - `RB_DELETED_OBJS` (records in Recycle Bin)
  - `RB_USER_DATA` (Rollbase users)
  - `RB_CUST_DATA` (customers)

- Make sure that the newly added columns follow the naming convention and use continuous numbering. For instance, you may create ten new Date/Time columns named DATE50, DATE51 to DATE59
- Make sure these new columns are added to all databases.
- Add entries to `shared.properties` to reflect the new limits on number of columns.

Property	Description	Default Value
MaxStrFields	Maximum number of VARCHAR(100) fields (in MySQL notation)	200
MaxIntgFields	Maximum number of BIGINT fields	150
MaxDblFields	Maximum number of DECIMAL(20,8) fields	50
MaxTxtFields	Maximum number of LONGTEXT fields	50
MaxDateFields	Maximum number of DATETIME fields	50

After server restart, you should be able to create more Date or Date/Time type fields per object definition.

## Applications Directory and Support Portal

Your Rollbase customers can publish their applications to the shared **Applications Directory**. Publishing an application will create a **Published Application** record in the **System Console**. The administrator of the master tenant manages this process by monitoring and updating **Published Application** records.

Apart from the fields which can be selected by the publisher, as an administrator in the master tenant, you can set the following fields:

- Check **Is System** only if this application explicitly includes the **USER** object. Progress recommends that you create your own system application that includes the **USER** object (similar to the Rollbase application) and publish it from a dedicated tenant.
- Check **Approved** to allow this application to be installed by any user browsing the **Applications Directory**.

Your customers can send feedback and report issues through the **Help > Support Tickets** menu. This will create a **Support Request** record in your master tenant **Support Center** application. You can use an existing workflow, or create your own actions to process these requests.

## Test Drive

You can set up dedicated customer tenants to provide Test Drive functionality for a particular application. To do that:

1. Publish the application in the **Applications Directory** and make it available using the **Publish** button on the application view page. Note: publishing is only available for the original application creator.
2. Publishing will create a record in **Published Apps** tab in the **System Console**. Edit that record and:
  - a. Check the **Approved** box.
  - b. Select the **Customer** where the published application was developed in the **Test Drive** lookup.
  - c. Verify that your published application appears on the **Applications Directory** portal.
3. Go back to the dedicated **Customer** and assign permissions for the **Portal User** user role. Progress recommends that you assign complete permissions for the **Test Drive** application.
4. Create a set of records to demonstrate functionality of your application. Only these records will be available for a **Test Drive** visitor.
5. Make sure that the **Test Drive** field is added to the **View Application** page on the **Application Directory** portal. You can also edit properties of the **Test Drive** field.
6. Open the **Application Directory** portal and locate the published application. On the view page you should see a **Test Drive** button. If **Test Drive** is configured correctly, this button will be enabled.

When a visitor to the **Applications Directory** portal clicks that button, they will be redirected into designated tenant without logging in. The visitor will have access to all tabs and objects according to permissions assigned to the **Portal User** role. The visitor will not be able to:

- Access administrative functions.
- Modify any data.
- Send emails, run import, etc.

## Set Up ISV Partners

If you wish to use Rollbase Private Cloud in an ISV model, you can create a new user record in the master tenant and assign them the **ISV Partner** role. This will give the user limited access to your master tenant through the **ISV Partner** application. See [Using the ISV Partner Application](#) on page 554

## Multi-server Environments

The performance of Rollbase is dependent on variables such as the following:

- Which database you use. Enterprise level databases, such as Progress OpenEdge or Oracle will provide better performance than MySQL in high usage scenarios.
- Large import jobs on the storage server use significant system resources. Running the storage server and production server on different PAS or Tomcat instances prevents users on the production server from being impacted by large import jobs.

- Devoting as much CPU and memory as possible for production servers and database servers will help increase performance.
- Low complexity applications can support up to 50 times more concurrent users than those that use a lot of formulas, templates, custom filtering, reports, and anything that is processor or database-intensive. You might be able to architect a system by creating different related applications rather than one all-inclusive application. This might allow you to limit the functionality that requires more processing to a smaller number of users.
- The fewer object definitions and applications in a particular customer tenant, the faster Rollbase can load it into memory when requested, and the smaller amount of memory it will require.

Rollbase includes a set of auxiliary runtime components which can be run on a single host for small loads or be deployed across multiple hosts for scaling purposes. As Rollbase usage grows, you can add instances of these runtime components to provide adequate performance for all users. When you run Rollbase production servers on multiple hosts, the router component calculates the least loaded server and directs newly logged in users to that server.

---

**Note:** Verify that system clocks and system time zones on all servers are synchronized, or the system will not be able to work properly.

---

## Planning Your Multi-server Architecture

Progress offers the following recommendations for scaling in a multi-server environment:

- Run database servers and PAS or Tomcat servers on different physical machines (not on virtual machines using the same physical server).
- Hosts for databases should have high performance CPUs and at least 8GB of RAM.
- Hosts for PAS or Tomcat should have a 64-bit OS and have at least 8GB of RAM each (the more the better; also make sure to allocate as much as possible to your Tomcat heap).
- If you are using Tomcat, be sure to install 64-bit versions of Java and Tomcat on each server.
- To provide load balancing, use an Apache server to perform the routing of requests to the Tomcat instances. You can have as many of these Apache servers as needed for load balancing.
- Progress recommends performing load tests with an initial version of your infrastructure. This can help you identify areas of slow performance and give you time to adjust before you have thousands of concurrent users using the system.

With powerful servers and a low complexity app, you should be able to handle anywhere from 250 to over 1000 concurrent users per server. However, this depends on the database and CPU demands of your Rollbase applications.

How you set up your environment depends on whether you are using PAS or configuring your own Apache Tomcat system:

- [Distributing Load with PAS](#) on page 481
- [Distributing Load with Apache and Tomcat](#) on page 487

## Distributing Load with PAS

The high level steps to use multiple servers with PAS include:

- Creating  $N$  number of production servers by creating new PAS instances as described in [Working with Instances](#) on page 482 and following the procedures described in [Adding Production Servers](#) on page 492.
- Configuring Customer tenants with the **Multi-Server Load** attribute so that requests from users will span multiple servers dynamically. When this setting is enabled, PAS and Rollbase distribute user sessions across all available production sever components.

**Note:** If you are using TCMAN to create and run multiple instances of the Master Server (those with the Master Web application in the webapps directory), you must copy the configuration files from the first Master Server to the others. All Master Servers must have identical configuration files.

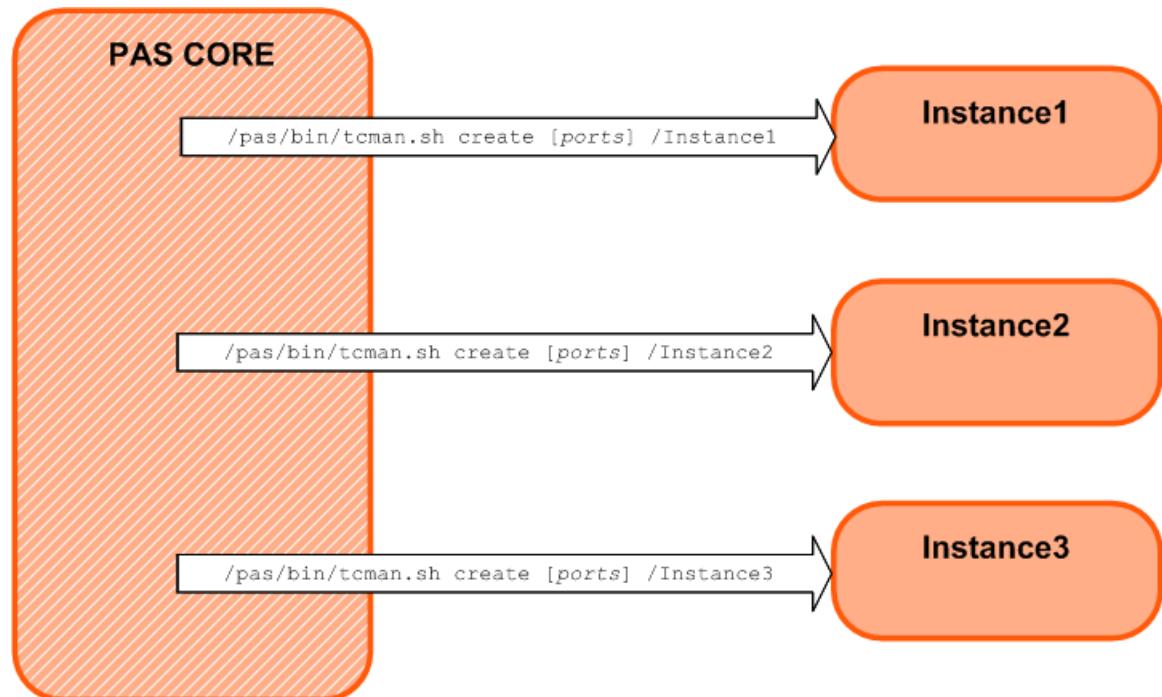
## Working with Instances

After you install the core Pacific Application Server, you can create an instance.

*Instances* are a standard Apache Tomcat feature. They allow you to create individual deployment and/or development servers that are based on the core Pacific Application Server that you installed.

The following figure illustrates the creation of multiple instances using the TCMAN command-line utility (with syntax simplified).

**Figure 3: Generating PAS instances**



Instances are independently running copies of the core Pacific Application Server. Each instance runs on its own JVM, has its own configuration with unique ports, and hosts its own web applications. However, each instance runs a Tomcat server that uses a number of common files from the same \$CATALINA\_HOME directory.

There are a number of advantages when you deploy your web applications to an instance of the Pacific Application Server, rather than deploying to the Pacific Application Server that you installed. This practice prevents accidental corruption of the core executables, configuration settings, and libraries. It also prevents accidental deletion of web applications if the core Pacific Application Server is removed when you uninstall a Progress PAS product.

Some additional advantages of instances are:

- Updates to the core Apache Tomcat server libraries and executables do not affect your web applications. You avoid the necessity of updating the applications and/or re-configuring them.
- You can establish different security policies for each of the instances.
- You can tailor the JVM for individual applications, since each instance runs in its own JVM with its own configuration.
- Instances provide you with quick way to create a test server for experimenting with new configurations and applications without the danger of permanently corrupting an existing server.
- You can package an instance as a Web application and deploy it to other PAS core servers.

You use `$CATALINA_HOME/bin/tcman.sh create` command to create a new instance.

When you create an instance, the root directory of the instance is assigned to the `CATALINA_BASE` environment variable within the scripts in its `/bin` directory. The root directory of the installed (core) Pacific Application Server is assigned to the `CATALINA_HOME` environment variable in the scripts in the instance's `/bin` directory. (Notice that the scope of these environment variables is limited to the context of an individual instance's `/binscripts`.)

All instances of a core Pacific Application Server execute a set of common JAR files, scripts, and libraries from the following directories on the parent server:

- `$CATALINA_HOME/lib`
- `$CATALINA_HOME/common/lib`
- `$CATALINA_HOME/bin`

However, each instance is created with :

- A `$CATALINA_BASE/bin/` directory with its own copy of some of the scripts from the core PAS. These include scripts for start up, shut down, deployment, running TCMAN actions, and so on.
- A `$CATALINA_BASE/conf/` directory with its own copy of properties and configuration files.
- A `$CATALINA_BASE/webapps/` which initially only contains the `ROOT` Web application.
- A number of directories that are initially empty. These include `/logs`, `/temp`, `/work`, and `/common/lib`.

## Creating instances with TCMAN

TCMAN is a Progress extension of the basic Tomcat administrative utilities that simplifies instance creation and management. A PAS instance runs the Tomcat executable of a core PAS, but it runs in a separate JVM, is configured with its own unique ports, and other properties. Using PAS instances allows you to run a variety of server configurations and to update the core server without re-deploying or re-configuring your Web applications.

---

**Note:** If you are using TCMAN to create and run multiple instances of the Master Server (those with the Master Web application in the `webapps` directory), you must copy the configuration files from the first Master Server to the others. All Master Servers must have identical configuration files.

---

To create an instance using the TCMAN utility:

1. Open a command shell and navigate to `$CATALINA_HOME/bin`.

`$CATALINA_HOME` is the directory where you installed the core Pacific Application Server.

2. Run `tcman.sh create basepath` (or `tcman.bat` on Windows systems).

The `base_path` parameter specifies the path name where you will create the instance. It is the only required parameter for the `create` action. If you are creating multiple running instances, you should override the default port assignments by specifying the following parameters:

Options	Description
<code>-p <i>port_num</i></code>	Specify the TCP port that listens for HTTP messages. The default is 8080.
<code>-P <i>port_num</i></code>	Specify the TCP port that listens for HTTPS messages. The default is 8443.

You can also activate these ports:

Options	Description
<code>-s <i>port_num</i></code>	Specify the TCP port to use to stop an instance. (Required on Windows systems, optional on UNIX.)
<code>-j <i>port_num</i></code>	Specify the TCP port that listens for AJP13 messages, an Apache protocol for handling requests from a web server to an application server. (Optional on both Windows and UNIX systems)

See [Create an instance \(create\)](#) on page 526 for information about other parameters.

3. (Optional) Deploy remote management applications from `$CATALINA_HOME/extras` to the instance.

Remote management applications are not pre-installed, and installing them is a security decision. For example, you might want to eliminate access to the configuration and control of instances by not deploying management applications to production servers, while deploying management applications to development servers.

To deploy a management application:

- a) Open a command shell and navigate to `$CATALINA_BASE/bin`.
- b) Run `tcman.sh deploy '$CATALINA_HOME/extras/admin_webapp.war'`.

The `admin_webapp.war` can be one of the following:

Options	Description
<code>host-manager.war</code>	A Tomcat administration application used to get server information and provide other functionality. It should not be necessary to deploy <code>host-manager.war</code> if you are using the TCMAN utilities.
<code>manager.war</code>	A Tomcat administration application which you must deploy in order to run some TCMAN actions. See the <a href="#">TCMAN Reference</a> for information on which TCMAN actions require deployment of <code>manager.war</code> .
<code>Progress applications</code>	Progress products can have web applications that enable the use of their own administrative tools.

For example the following command line creates an instance of `/psc/pashome` in `/psc/acme1` and specifies its ports:

```
$: /psc/pashome/bin/tcman.sh create -p 8501 -P 8601 -s 8701 /psc/acme1
Server instance acme1 created at /psc/acme1
```

## Instance management with TCMAN

TCMAN includes actions for configuring, starting, stopping, monitoring, and deleting instances.

The following table is a brief description of the instance management actions that you can perform with TCMAN. Entries link to the reference topics that provide more details, syntax, and examples.

Action	Purpose
<a href="#">create</a>	Create an instance of the Pacific Application Server.
<a href="#">delete</a>	Remove the directory tree and all of the files in an instance.
<a href="#">start</a>	Start an instance of a Pacific Application Server.
<a href="#">stop</a>	Stop a running instance.
<a href="#">config</a>	View, add, update, or delete the property values specified in <code>../conf/appserver.properties</code> .
<a href="#">test</a>	Displays information on the configuration and environment of an instance. It also displays information about error conditions.
<a href="#">instances</a>	Display all the instances created from the Pacific Application Server installed in <code>\$CATALINA_HOME</code> .
<a href="#">unregister</a>	Stop tracking an instance by removing the instance's entry from the <code>\$CATALINA_HOME/conf/instances.[unix windows]</code> file.
<a href="#">register</a>	Register an instance for tracking purposes. (Note that instances are registered for tracking by default when they are created. The register action is only necessary if you explicitly unregistered an instance.)
<a href="#">clean</a>	Truncate, move, or delete the log files located in the <code>/logs</code> directory of either the core server or an instance.
<a href="#">version</a>	Show the Apache Tomcat runtime version and OS information for an instance.

## Installing and running an instance as a Windows service

To install a Pacific Application Server (PAS) instance as a Windows service, you must have administrator privileges. On systems with User Account Control (UAC), you must disable UAC as well.

In addition, the instance must be registered with a core PAS server, which you can accomplish with the `tcman.bat register` action.

A service (called a daemon process on UNIX systems) is an application without a user interface that runs in the background and provides core operating system functionality. Web servers like PAS and Tomcat typically run as Windows services or UNIX daemons.

---

**Note:** If you run a PAS instance with `tcman.bat start`, the instance runs in the context of the command shell process. It is not available as a system service that can handle external client requests. The instance must be installed as a Window service before you can run it as a functioning Web server.

---

This is a summary of how to install and run a PAS instance as a Windows service:

1. Open a command prompt (`cmd.exe`) window.
2. Navigate to the core PAS `/bin` directory (`$_CATALINA_HOME/bin`).
3. Run the `service.bat` script using the following syntax:

```
service.bat install instance_name
```

`instance_name` is the name of an existing instance of the core PAS.

---

**Note:** See the *Windows service HOW-TO* help topic in the Apache Tomcat Documentation (<http://tomcat.apache.org>) for more information about installing instances as Windows services.

---

4. Use the Services Microsoft Management Console (MMC) or the `sc config` command to start the service.

---

**Note:** Refer to Windows help for more information on starting, stopping, and other administrative tasks with regard to Windows Services.

---

## Installing and running a PAS instance as a Linux daemon

A daemon process (called a service on Windows systems) is an application without a user interface that runs in the background and responds to requests. Web servers like PAS and Tomcat typically run as Windows services or Linux daemons.

---

**Note:** If you run a PAS instance with `tcman.sh start`, the instance runs in the context of the command shell process. It is not available as a system service that can handle external client requests. The instance must be installed as a daemon process before you can run it as a functioning Web server.

---

The file `$_CATALINA_HOME/bin/daemon.sh` can be used as a template for starting Tomcat automatically at boot time as a child of the `init` process. For more information, see:

[https://tomcat.apache.org/tomcat-7.0-doc/setup.html#Unix\\_daemon](https://tomcat.apache.org/tomcat-7.0-doc/setup.html#Unix_daemon)

However, you will need to consult with a system administrator before you can configure and run PAS as a daemon process due to differences among Linux systems and because you need administrative privileges for access to the system.

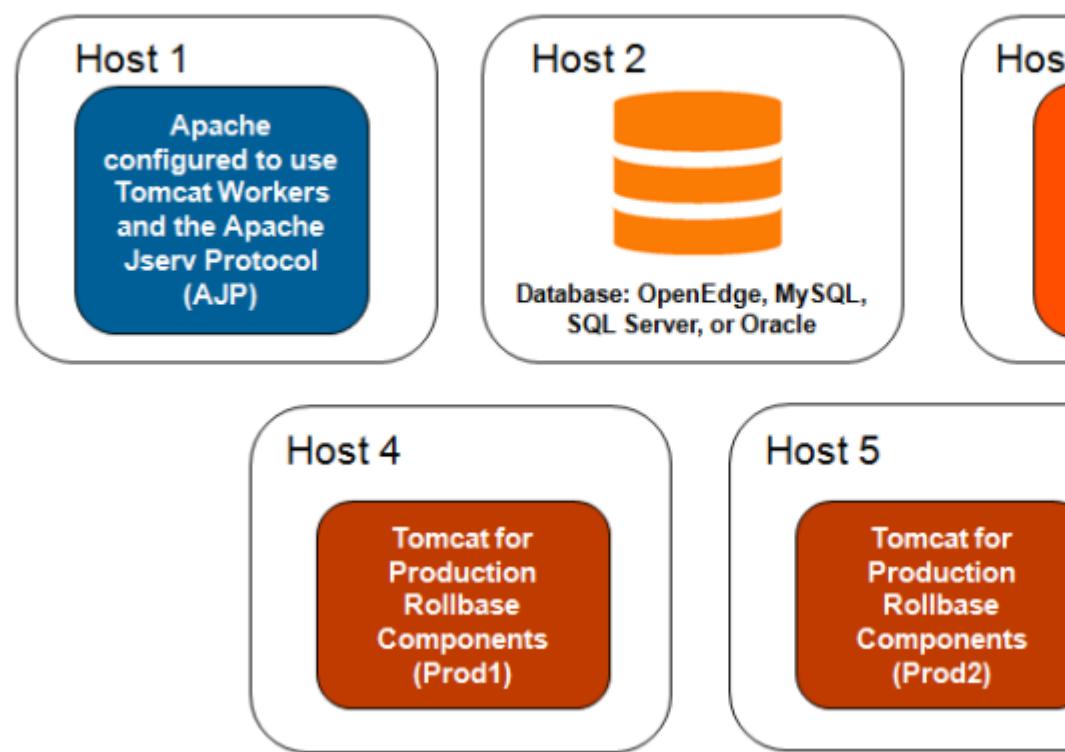
## Distributing Load with Apache and Tomcat

The following graphic illustrates a multi-server architecture using 4 servers:

- Server 1: Apache configured to use Workers
- Server 2: Dedicated to the database
- Server 3: Tomcat running a Master Server, Router Server and other standard Rollbase components
- Server 4: Tomcat running Production Server 1 (PROD1)
- Server 5: Tomcat running Production Server 2 (PROD2)

You can enable appliance-based load balancing in front of Apache servers if you have multiple Apache servers configured (which would be recommended with such a large number of users) but this should not be done for Tomcat servers.

## Multiple Server Architecture



The high level steps to use multiple servers include:

- Creating  $N$  number of production servers (by installing Tomcat and production server component on a virtual or physical machine).
- Configuring one or more Apache servers with Workers to recognize each production server.

- Configuring Customer tenants with the **Multi-Server Load** attribute so that requests from users will span multiple servers dynamically. When this setting is enabled, Tomcat and Rollbase distribute user sessions across all available production sever components.

To install Rollbase instances on multiple hosts, verify that the Tomcat instances are installed on every machine that will run Rollbase components. When copying WAR files, copy only the components that you wish to run on particular machine. Make sure Production Server instances get WAR names that match component names: for example, you would need to rename prod1.war after copying it to a different host. In the example shown above, the following WAR files should be copied:

- Host 3: master.war, rest.war, router.war, rss.war, search.war, storage.war, webapi.war, workflow.war
- Host 4: prod1.war
- Host 5: prod2.war (renamed version of downloaded prod1.war)

## Configuring Apache and Tomcat for Private Cloud

You can configure an Apache server as a front-end for your Tomcat instances. After you have installed Apache and each of your Tomcat instances, you will need to configure Tomcat Workers in Apache to tell it how to route incoming requests to Rollbase. Apache has extensive documentation on this at [http://tomcat.apache.org/connector-doc/generic\\_howto/quick.html](http://tomcat.apache.org/connector-doc/generic_howto/quick.html)

First verify that you have Tomcat installed on the same server as Apache. Then, follow these steps:

1. [Install the JK Module.](#)
2. [Create a workers.properties file.](#)
3. [Configure the httpd.conf file.](#)

### Install the JK Module

Verify that you have the JK module in your Apache modules directory. JK is the protocol used to communicate between Apache and Tomcat. You will need to locate and download the version of mod\_jk.so that applies to the version of Apache you are running. You can typically find the latest versions of mod\_jk.so here: <http://tomcat.apache.org/download-connectors.cgi/>

A typical location for the Apache modules directory on a Windows machine would be:

C:\Program Files\Apache Group\Apache2\modules\

### Create a workers.properties File

Once you have the JK module installed you will need to create a new file called workers.properties in Apache's conf directory (the same location that httpd.conf resides in). You can support any number of Production Server components in a multi-server architecture by pointing to them with worker properties as shown in the example below. You would duplicate the worker.prod1.x entries for each Production Server instance you have set up.

The `workers.properties` file should look something like the following. Note that each `worker.xyz` defines a worker named `xyz`. These workers will be referenced by this name in Apache's `httpd.conf` file.

```
# Tomcat home directory on same server as Apache
workers.tomcat_home=C:/Tomcat6.0
# Java home directory on same server as Apache
workers.java_home=C:/jdk6.0

# List of workers (defined in httpd.conf; one for each Tomcat)
worker.list=master,prod1

# The "master" worker is your Rollbase master server
# type should always be ajp13 (AJP13 is the protocol used to communicate
with Apache)
worker.master.type=ajp13
# host should be the network name or IP of the target server
worker.master.host=335090-web2
# port should be 8009 by default; do not change this unless you reconfigured
Tomcat to accept AJP requests on a different port
worker.master.port=8009
# The following parameters should not be changed without detailed
understanding of Tomcat workers
worker.master.lbfactor=1
worker.master.socket_timeout=0
worker.master.socket_keepalive=1
worker.master.connection_pool_timeout=60
worker.master.connection_pool_size=300
worker.master.connection_pool_minsize=50

# Create a "prodX" worker for each of your Rollbase production server Tomcats
as shown here
worker.prod1.type=ajp13
worker.prod1.host=335091-web2
worker.prod1.port=8009
worker.prod1.lbfactor=1
worker.prod1.socket_timeout=0
worker.prod1.socket_keepalive=1
worker.prod1.connection_pool_timeout=60
worker.prod1.connection_pool_size=300
worker.prod1.connection_pool_minsize=50
```

## Configure the `httpd.conf` File

Once you have installed `mod_jk.so` and created your `workers.properties` file, you need to configure your `httpd.conf` file to enable `mod_jk` and define your workers. To enable `mod_jk` add the following line to `httpd.conf`:

```
# Loads the Jakarta Tomcat connector protocol module
LoadModule jk_module modules/mod_jk.so
```

Add the following to define workers configuration:

```
# Tells the module the location of the workers.properties file
JkWorkersFile "C:/Program Files (x86)/Apache Software
Foundation/Apache2.2/conf/workers.properties"

# Specifies the location for this module's specific log file
JkLogFile "C:/Program Files (x86)/Apache Software
Foundation/Apache2.2/conf/logs/mod_jk.log"

# Sets the module's log level to info
JkLogLevel info

# Uses the below worker mounts for all virtual hosts; otherwise disabled for
SSL
JkMountCopy All
```

Now for each named worker in `workers.properties`, you need to specify the subdirectories that should be available in inbound requests. For the Master Server, Rollbase requires the following:

```
JkMount /workflow master
JkMount /workflow/* master
JkMount /storage master
JkMount /storage/* master
JkMount /webapi master
JkMount /webapi/* master
JkMount /search master
JkMount /search/* master
JkMount /rest master
JkMount /rest/* master
JkMount /rss master
JkMount /rss/* master
JkMount /master master
JkMount /master/* master
JkMount /router master
JkMount /router/* master
```

For each production server add the following code (make sure the name `prodX` corresponds to the name of each production server worker you defined in `workers.properties`):

```
JkMount /prod1 prod1
JkMount /prod1/* prod1
```

## Apache Troubleshooting

If you have trouble getting a multi-server environment to work, it is helpful to:

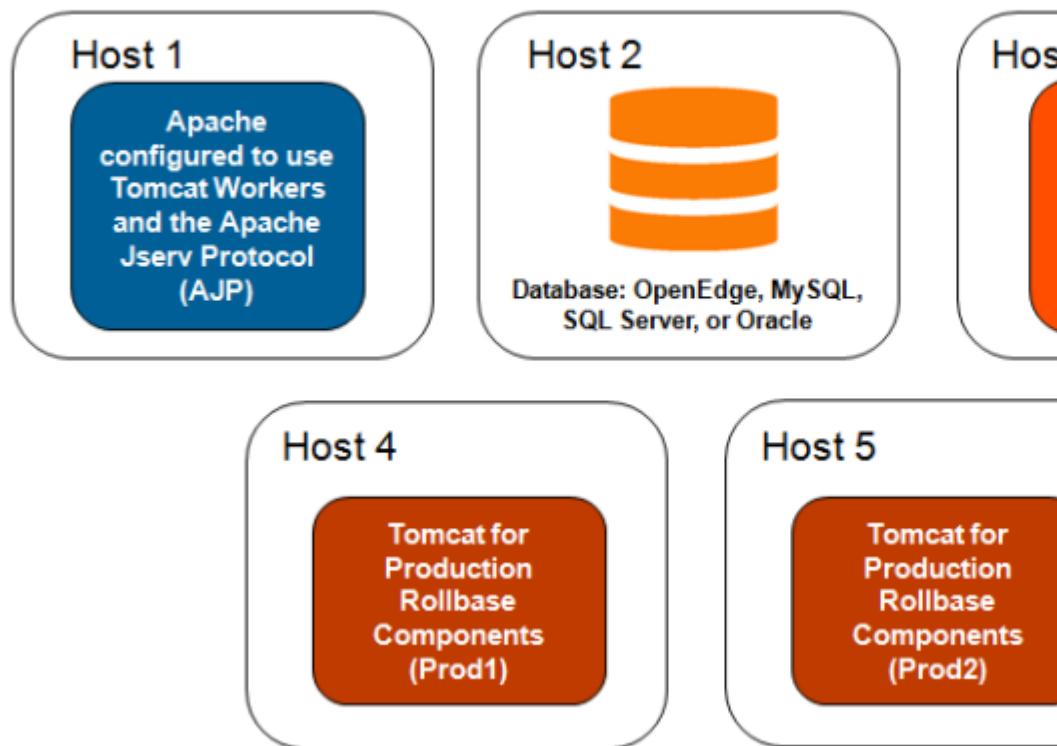
- Review the apache error logs for configuration errors.
- Review the `mod_jk` log file for JK related configuration errors.
- Review documentation at [tomcat.apache.org](http://tomcat.apache.org).

If all else fails, contact Progress through the Community forums or open a support ticket.

## Configuring Multiple Instances of Rollbase Components

With the appropriate license, you can distribute Rollbase components across multiple hosts. The architecture shown in the following diagram illustrates one way components could be distributed if you are using Apache and Tomcat:

## Multiple Server Architecture



Your license specifies the number of hosts that can run Rollbase software and must include the domain name provided by your web server as the host name. You should use this domain name for all external URLs in the `components.xml` file. The example configuration shown in the illustration requires a Rollbase license for three servers.

To set up the example architecture shown above, follow these general steps:

1. Install PAS or Apache and Tomcat as described in [Distributing Load with PAS](#) on page 481 or [Distributing Load with Apache and Tomcat](#) on page 487
2. Use the multi-server license provided by Progress for the Rollbase Master Server installation.
3. Install the components on their respective hosts.
4. Edit the `components.xml` file for the Master Server to include entries for all system components. Internal URLs must use IP addresses for the physical servers. For External URLs it is best to use `{ !#HOST_NAME }` token. See [The components.xml File](#) on page 497 for more information.
5. Configure your Web server (PAS, Tomcat, or Apache) to provide access to both production servers through a single external URL.
6. Start the database(s).
7. Start the Web server that is hosting the Rollbase Master Server instance.

8. Start the Web server on the hosts for production components.
9. Navigate to the **System Console** application **System** tab to verify that all of your components can be monitored from the Master Server. Make sure that production servers have the correct parameters.

When the system is running, you can add production servers up to the limit enforced by your license without stopping and restarting the Master Server. Production servers should only be started after they have been configured as described in [Adding Production Servers](#) on page 492. To add auxiliary components, you must stop and restart the Master Server as described in [Adding Auxiliary Servers](#) on page 493.

---

**Note:** If you are using TCMAN to create and run multiple instances of the Master Server (those with the Master Web application in the webapps directory), you must copy the configuration files from the first Master Server to the others. All Master Servers must have identical configuration files.

---

## Adding Production Servers

With the appropriate license, you can add production servers and Rollbase will automatically balance the load across them. The general steps for adding a production server include:

- Installing the PAS or Tomcat instance on the new host. See [Working with Instances](#) on page 482 or [Distributing Load with Apache and Tomcat](#) on page 487
- Installing the production server on the new host.
- Using the **System Console** application to add the configuration for the production server.
- Starting PAS or Tomcat.

To accomplish this, after you have installed PAS or Tomcat on the host, follow these steps:

1. Copy `prod1.war` from your Rollbase installation to the PAS or Tomcat instance `webapps` folder on the new host.
2. Rename `prod1.war` to have a unique number in its name.
3. Log into the master tenant with an administrative account.
4. From the drop-down menu of applications, select **System Console**.
5. Select the **System** tab.

If you have not reached the limit set by your license for production servers, you will see a button labeled **Add Prod Server**.

6. Click **Add Prod Server**.

The **New Prod Server** screen opens.

The screenshot shows a 'New PROD Server' configuration dialog. At the top, there are tabs for 'System', 'Customers', 'Subscribers', 'Published Apps', and a '+' button. The 'System' tab is selected. Below the tabs, the title 'New PROD Server' is displayed. At the top right are 'Save' and 'Cancel' buttons. The main area is titled 'Database Info' and contains five input fields: 'PROD Server Name' (with a placeholder 'PROD'), 'Display Name' (with a placeholder 'PROD'), 'Internal URL' (with a placeholder 'http://localhost:8080/rollbase'), 'External URL' (with a placeholder 'http://localhost:8080/rollbase'), and 'Power' (with a placeholder '1.0'). Below these fields is a checkbox labeled 'Check if the server is dedicated'. At the bottom are 'Save' and 'Cancel' buttons, and a footer navigation bar with links to 'System', 'Customers', 'Subscribers', and 'Published Apps'.

7. Enter values for the following (see [The components.xml File](#) on page 497 for more details on these settings):

- **Prod Server Name** — A unique internal name to distinguish the new server from others.
- **Display Name** — A unique name to display in the **System** tab for this server.
- **Internal URL** — The URL to the root of the production server as seen from inside your firewall.
- **External URL** — The URL to the root of web component as seen from outside your firewall.
- **Power** — A decimal number between 0.5 and 2.0 that indicates the relative computational power. Rollbase assigns higher loads to servers with a higher power value.
- **Check if the server is dedicated** — Optional checkbox. If checked, this production server will only handle the customer tenants you assign to it. See [Assigning a Customer to a Dedicated Production Server](#) on page 495.

8. Click **Save**.

The new production server will appear in the list of system components.

9. Start the new production server.

The `components.xml` configuration file will now include the new production server.

If you need to edit the production server settings later, stop the production server and click its name on the **System** tab.

## Adding Auxiliary Servers

In Rollbase Multi-server Edition, in addition to production servers, you can also run multiple instances of:

- Master Servers
- REST Servers
- Router Servers

- SOAP Servers
- Search Servers
- Storage Servers
- Workflow Servers

---

**Note:** Progress recommends that you install your Router, SOAP, and REST Servers on a single computer. This leads to faster REST and SOAP API execution.

---

Installation of multiple auxiliary servers is similar to the installation of multiple production servers:

- At most, one server of each type can be installed on a virtual or physical machine.
- Edit the `components.xml` file in the Master Server `config` directory:
  - `name` attributes must be unique.
  - `type` attribute must match the type of component
- Make sure that the name of the WAR file used to install every server matches the name used in `components.xml`. This might require renaming, as described in the example below.
- Copy the license file to the installation `config` directory on each host.
- On the host machines, define an environment variable for the type of server they will run as summarized below.
- When you are finished, restart the Master Server first and then the production and auxiliary servers.

Environment Variable	Description	Default Value
MASTER_SERVER	Name of Master Server as defined in the <code>components.xml</code> file.	MASTER
PROD_SERVER	Name of Production Server as defined in the <code>components.xml</code> file.	PROD1
REST_SERVER	Name of REST Server as defined in the <code>components.xml</code> file.	REST
ROUTER_SERVER	Name of Router Server as defined in the <code>components.xml</code> file.	ROUTER
SOAP_SERVER	Name of SOAP Server as defined in the <code>components.xml</code> file.	WEBAPI
SEARCH_SERVER	Name of Search Server as defined in the <code>components.xml</code> file.	SEARCH

Environment Variable	Description	Default Value
STORAGE_SERVER	Name of Storage Server as defined in the components.xml file.	STORAGE
WORKFLOW_SERVER	Name of Workflow Server as defined in the components.xml file.	WORKFLOW

## Example for Multiple REST servers

Consider an example where two REST servers are installed on two physical machines. The components.xml file describes them as follows:

```

<Component name="REST1" type="REST">
<DisplayName>REST1 Server</DisplayName>
<InternalRoot>http://server1:8080/rest/</InternalRoot>
<ExternalRoot>http://{!!#HOST_NAME}/rest1/</ExternalRoot>
</Component>

<Component name="REST2" type="REST">
<DisplayName>REST2 Server</DisplayName>
<InternalRoot>http://server2:8080/rest/</InternalRoot>
<ExternalRoot>http://{!!#HOST_NAME}/rest2/</ExternalRoot>
</Component>

```

The additional configuration required includes:

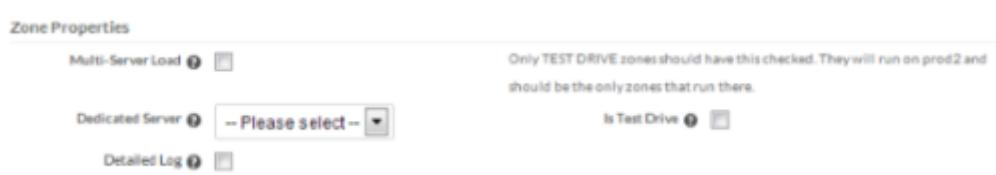
1. Setting a REST\_SERVER=REST1 environment variable on Server 1.
2. Renaming the WAR file on Server 1 to REST1.war
3. Setting a REST\_SERVER=REST2 environment variable on Server 2.
4. Renaming the WAR file on Server 2 to REST2.war
5. Setting up load balancing to divide incoming REST requests between Server 1 and Server 2.

## Assigning a Customer to a Dedicated Production Server

By default, customers in a multi-server environment are loaded dynamically to any available production server. An administrator logged into the master tenant can change this behavior by assigning any number of customers to a specific production server. To assign customers to a dedicated server:

1. Navigate to the **System Console** application.
2. Edit the production server in one of the following ways:
  - If you are setting up a new production server, follow the steps described in [Adding Production Servers](#) on page 492 and check the box to indicate the server will be dedicated to a subset of customers and skip to [Step 4](#)
  - If the production server exists and has not been configured to accept dedicated customers, follow the procedures in [Step 3](#):
3. To configure an existing production server:

- a) Stop the production server.
  - b) From the **System** tab, click the inactive production server's name.  
An error message displays.
  - c) Click the link in the error message to edit the server.
  - d) Check the **Check if the server is dedicated** checkbox.
  - e) Click **Save**.
4. Edit the customer records for the customers you want to assign to the production server:
- a) Select the **Customers** tab.
  - b) Find the appropriate customer and click **Edit**.
  - c) Scroll down to the **Zone Properties** section:



- d) From the **Dedicated Server** menu, select the production server to assign for this customer.
- e) Click **Save**.

## Load Balancing a Large Customer Tenant

With a multi-server license, you have the option to have large customer tenants (with large number of concurrent users) run on several servers simultaneously. To do that, create a check box with the integration name `multiServer` on your **Customer** object and make sure it is checked for your large customer tenants. The router will load large customer tenants on more than one server, depending on current load. When administrators modify metadata (create objects etc.) these changes will be visible for the rest of loaded users within some period of time (usually five minutes).

## Configuration File Reference

The topics in this section describe Rollbase configuration files. The source of these files should be in the `config` folder of the Master Server instance. When Rollbase starts, it saves these files, including the license, in the database so the configuration is applied across all instances. You can edit many settings through the UI. Those changes will be saved in the database and in the configuration files. If you change a setting by manually editing a configuration file, you need to restart the PAS or Tomcat instance that is hosting the Master Server.

## The components.xml File

This XML file has a `Component` node for each system component, that is, each component deployed with a WAR file. It is saved in the Rollbase database to allow you to add production servers without shutting down Rollbase. See [Adding Production Servers](#) on page 492 for more information. If you want to add other types of components, you need to edit the `components.xml` in the `config` directory of the Master Server and restart Rollbase to have the configuration take affect. If you change the default port 8080 or use a host name other than `localhost`, please make sure that `InternalRoot` values are properly adjusted for each component.

Since `InternalRoot` parameters are used for communications behind firewall Progress recommends the following to improve performance:

- Do not use the HTTPS protocol
- Use `localhost` or an explicit IP address instead of host name of your web server

Each `Component` node has the following elements:

XML Node/Attribute	Description
<code>name</code>	Unique name for this component used internally by the system.
<code>type</code> (attribute)	<p>Type of this component. Can be one of the following: MASTER, PROD, REST, RSS, ROUTER, SEARCH, STORAGE, WEBAPI, WORKFLOW</p> <p><b>Note:</b> If your license does not support multiple servers, you cannot have multiple nodes of the same type.</p>
<code>DisplayName</code>	Display name for this component.
<code>ExternalRoot</code>	<p>URL to the root of the web component as seen from outside of your firewall. <code>{!#HOST_NAME}</code> token will be automatically substituted by host name from your license file or <code>HostName</code> parameter specified in <code>shared.properties</code>. This URL is used for external access to components. Example:</p> <pre>&lt;ExternalRoot&gt;http://{!#HOST_NAME}/master/&lt;/ExternalRoot&gt;</pre> <p>For any plan other than a free evaluation, <code>ExternalRoot</code> must contain the <code>HostName</code> parameter as specified in your <code>license.xml</code>. You cannot deploy Rollbase on a different domain than it was licensed for.</p>
<code>InternalRoot</code>	<p>URL to the root of the web component as seen from inside your firewall (this can be the same as <code>ExternalRoot</code>). This URL is used for internal communication between components. Example:</p> <pre>&lt;InternalRoot&gt;http://localhost:8080/master/&lt;/InternalRoot&gt;</pre>
<code>Props</code> (optional)	Contains component-specific properties, see <a href="#">Component Specific Properties</a> on page 498.

## Component Specific Properties

This topic describes optional component-specific properties that you can specify in [The components.xml File](#) on page 497

Component	Property	Description	Default Value
MASTER	ResChecker	The amount of time (in hours) to wait between checking that customers have not exceeded their limits on resource usage (number of records, etc).	8
MASTER	ExpirationDays	Number of days for Free Trail to SaaS customers (if 0, the <b>Free Trial</b> box will not be displayed in the sidebar and trials will have no expiration date).	30
PROD N	Power	A decimal number between 0.5 and 2.0 indicating relative computational power of a Production Server. Servers with higher power will get a higher load.	1.0
PROD N	IsDedicated	If set to true, the Production server will only handle the load for selected customers.	false
RSS	MinRequestIntervalMins	If the interval between requests from RSS reader is shorter this value (in minutes), a cached version of the RSS feed will be sent.	15
ROUTER	StatusCheckInterval	Interval in minutes between checks of status for Production Servers.	10

Component	Property	Description	Default Value
ROUTER	CustWeight	The coefficient to be used in a formula to calculate load on a particular Production Server (only used in Multi-server Edition).	3
ROUTER	SynchIntervalMins	Interval in minutes to synchronize cached metadata for large customers loaded on multiple servers.	10
ROUTER	CheckLoginURL	If true, every 5 minutes send HTTP request to login page and check whether it is accessible.	true
SEARCH	IndexDir	Directory to store Lucene index files.	ROOT/search, where ROOT is shared Rollbase directory
SEARCH	CloseAfterMin	Remove metadata from cache if it is not being used for specified number of minutes.	60
SEARCH	LockTimeoutSec	Timeout, in seconds, to obtain a lock for index writing.	30
SEARCH	RamBufferMB	Size of RAM buffer, in MB, for indexing operations.	64
SEARCH	MaxSearchResults	Max number of search results to bring back for a single full text search.	200
STORAGE	StorateDir	Directory to store uploaded files, logs, templates, etc.	ROOT/storage, where ROOT is shared Rollbase directory
STORAGE	LogDir	Directory to store log files, if log files should be kept in separate location	Same as StorageDir

Component	Property	Description	Default Value
STORAGE	LogFormat	The log4j format for log messages.	[ %d ] %m%n
STORAGE	MaxBackupIndex	Determines how many backup files are kept before the oldest is erased.	1
STORAGE	CleanupDays	Delete log and backup files older than specified number of days (if specified value > 0).	0
STORAGE	CloseAfterMin	Close log4j logger after number of minutes of inactivity.	30
STORAGE	LogFileSize	Maximum size of particular log file, such as <code>main.log</code> .	300KB
STORAGE	LogLevel	A value of 1 means to log only important messages. A value of 2 means to log all users' activity.	2
STORAGE	MaxSystemBackups	Max number of system backup files to keep per customer. The system will delete older files if this number is exceeded.	7
WORKFLOW	EventCheckMins	Interval in minutes between checks to run matured delayed triggers.	5

#### **`databases.xml`**

This XML file requires a `Database` node for each database used by your Rollbase Private Cloud system. Each database node should contain the following child elements:

Note: The Evaluation Edition does not support multiple databases.

XML Node/Attribute	Description
name (attribute)	Symbolic name for database, used internally Default: RB
isDefault (attribute)	Marks database used by default for new Customers Default: None
isExternal (attribute)	Marks database which contains External data tables Default: None
MinConnections (attribute)	Number of database connections initially created in this pool Default: MinConnections from shared.properties
MaxConnections (attribute)	Max number of database connections in this pool Default: MaxConnections from shared.properties
MaxInUseConnTimeMins (attribute)	Max time (in minutes) allowed database connection to be in use, connection will be closed when time is up Default: MaxInUseConnTimeMins from shared.properties
MaxNotUsedConnTimeMins (attribute)	Max time (in minutes) allowed database connection in a pool to be idle, connection will be closed when time is up Default: MaxNotUsedConnTimeMins from shared.properties
MaxConnLifetimeMins (attribute)	Max connection lifetime before closure (in minutes) Default: MaxConnLifetimeMins from shared.properties
ConnTimeoutSec (attribute)	Timeout (in seconds) used when new database connection is created Default: None, uses database default
TxIsolation	Consult your database manual regarding Transaction Isolation level, if not sure about this setting - do not use it (database default)
useTxRecovery (attribute)	Enables or disables the default connection pooling's transaction recovery and retry feature Omit this attribute. When using Oracle or SQLServer databases and DataDirect drivers, Progress recommends that customers use the driver's retry feature instead.
Driver	Class name for JDBC driver used for this database Default: com.mysql.jdbc.Driver
Url	JDBC URL to database's service jdbc:mysql://localhost:3306/RB_DBO

XML Node/Attribute	Description
ConnectionRetryCount (attribute)	Number of times the driver retries connection attempts to the database server until a successful connection is established. Default: 0
ConnectionRetryDelay (attribute)	Number of seconds the driver waits between connection retry attempts when the ConnectionRetryCount attribute is set to a positive integer Default: 3
DbUser	Database user Default: account root
Password	Password for user account Default: None (must be specified prior to installation)

#### **events.xml**

This system file contains definitions for trigger types available in the system. Rollbase can provide more information to paying customers regarding how to develop and enable your own types of triggers for integration, etc. Progress recommends you do not modify this file unless you develop custom triggers.

Note: Custom Triggers developed by Private Cloud customers must be registered in `events.xml`. Please see Custom Development Kit for details.

#### **fieldgroups.xml**

This file contains definitions for object attributes, such as Location. Each attribute comes with a group of fields. Experienced Private Cloud administrators can add their own object attributes here.

#### **legacyobjects.xml**

This system file contains definitions for system tables, which can also be treated as object records for reporting purposes. Rollbase can provide more information to paying customers on how to integrate legacy database tables. Progress recommends that you do not modify this file.

#### **listitems.xml**

This file contains a list of shared Picklist Items (countries, states, etc.) to be added to each tenant during Customer creation. You can modify this file.

**license.xml**

Progress provides this file to paying customers. Save the license file in the `config` directory and restart your application server.

**Note:** Altering the contents of the license file will cause a system error.

**securitylevel.xml**

This file defines available security levels, see [Built-in Security Levels](#) on page 377. You can feel free to modify this file: change default levels or add more levels according to your security needs.

A `SecurityLevel` XML node with the attributes shown in the example below represents each level. No default values exist, so make sure to set a value for each property.

Example:

```
<SecurityLevel level="1" name="Low"
  inactiveSessionExpireMins="240" loginSessionExpireMins="480"
  maxFailedLogins="0" blockTimeMins="0" lockExpirationMins="120"
  minPasswordLength="6" nonLettersInPassword="false"
  caseSensitivePassword="false"
  sequentialCharsInPassword="true"
/>
```

XML Attribute	Description
Level	Numeric unique ID for this level
Name	Display name
inactiveSessionExpireMins	Expire user session after being idle for a number of minutes
loginSessionExpireMins	Expire user session after a number of minutes from login
maxFailedLogins	Temporary block user account after a number of unsuccessful login attempts
blockTimeMins	Block user account after unsuccessful login attempts for a number of minutes
lockExpirationMins	Expire record's lock after a number of minutes
minPasswordLength	Minimum number of characters in users' passwords
nonLettersInPassword	If set to "true" - users' passwords must include at least one non-alphabetical character

XML Attribute	Description
caseSensitivePassword	If set to "true" - users' passwords are case-sensitive (case-insensitive otherwise)
sequentialCharsInPassword	If set to "false" - sequential numbers of letters are not allowed in users' passwords.

#### **servicelevel.xml**

This file defines available Service Levels. Service level is assigned to each customer at creation time. You can modify this file: change default levels or add more levels according to your business needs.

Each Service Level is represented by XML node with the following attributes (no default values are used):

**Note:** Note: The following values defined in Service Level used only as default values and can be overridden per individual Customer: maxUsers, maxObjRecords, maxObjectDefs, maxFieldDefs, maxPortals, maxApplications.

XML Attribute	Description
Level	Numeric unique ID for this level
Name	Display name
maxUsers	Maximum number of User accounts which can be created
maxObjRecords	Maximum number of Object records which can be created
maxObjectDefs	Maximum number of Object definitions which can be created
maxFieldDefs	Maximum number of Field definitions which can be created
maxPortals	Maximum number of Portals which can be created
maxApplications	Maximum number of Applications which can be created
maxApiHitsDaily	Maximum number of API hits per 24 hours
maxStorageMB	Maximum size of locally stored files (in MB)

XML Attribute	Description
maxForeignLangs	Maximum number of foreign languages which can be assigned to customer (from 0 to 4)
maxDelayedEvents	Maximum number of delayed triggers for 24 hours

#### **shared.properties**

This file stores system-level properties to be shared among all Rollbase components. Each property is placed on a separate line in the following format:

```
key = value
```

The table below lists all available properties and their default values. A few important properties to note:

- The `HostName` property defaults to `localhost:8080` To use a different host name or port, adjust this property accordingly, or your server will not start.
- The `shared.properties` file uses ISO-8859-1 encoding. If you wish to include multi-byte characters in this file use Unicode \u notation such as: \u00A9 for a copyright symbol.

Property	Description
AdminEmail	Email address of the first Administrator user created during installation  Default: None (must be specified prior to installation)
AutoReplyAddress	Email address used as reply-to when no other address is provided  Default: same value as AdminEmail
Charts	URL to FusionCharts folder. This property must only be used if you are using flash-based FusionCharts.  <b>Note:</b> Rollbase 3.1 onwards, Rollbase supports HTML-based FusionCharts (See <a href="#">FusionCharts</a> on page 448).  Default: ../Charts/
CleanupMins	Interval in minutes between cleanup operations on each component  Default: 60
Copyright	Copyright string displayed on the bottom of each page.  Default: None

Property	Description
CustomValidationViaAPI	<p>When set to <code>true</code>, this property enables all validations required before executing an API operations. Therefore, any operations performed using the product UI and API will be validated.</p> <p>Default: <code>true</code></p>
DefaultDateFormat	<p>Index of date/time format used when no selection is made or available, use a type supported by Rollbase</p> <p>Default: US date/time format</p>
DefaultLangCode	<p>ISO language code used by default (when no language selection is made or available)</p> <p>Default: <code>en</code> (English)</p>
D2C_ConnPoolSize	<p>Specifies the number of connections to be cached and reused in order to avoid creating new connections to the datasources in DataDirect Cloud (D2C) frequently. The property takes values between two to ten.</p> <ul style="list-style-type: none"> <li>• Minimum value: 2</li> <li>• Maximum value: 10</li> </ul> <p>Default: 5</p>
D2C_ConnIdleTimeInMins	<p>Specifies the duration in minutes until which an existing DataDirect Cloud connection can stay idle. If a connection is found idle for the specified duration, a new connection is created and used. The property takes values between one to five.</p> <ul style="list-style-type: none"> <li>• Minimum value: 1</li> <li>• Maximum value: 5</li> </ul> <p>Default: 3</p>
DocumentationURL	<p>URL to Documentation Page: if not set, "Documentation" menu is not displayed</p> <p>Default: None</p>
EmergencyAddress	<p>Email address to receive emergency notifications when a system component is unavailable, number of threads or database connections exceeds a threshold, or another serious error occurs.</p> <p>Default: copied from <code>AdminEmail</code> property</p>
FontDirectory	<p>Directory where system fonts are stored, used by PDF Converter (see below)</p> <p>Default: None</p>

Property	Description
ForumURL	URL to Forum Page, if not set, "Forums" menu is not displayed Default: None
GoogleApplicationName	Google Application name to access Google Calendar and Docs (Spreadsheets), preferably have the format [company-id]-[app-name]-[app-version] Default: { !SystemName }- { !SystemName }-1
GettingStartedEnabled	When set to <code>false</code> , this property makes the <b>Getting Started</b> component unavailable in the page editor. Note that if any of your pages use the <b>Getting Started</b> component, you must manually remove the component and save the page using the page editor for the <b>Getting Started</b> component. Default: <code>true</code>
HostName	Host name of your Master Server Default: <code>localhost:8080</code>
HideHtmlEditorMenu	When set to <code>True</code> , this property hides the HTML editor menu bar. Default: <code>false</code>
HtmlEditorButtonRow1	Specifies the shortcut options of the first toolbar row (of the HTML editor) and its formatting. When this property is not set, this toolbar is disabled. Default: <code>bold,italic,underline,  ,forecolor,backcolor,  alignleft,aligncenter, alignright,alignjustify, , bullist,numlist, ,outdent, indent,blockquote,  ,undo,redo, ,link,unlink,  ,image, ,table,emoticons,  ,fontselect,fontsizeselect,  ,code,fullscreen,preview</code>
HtmlEditorButtonRow2	Specifies the shortcut options of the second toolbar row (of the HTML editor) and its formatting. When this property is not set, this toolbar is disabled. Default: <code>&lt;blank&gt;</code>

Property	Description
InactiveSessionExpireMins	Time of user's inactivity (in minutes) before user session expires Default: 240
LinkPrivacy	Link to "Privacy Statement" page on your website to be rendered at the bottom of each page Default: None (no link is rendered in this case)
LinkSecurity	Link to "Security Statement" page on your website to be rendered at the bottom of each page Default: None (no link is rendered in this case)
LinkTerms	Link to "Terms of Service" page on your website to be rendered at the bottom of each page Default: None (no link is rendered in this case)
LoginSessionExpireMins	Time from user's login (in minutes) before user session expires. Default: 480
MailHost	Host name of mail server used for outgoing emails Default: None, but requires a value
MailPassword	Password for Mail User account Default: None
MailPort	Port used to access mail server Default: 25
MailUser	User name on mail server used for outgoing emails Default: Copied from AdminEmail, which is required
MailUseSSL	If true, use SSL to access mail server Default: false
MaxAjaxPerSession	Max number of AJAX API calls per user login session 1000
MaxAlarmConnections	The system will send a message to admin if number of database connections exceeds this value Default: 90

Property	Description
MaxAlarmThreads	<p>The system will send a message to admin if number of Java threads exceeds this value. Default: 200</p>
MaxAttachmentSizeKB	<p>Max size of email attachment in KB Default: 2048</p>
MaxBackupsPerMonth	<p>Max number of times a backup activity can be performed by a customer tenant per month. Default: 10</p>
MaxCachedEntities	<p>Max number of object records cached by servers. If set to 0, no caching is enabled. Default: 1000</p>
MaxChartCols	<p>Maximum number of columns displayed in charts Default: 1000</p>
MaxConnections	<p>Max number of database connections in each connection pool. Default: 100</p>
MaxConnLifetimeMins	<p>Max connection lifetime before closure (in minutes) Default: 60</p>
MaxDbLongStrLength	<p>Max size (in chars) of long text fields (CLOBs) to be stored in the database Default: 80000</p>
MaxDelayedTriggers	<p>Max number of triggers to be executed for delayed updates Default: 20</p>
MaxEmailAttachments	<p>Max number of email attachments. Default: 3</p>
MaxEmailBodyKB	<p>Max size of email message body in KB Default: 2048</p>

Property	Description
MaxEmailQueueSize	<p>The maximum number of emails to enqueue and send asynchronously. When the emails in the queue reach MaxEmailQueueSize, new emails will be sent synchronously and will not be enqueued. The mail jobs in queue will continue to be sent by another thread. As the queue drains, and space becomes free, jobs can then be enqueued once again. A value of 0 causes all emails to be sent synchronously.</p> <p>Default: 100</p>
MaxExportRows	<p>Max number of report rows to be exported as spreadsheet and max number of items in drop-down lookup. Important: for Oracle database this value cannot exceed 1000</p> <p>Default: 1000</p>
MaxFileSizeKB	<p>Max size in KB for upload files.</p> <p>Default: 5120</p>
MaxFilters	<p>Max number of filters for Views.</p> <p>Default: 5</p>
MaxFormulaSize	<p>Max size (in characters) of parsed formula. This size is checked before sending formula to JavaScript Engine.</p> <p>Default: 10240</p>
MaxHTTPParamLength	<p>Max size (in chars) of HTTP parameters processed by the system.</p> <p>Default: 80000</p>
MaxInUseConnTimeMins	<p>Max time (in minutes) allowed database connection to be in use. Connection will be forced to close after that.</p> <p>Default: 30</p>
MaxJSTimeMs	<p>Max elapsed time for server-side JavaScript formula engine per formula in ms (formula will be aborted afterwards).</p> <p>Default: 3000</p>
MaxListTotals	<p>Max number of columns with totals in Views and Reports</p> <p>Default: 3</p>
MaxNotUsedConnTimeMins	<p>Max time (in minutes) allowed database connection in a pool to be idle. Connection will be closed after that.</p> <p>Default: 30</p>

Property	Description
MaxQueryLength	Maximum number of characters in query sent to Query API Default: 2000
MaxRecurrEvents	Maximum number of recurrent calendar events Default: 300
MaxReqsInQuery	Maximum number of records which can be selected in View or fetched by SQL query Default: 20,000
MaxRuntimeTriggers	Max number of triggers to be executed for non-delayed updates. Default: 100
MaxSearches	Max number of conditions to be used in Detailed Search component. Default: 20
MaxTimeToRunTrigMS	Max time to run a group of triggers ON_AFTER_UPDATE etc. Default: 30000 (30 seconds)
MaxTransactionTime	Max time in minutes allowing database transaction to run (will be rolled back after that) Default: 10
MinConnections	Number of database connections initially created in each connection pool. Default: 1
MinRecursType	Minimum allowed recursion interval for triggers and Batch Jobs: <ul style="list-style-type: none"><li>• 1: day (24 hours)</li><li>• 4: hour</li><li>• 5: minute 1</li></ul> Default: 1
OnlineHelpBaseUrl	Specifies the online address of Progress Rollbase product documentation. Default: <a href="http://documentation.progress.com/output/Rollbase/index.html#context/rb/">http://documentation.progress.com/output/Rollbase/index.html#context/rb/</a>
PacificUIDisabled	Controls whether the UI has the "Pacific" look and feel or the "classic" look and feel. Defaults to <code>false</code> for new installations. A value of <code>true</code> maintains the "classic" look and feel.

Property	Description
PurgeAfterDays	<p>Purge deleted records from the Recycle Bin after specified number of days</p> <p>Default: 30</p>
RBMobile_CORS.Allow_origin.HOST	<p>Specifies a regular expression that directs Rollbase Mobile to accept requests from the specified source. This is as per CORS mechanism.</p> <p>Default: %RBMobileCORSOriginRegx%</p>
ShowDebugInfo	<p>Set this flag to true if you use server for development work only. This will eliminate waiting time when deleting applications and objects.</p> <p>Default: false</p>
ShowHelpTips	<p>When set to false, this property hides all the help tips and the <b>Help Me</b> button in the footer of your Rollbase instance.</p> <p>Default: true</p>
SkipEmails	<p>Do not send email messages but dump them on standard output instead. Use this option for development version when no outbound email is available.</p> <p>Default: false</p>
SQLKeywords	<p>It is a comma-separated SQL keywords of your Database. Progress recommends that you verify and add any reserved keywords in this property before generating SQL queries for External objects.</p> <p>Default: All the terms identified by SQL as a keyword.</p>
SQLSpecialChars	<p>It is a comma-separated SQL special characters of your Database. Progress recommends that you verify and add any special characters in this property before generating SQL queries for External objects.</p> <p>Default: All the terms identified by SQL as a Special Character.</p>
StatusCheck	<p>Interval in minutes between status checks on each component.</p> <p>Default: 5</p>
storageUsageLimitForBkp	<p>Storage usage limit (in MB) of a full backup. For information on using this property, see <a href="#">Backup and Restore</a> on page 426.</p> <p>Default: 20</p>

Property	Description
SubscribeNowURL	<p>URL to be invoked when "Subscribe Now" button is clicked (this button is displayed on sidebar for trial customers)</p> <p>Default: None ("Subscribe Now" will redirect to Support Portal if this setting is not filled)</p>
SupportTicketURL	<p>Specifies the URL of the <b>Support Tickets</b> link in the footer of your Rollbase instance. Comment-out the property to hide the <b>Support Tickets</b> link. When left blank, the <b>Support Ticket</b> link navigates to a Progress Rollbase Support page in which users can raise a support ticket for the master tenant to acknowledge.</p> <p>Default: &lt;blank&gt;</p>
SupportURL	<p>Specifies the URL of the <b>Support</b> link in the footer of your Rollbase instance. Typically, you direct your users to a Web page where they can get product support.</p> <p>Default: <a href="http://www.progress.com/support-and-services">http://www.progress.com/support-and-services</a></p>
SystemName	<p>Displayed in upper-left corner of each page as a link to your web site.</p> <p>Default: Rollbase</p>
TimeZones	<p>Comma-separated list of Time Zones. If required, add to this list any TimeZone IDs which are a part of <code>TimeZone.getAvailableIDs()</code>.</p>
TitleTemplate	<p>Template for page's title. Can be overridden by ISV partners. See <a href="#">Creating and Managing Customer Tenants</a> on page 555 for more information.</p> <p>Default: <code>{!A}   {!S}   {!P}</code></p>
UnloadCustAfterMins	<p>Unload customer from cache after period of time (in minutes)</p> <p>Default: 60</p>
UseDocxHelp	<p>Use links to DOCX files from "Rollbase in Action"</p> <p>Default: <code>true</code></p>
UseZipSearch	<p>Enable search by distance from ZIP/Postal code. Enabling this setting requires that <code>RB_ZIP_CODES</code> table be populated with actual data - not supplied with Private Cloud</p> <p>Default: <code>false</code></p>

Property	Description
WebSiteHTTP	HTTP URL to root of your web site Default: <code>http://HostName</code>
WebSiteHTTPS	HTTPS URL to root of your web site Default: <code>https://HostName</code>

## PAS Command Line Reference

This section describes PAS tcman utility commands. For more information about PAS, including its security model, see <http://documentation.progress.com/output/ua/PAS/>.

### The tcman command

#### Purpose

TCMAN is a command-line utility for managing and administering the Pacific Application Server. On UNIX systems, you run the `tcman.sh` script followed by appropriate TCMAN actions and options. On Windows systems, you run the `tcman.bat` batch file, which is identical syntactically and functionally with `tcman.sh`.

---

**Note:** For the sake of brevity, all the syntax statements and examples in this reference show the `tcman.sh` script.

---

#### Syntax

```
{ $CATALINA_HOME | $CATALINA_BASE } /bin/tcman.sh action [ general_options ]
[ action_options ]
```

#### Parameters

`$CATALINA_HOME` | `$CATALINA_BASE`

Specify whether to run TCMAN from the root directory of the installed Pacific Application Server (`$CATALINA_HOME`) or from the root directory of an instance (`$CATALINA_BASE`). The context of where you run TCMAN (whether from the `/bin` directory of the parent, or the `/bin` directory of an instance) affects which server the utility acts on.

---

**Note:** TCMAN automatically determines the value of CATALINA\_BASE from the directory where you start it. When you run it from the /bin directory of an instance, the value of CATALINA\_BASE is the root directory of the instance. If you run it from the /bin directory of the installed Pacific Application Server, the value of CATALINA\_BASE is the root directory of the installed server (which is the same value as CATALINA\_HOME).

---

*action*

Specify which TCMAN action to invoke.

*general\_options*

Specify one or more of the TCMAN common options that can apply to most actions. Note that one or more of the general options may be required by a specific action. For example, the `list` action requires `-u` in order to pass a user name and password.

The output of `tcman.sh help action` includes a list of general options that are applicable to a particular action.

The following table is a list of the common options:

**Table 5: TCMAN general options**

Common options	Function
<code>-u user_name:password</code>	Pass a valid user name and a password for HTTP Basic access authentication.
<code>-v</code>	Display verbose output.
<code>-M URL</code>	Override the default manager that manages Web applications by specifying the URL of an alternative manager. <i>URL</i> is expressed in the following format: <code>{http https}://host:port/manager_application</code>
<code>-B</code>	Override default CATALINA_BASE environment settings.
<code>-n</code>	Debug the TCMAN action but do not execute changes.
<code>-I instance_name</code>	Run TCMAN from the /bin directory of the specified instance.

*action\_options*

Specify an option that applies to the selected *action*. These options are explained in the topics that describe each action.

## Example

Run the `help` action from the core server (`/psc/pashome`) to display a list of available TCMAN actions:

```
/psc/pashome/bin/tcman.sh help
usage: tcman action [options...]
manager actions:
  list      list deployed applications
  info      list server info
  deploy    deploy application
  undeploy  undeploy application
  reload    reload application
  status    show server status
  leaks     show server memory leaks
  enable    start web application running
  disable   stop running web application
  resources list server global resources
  sessions  list a web application's sessions
server actions:
  create   create a new server instance
  delete   delete server instance
  config   dump CATALINA_BASE configuration
  clean    clean/archive log files
  instances list tracked server instances
  register manually register an instance
  unregister manually unregister an instance
  start    start this server
  stop     stop this server
  version  show the server version information
  test     test the server's configuration
general actions:
  env      show tcman execution environment
  help     show this information
```

## Manager actions

This section details the actions available for deploying, running, and monitoring web applications on a server instance.

### List deployed applications (list)

#### Purpose

Display all the web applications that are deployed on an instance.

**Note:** This command may be used whether the instance is online or offline. However, the output differs. When used offline, TCMAN simply shows a list of deployed application directories in the instance's web applications directory. When used online, it provides additional run-time details about the deployed web applications.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance if the instance is online. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

## Syntax

```
tcman.sh list [general_options] [-u user_id:password]
```

## Parameters

*general\_options*

Specify one or more of the options that can be used with any TCMAN action.

-u *user\_id:password*

Specify a valid user name and password for HTTP Basic access authentication. (The default is -u tomcat:tomcat.)

**Note:** This option is required if the server is online. It is not required if the server is offline.

## Example

Show the Web applications deployed to acme1 when the instance is online:

```
/psc/acme1/bin/tcman.sh list -u tomcat:tomcat
OK - Listed applications for virtual host localhost
/:running:0:ROOT
/manager:running:4:manager
/oemanager:running:0:oemanager
/oadapters:running:0:oeabl
```

Show the Web applications deployed to acme1 when the instance is offline:

```
/psc/acme1/bin/tcman.sh list
OK - Listing directories for /psc/acme1/webapps
/manager:stopped:0:manager
/oadapters:stopped:0:oeabl
/oemanager:stopped:0:oemanager
/:stopped:0:ROOT
```

## Display OS and server information (info)

### Purpose

Display server and OS information for a running instance.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

Use the `test` action to show configuration information about a server that is not running.

## Syntax

```
tcman.sh info [general_options] -u user_name:password
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help info` to see which general options are appropriate.

*-u user\_name:password*

Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

## Example

Display the OS and server information for the running instance named `acme1`:

```
$: /psc/pashome/tcman.sh info -I acme1 -u tomcat:tomcat
OK - Server info
Tomcat Version: Apache Tomcat/7.0.42
OS Name: Linux
OS Version: 2.6.18-164.el5
OS Architecture: amd64
JVM Version: 1.7.0_02-b13
JVM Vendor: Oracle Corporation
```

## Deploy a Web application (deploy)

### Purpose

Deploy a Web application (.war file) to a Pacific Application Server instance whether the server is running (online) or is not running (offline). TCMAN copies the web application to the server's web application directory. If the server is online, you must stop and restart it in order to complete the deployment.

## Syntax

```
tcman.sh deploy [general_options] [-u user_id:password] [-a app_name]
war_file_path
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help deploy` to see which general options are appropriate.

`-u user_id:password`

Specify a valid user name and password for HTTP Basic access authentication.

---

**Note:** This option is required if the server is online. It is not required if the server is offline.

---

`-a app_name`

Specify a name for the web application. If you do not use this option, the application name will be the same as the `.war` file name.

`war_file_path`

Specify the location of the web application `.war` file that you want to deploy.

## Example

Deploy and rename `oeabl.war` (a web application that implements OpenEdge adapters) to the `acme1` instance of the core `pashtome` server:

```
/psc/acme1/bin/tcman.sh deploy -a oeadapters /psc/pashome/extras/oeabl.war
OK - deployed /psc/pashome/extras/oeabl.war to local directory
/psc/acme1/webapps
```

---

**Note:** The `$CATALINA_HOME/extras` directory (`/psc/pashome/extras` in the example above) also contains number of instance management applications, including `host-manager.war`, `manager.war`, and `oemanager.war`.

---

## Undeploy a Web application (undeploy)

### Purpose

Remove a Web application from running (online) or stopped (offline) instances. If the instance's autodeploy option is off, you must stop and restart a running server to complete removal. Note that the autodeploy option is set in the `.../conf/appserver.properties` file and is off by default.

### Syntax

```
tcman.sh undeploy [general_options] [-u user_id:password] app_name
```

### Parameters

`general_options`

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help undeploy` to see which general options are appropriate.

*-u user\_id:password*

Specify a valid user name and password for HTTP Basic access authentication. (The default is *-u tomcat:tomcat*.) This option is required if you are accessing an online instance.

*app\_name*

Specify the name of the web application to remove.

## Example

Remove the `oemanager` application from the `acme1` instance:

```
/psc/acme1/bin/tcman.sh undeploy -u tomcat:tomcat oemanager
OK - Undeployed application at context path /oemanager
```

## Reload a Web application (reload)

### Purpose

Restart a deployed, running Web application so that the application can pick up changes to its classes or libraries.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

---

**Note:** The reload action does not reload the web application's `web.xml` file. To begin using changes to `web.xml`, you must stop and restart the web application.

---

### Syntax

```
tcman.sh reload [general_options] -u user_id:password app_name
```

### Parameters

*general\_options*

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help reload` to see which general options are appropriate.

*-u user\_id:password*

Specify a valid user name and password for HTTP Basic access authentication. (The default is *-u tomcat:tomcat*.)

---

**Note:** This option is required if the server is online. It is not required if the server is offline.

---

*app\_name*

Specify the name of the web application to restart.

## Example

Reload the `oemanager` web application running on the `acme1` instance:

```
/psc/acme1/bin tcman.sh reload -u tomcat:tomcat oemanager
OK - Reloaded application at context path /oemanager
```

## Display detailed server status (status)

### Purpose

List information from the core server's memory, including web application statistics. Information includes memory pool usage, connector thread status, and connector status. Output is in XML format. (Note that redirecting the output to an XML viewer makes it more readable.)

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh status [general_options] -u user_name:password [-f]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help status` to see which general options are appropriate.

`-u user_name:password`

Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

`-f`

Return full status information.

## Example

Display core server's memory and web application statistics and use `xmllint` to format for readability:

```
$: tcman.sh status -u tomcat:tomcat | xmllint --format -
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="/manager/xform.xsl" ?>
<status>
  <jvm>
    <memory free="453196832" total="520028160" max="1051394048"/>
    <memoriypool name="PS Eden Space" type="Heap memory" usageInit="50331648" usageCommitted="48758784" usageMax="55967744" usageUsed="1525560"/>
    <memoriypool name="PS Old Gen" type="Heap memory" usageInit="469762048" usageCommitted="469762048" usageMax="1006632960" usageUsed="63861584"/>
    <memoriypool name="PS Survivor Space" type="Heap memory" usageInit="8388608" usageCommitted="1507328" usageMax="1507328" usageUsed="1444184"/>
    <memoriypool name="Code Cache" type="Non-heap memory" usageInit="2555904" usageCommitted="3407872" usageMax="50331648" usageUsed="3303104"/>
    <memoriypool name="PS Perm Gen" type="Non-heap memory" usageInit="67108864" usageCommitted="67108864" usageMax="134217728" usageUsed="47406400"/>
  </jvm>
  <connector name=""http-bio-8601"">
    <threadInfo maxThreads="150" currentThreadCount="0" currentThreadsBusy="0"/>
    <requestInfo maxTime="0" processingTime="0" requestCount="0" errorCount="0" bytesReceived="0" bytesSent="0"/>
    <workers/>
  </connector>
  <connector name=""http-bio-8501"">
    <threadInfo maxThreads="300" currentThreadCount="10" currentThreadsBusy="1"/>
    <requestInfo maxTime="2008" processingTime="2116" requestCount="10" errorCount="0" bytesReceived="0" bytesSent="5838"/>
    <workers>
      <worker stage="S" requestProcessingTime="2" requestBytesSent="0" requestBytesReceived="0" remoteAddr="127.0.0.1" virtualHost="localhost" method="GET" currentUri="/manager/status" currentQueryString="XML=true" protocol="HTTP/1.1"/>
    </workers>
  </connector>
</status>
```

## Display memory leaks (leaks)

### Purpose

List Web applications with potential memory leaks.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh leaks [general_options] -u user_name:password
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help leaks` to see which general options are appropriate.

`-u user_id:password`

Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

## Example

Display memory leaks for web applications deployed on the `acme1` server instance:

```
/psc/acme1/bin/tcman.sh leaks -u tomcat:tomcat
OK - Found potential memory leaks in the following applications:
/warehouse
```

## Start a Web application (enable)

### Purpose

Start a web application that is deployed but not running.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh enable [general_options] -u user_id:password app_name
```

## Parameters

*general\_options*

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help start` to see which general options are appropriate.

`-u user_id:password`

Specify a valid user name and password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

*app\_name*

Specify the name of the web application to start.

---

**Note:** To start the ROOT web application, you can specify `/` or `ROOT`.

---

## Example

Start the oeabl application deployed on the acme1 instance:

```
tcman.sh enable -u tomcat:tomcat oeabl
OK - Started application at context path /oeabl
```

## Stop a Web application (disable)

### Purpose

Stop a running Web application.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh disable [general_options] [-u user_id:password] app_name
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help disable` to see which general options are appropriate.

*-u user\_id:password*

Specify a valid user name and password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

*app\_name*

Specify the name of the web application to disable.

---

**Note:** To disable the ROOT web application, you can specify `/` or `ROOT`.

---

### Example title

Disable the oeabl application running on the acme1 instance:

```
/psc/acme1/bin/tcman.sh disable -u tomcat:tomcat oeabl
OK - Stopped application at context path /oeabl
```

## Display global server resources (resources)

### Purpose

List the global resources used by the core server.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh resources [general_options] -u user_name:password
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help resources` to see which general options are appropriate.

`-u user_name:password`

Pass a valid user name and a password for HTTP Basic access authentication.

(The default is `-u tomcat:tomcat`.)

### Example

Display global resources for the running instance, `acme1`:

```
$: /psc/acme1/bin/tcman.sh resources -u tomcat:tomcat
OK - Listed global resources of all types
ServiceRegistry/ServiceRegistryFactory:com.progress.appserv.services.naming.ServiceRegistry
UserDatabase:org.apache.catalina.users.MemoryUserDatabase
```

## Display Web application HTTP sessions (sessions)

### Purpose

Display how many sessions are active for the specified Web application, categorized by their duration.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

### Syntax

```
tcman.sh sessions [general_options] -u user_id:password app_name
```

## Parameters

*general\_options*

Specify one or more of the options that can be used with any TCMAN action.

*-u user\_id:password*

Specify a valid user name and password for HTTP Basic access authentication. (The default is *-u tomcat:tomcat*.)

*app\_name*

Specify the name of the web application to analyze for session information.

## Example

Show the active sessions for the `manager` application deployed on the `acme1` instance:

```
/psc/acme1/bin/tcman.sh sessions -u tomcat:tomcat manager
OK - Session information for application at context path /manager
Default maximum session inactive interval 30 minutes
<1 minutes: 1 sessions
8 - <9 minutes: 2 sessions
9 - <10 minutes: 1 sessions
```

## Server actions

This section details the actions available for creating and monitoring server instances.

## Create an instance (create)

### Purpose

Create a new instance of the core Pacific Application Server server by running this action from `/bin` directory of the core server (`$CATALINA_HOME/bin/tcman.sh create`).

### Syntax

```
tcman.sh create [general_options] [-f] [-p port_num] [-P port_num]
[-s port_num] [-j port_num] [-W pathname] [-N instance_name]
[-U user_id] [-G group_id] base_path
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help create` to see which general options are appropriate.

**-f**

Copy all deployed web application archives (.war files) from \$CATALINA\_HOME to the new instance.

**-p port\_num**

Specify the TCP port that listens for HTTP messages. The default is 8080.

**-P port\_num**

Specify the TCP port that listens for HTTPS messages. The default is 8443.

**-s port\_num**

Specify the TCP port to use to stop an instance. On Windows systems, you must specify a shutdown port. On UNIX, shutdown ports are optional.

**-j port\_num**

Specify the TCP port that listens for AJP13 messages (an Apache protocol for handling requests from a web server to an application server). The default is 8009.

**-W pathname**

Specify the directory where web applications will be deployed. The default is \$CATALINA\_BASE/webapps.

**-N instance\_name**

Specify a name for the instance. The default is the name of the directory where the instance is created.

All instances are automatically registered for tracking when they are created. If you intend to track an instance, the instance name cannot contain spaces or any of the following characters: "[ . # | ] \$ ? + = { / , }"

**-U user\_id**

Specify the user-id of the owner of all the files and directories of the instance. The default is the user-id of the current process. **-G** must be specified if you use this option.

**-G group\_id**

Specify the group-id of the owner of all the files and directories of the instance. The default is the group-id of the current process. **-U** must be specified if you use this option.

**base\_path**

Specify the pathname where you will create the instance.

## Example

Create an instance of /psc/pashome in /psc/acme1:

```
$: /psc/pashome/bin/tcman.sh create -p 8501 -P 8601 -s 8701 /psc/acme1
Server instance acme1 created at /psc/acme1
```

## Delete an instance (delete)

### Purpose

Remove the directory tree and all of the files in an instance. Alias tracking is disabled for servers that are removed.

To execute this action, the instance cannot be running.

---

**Note:** You cannot recover any files or directories removed by the `delete` action. Backup anything you want to save before launching this action.

Also note that you cannot use `delete` to remove the installed, root server ( `$CATALINA_HOME` ).

---

### Syntax

```
tcman.sh delete [general_options] [-y] [base_path | alias_name]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help delete` to see which general options are appropriate.

`-y`

Delete everything without prompting for confirmation.

*base\_path*

Specify the pathname of the instance that you intend to delete.

*alias\_name*

Refer to the instance that you intend to delete by its alias rather than its pathname.

## Example

Delete the instance of `pashome` that was created in `/psc/acme3`:

```
$: /psc/pashome/bin/tcman.sh delete /psc/acme3
The following directory tree will be removed permanently:
  ( WARNING all deployed web applications will be DELETED!! )
/PAS/wrkdir/acme3
/PAS/wrkdir/acme3/conf
/PAS/wrkdir/acme3/temp
/PAS/wrkdir/acme3/common
/PAS/wrkdir/acme3/common/lib
/PAS/wrkdir/acme3/logs
/PAS/wrkdir/acme3/webapps
/PAS/wrkdir/acme3/webapps/ROOT
/PAS/wrkdir/acme3/webapps/ROOT/static
/PAS/wrkdir/acme3/webapps/ROOT/static/error
/PAS/wrkdir/acme3/webapps/ROOT/static/auth
/PAS/wrkdir/acme3/webapps/ROOT/META-INF
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters/rest/PingService
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters/soap
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com/progress
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com/progress/appserv
/PAS/wrkdir/acme3/work
/PAS/wrkdir/acme3/bin
Type 'yes' to continue
yes
Delete operation complete
server removed at /PAS/wrkdir/acme3
```

## Display and manage an instance's configuration (config)

### Purpose

View, add, update, or delete the property values specified in `..../conf/appserver.properties`.

When you run `tcman.sh config` with no parameters, it displays the core Tomcat server's configuration, and all the properties in both `..../conf/appserver.properties` and `..../conf/jvm.properties`. Note, however, that you can only view `jvm.properties`. You cannot modify its contents with the `config` action.

### Syntax

```
tcman.sh config [general_options]
[prop_name | prop_name=value | +prop_name=value | ~prop_name]
```

### Parameters

`general_options`

Specify one or more of the general TCMAN options. Run `tcman.sh help config` to see which general options are appropriate.

*prop\_name*

Display the specified property and its value.

*prop\_name=value*Set the value of a property that exists in *.../conf/appserver.properties*.*+prop\_name=value*Add a new property to *.../conf/appserver.properties* and set its value.*~prop\_name*Remove the specified property from *.../conf/appserver.properties*.

## Examples

Show the configuration and properties of `acme1`, an instance of the core server, `pashome`:

```
$: /psc/acme1/bin/tcman.sh config
Using CATALINA_BASE:      /psc/acme1
Using CATALINA_HOME:      /psc/pashome
Using CATALINA_TMPDIR:    /psc/acme1/temp
Using JRE_HOME:           /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID:       /psc/acme1/temp/catalina.pid
Server version: Apache Tomcat/7.0.42
Server built:  Jul 2 2013 08:57:41
Server number: 7.0.42.0
OS Name:      Linux
OS Version:   2.6.18-164.el5
Architecture: amd64
JVM Version:  1.7.0_02-b13
...
...
```

Display the value of a single property:

```
$: /psc/acme1/bin/tcman.sh config psc.as.http.port
psc.as.http.port=8501
```

Update the value of a property that exists in the `appserver.properties` file and then check the value:

```
$: /psc/acme1/bin/tcman.sh config psc.as.http.port=6543
$: tcman.sh config psc.as.http.port
psc.as.http.port=6543
```

Add a new property/value pair to the `appserver.properties` file and then check the value:

```
$: /psc/acme1/bin/tcman.sh config +my.home.dir=/home/jarhead
$: tcman.sh config my.home.dir
my.home.dir=/home/jarhead
```

Remove a property/value pair from the `appserver.properties` file and check if deletion was successful:

```
$: /psc/acme1/bin/tcman.sh config ~my.home.dir
$: tcman.sh config my.home.dir
Property does not exist - my.home.dir
```

---

**Caution:** There are no restrictions to property removal. You can render the server unable to start if you remove a property required by `conf/server.xml`.

---

## Notes

- All property names are case sensitive.
- You cannot enter multiple property names (`prop_name`) on the command line to view, update, or add properties to the `appserver.properties` file.
- You cannot use the config action to update existing values or add new values to the `jvm.properties` file

# Display or modify the server features of an instance (feature)

## Purpose

View, enable, or disable the server features contained in the `/conf/server.xml` file of an instance.

When you run `tcman.sh feature` with no parameters, it displays a list of the features (and their current status) that you can enable or disable. You can also display the status of a single server feature. After viewing the status of a feature, you can use `tcman.sh feature` to change its setting.

## Syntax

```
tcman.sh feature [general_options] [feature_name [= {on | off}]]
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help feature` to see which general options are appropriate.

*feature\_name*

Specify one of the features defined in an instance's `conf/server.xml` file. Running `tcman.sh feature` without *feature\_name* displays a list of all the features.

on

Enables the named feature.

off

Disables the named feature.

## Example

Display the list of server feature settings for `acme1`, enable AJP13 (Apache JServ Protocol version 1.3), and verify that the feature is enabled:

```
$: /psc/acme1/bin/tcman.sh feature
SecurityListener=off
JMXLifecycle=off
PSCRegistry=on
HTTP=onHTTPS=on
AJP13=off
Cluster=off
UserDatabase=on
JAASRealm=off
LDAPRealm=off
PASInstrument=off
RemoteHostValve=on
RemoteAddrValve=onSingleSignOn=on
AccessLog=on
CrawlerSessionManager=on
StuckSessionValve=on

$: /psc/acme1/bin/tcman.sh feature AJP13=on

$: /psc/acme1/bin/tcman.sh feature AJP13
AJP13=on
```

## Notes

- Server features for instances are set in `$CATALINA_BASE/conf/server.xml`. You can change feature status by manually editing this file. However, it is safer to use `tcman.sh feature` to avoid corrupting the file with erroneous entries.
- Run `tcman.sh feature` when the instance is offline.

## Clean up or archive server log files (clean)

### Purpose

Truncate, move, or delete the log files located in the `/logs` directory of the core server or instance. If the server is running, clean truncates log files to zero length. If the server is not running, clean deletes the log files from the file system.

You have the option to save log files to a subdirectory of `/logs`.

## Syntax

```
tcman.sh clean [general_options] [-A]
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help clean` to see which general options are appropriate.

`-A`

Archive log files to a subdirectory of `$CATALINA_BASE/logs`. The directory is automatically named with a month-day-year-second (`MM-DD-YYYY-ss`) time-stamp format. If the server is not running, the files in `$CATALINA_BASE/logs` are deleted.

## Example

Archive the log files of `acme1`, an instance of the core server `pashome`, and save to a file:

```
/psc/pashome/tcman.sh clean -I acme1 -A
```

## Display server instances (instances)

### Purpose

Show the names and locations of the instances created from the Pacific Application Server installed in `$CATALINA_HOME` by displaying the contents of the file where instances are registered for tracking.

By default, instances are registered for tracking

`$CATALINA_HOME/conf/instances.{windows|.unix}`. The file name extension indicates the OS platform where the PAS server is installed.

## Syntax

```
tcman.sh instances [general_options]
```

## Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help instances` to see which general options are appropriate.

## Output format

The following is the format of the output from a TCMAN `instances` action:

```
alias-name | full-file-path | type | state
```

*alias-name*

The user-defined name for the instance.

*full-file-path*

The location, in the OS file system, of the instance's root directory.

*type*

The designation of the server instance type (for example: `instance`, `service`, ...).

*state*

An indication of the instance's validity. `OK` is returned for a valid server and `invalid` is returned for a corrupted or non-existent server.

## Example

Display the instances of the core server installed in `/psc/pashome`:

```
/psc/pashome/bin/tcman.sh instances
acme1 | /psc/wrk/acme1 | instance | ok
acme2 | /psc/wrk/acme2 | instance | ok
```

## Notes

- By default, instances are registered when you execute a `$CATALINA_HOME/bin/tcman{.sh|.bat}` `create` action, which automatically adds instance entries to an `instances` file. TCMAN removes instance entries from the file when you execute a `delete` action.

You can manually add or remove instance entries from `instances` by using the `register` or `unregister` actions.

- By default, the name and location of the file where instances are registered is `$CATALINA_HOME/conf/instances.{windows|.unix}`.

You can change the location of the instance registration file by adding and setting the `psc.as.instdir` property in the `appserver.properties` file. Use the TCMAN config action as in the following example:

```
tcman.sh config '+psc.as.instdir=PATH'
```

where `PATH` is a path name or an environment variable.

You can also change the location and/or name of instance registration files by setting the environment variables, `PAS_AS_INSTANCE_DIR`, and `PAS_AS_INSTANCE_FILE`.

# Register an instance for tracking (register)

## Purpose

Register an instance for tracking purposes.

---

### Note:

Instances are automatically registered for tracking when you execute a `create` action. You use the `register` action to restart tracking on instances after tracking was stopped.

A typical use for unregistering and then re-registering an instance is to make configuration changes when moving instances from one location (core server) to another. The `register` action enables tracking and also updates the value of `CATALINA_HOME` in all of the executable scripts in the instance's `/bin` directory to refer to the new core server.

---

## Syntax

```
tcman.sh register alias_name instance_path
```

## Parameters

*alias\_name*

Specify a meaningful name for the instance. The alias name must be unique in the instances file.

*instance\_path*

Specify the OS file system path to where the instance exists. This value will be expanded into a fully qualified OS directory path and will be verified to exist.

## Example

Track `test1`, which is an alias for the instance `/psc/acme1`:

```
/psc/pashome/bin/tcman.sh register test1 /psc/acme1
```

## Notes

When you register an instance for tracking or create a new instance with the `create` command, an entry is created in the core Pacific Application Server's `$CATALINA_HOME/conf/instances.[unix|windows]` file.

The `instances.[unix|windows]` file is a simple text file, which can be manually edited (with care) in the event that it becomes out of date. The format for entries is:

```
instance_name = base_path
```

An `instances.unix` file uses Unix OS file path syntax (forward slashes), and an `instances.windows` file uses Windows OS file path syntax (backslashes) to specify `base_path`.

Also note that in an `instances` file:

- Any line starting with a pound-sign (#) is a comment line.
- Blank lines are skipped.
- Instance names cannot contain spaces or any of the following characters: "[ . # | ] \$ ? + = { / , }"

## Stop tracking an instance (unregister)

### Purpose

Stop tracking an instance by removing the instance's entry from the `$CATALINA_HOME/conf/instances.[unix|windows]` file.

---

### Note:

You use the `register` action to restart tracking on instances after tracking was stopped with `unregister`.

A typical use for unregistering and then re-registering an instance, is to make configuration changes when moving instances from one location, or core server, to another. The `register` action not only enables tracking, it also updates the value of `CATALINA_HOME` in all of the executable scripts in the instance's `/bin` directory to refer to the new core server.

---

### Syntax

```
tcman.sh unregister alias_name
```

### Parameters

`alias_name`

Specify the alias name of the instance that you want to stop tracking. The alias name must exist in an `instances.[unix|windows]` file.

### Example

Stop tracking `test1`, which is an instance of `/psc/pashome`:

```
/psc/pashome/bin/tcman.sh unregister test1
```

## Start an instance (start)

### Purpose

Start an instance of a Pacific Application Server, optionally in debug mode.

---

## Syntax

```
tcman.sh start [general_options] [-D | -J]
```

## Parameters

### general\_options

Specify one or more of the general TCMAN options. Run `tcman.sh help start` to see which general options are appropriate.

`-D`

Start the server in Tomcat debug mode. `-D` overrides the `-J` option.

`-J`

Start the server in debug mode using the JDPA (Java Platform Debugger Architecture) APIs for debugging. `-J` cannot be used if the `-D` option is specified.

Before you run a server with the `-J` option, you must define a port for the JDPA debugger by setting the `JDPA_ADDRESS` environment variable to a unique TCP network port number.

## Example

Start the server in `/psc/acme1`, which is an instance of the core server in `/psc/pashome`:

```
/psc/acme1/bin/tcman.sh start
Using CATALINA_BASE:      /psc/acme1
Using CATALINA_HOME:      /psc/pashome
Using CATALINA_TMPDIR:    /psc/acme1/temp
Using JRE_HOME:           /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID:       /psc/acme1/temp/catalina.pid
```

## Notes

- When the TCMAN utility starts the server, it verifies the creation of the OS process and then records the server's process-id in a `.pid` file. The location of the `.pid` file is:

OS	PID File Path
UNIX	<code>\$CATALINA_BASE/temp/catalina-instance_name.pid</code>
Windows	<code>\$CATALINA_BASE\logs\catalina-instance_name.pid</code>

- You can obtain the process id of a server by running the TCMAN `env` action.

## Stop an instance (stop)

### Purpose

Stop a running instance, either gracefully or forcibly.

**Note:** TCMAN supports stopping a server instance that is not configured with a shutdown port.

On UNIX platforms stopping the running server instance is accomplished by sending a UNIX signal to the PAS process. Therefore, the administrator's process must have the UNIX permissions to signal the PAS process. On Windows platforms, the instance is identified using an OS process id that is used to stop server processes.

### Syntax

```
tcman.sh stop [general_options] [-F [-w seconds]]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help stop` to see which general options are appropriate.

`-F`

Kill the sever process if it does not stop after a default wait time (5 seconds on UNIX, 10 seconds on Windows). Change the default wait interval by using the `-w` option.

`-w seconds`

Optionally specify the number of seconds to wait before killing a server process.

### Example

Stop the server in `/psc/acme1`, which is an instance of the core server in `/psc/pashome`:

```
/psc/acme1/bin/tcman.sh stop
Using CATALINA_BASE:  /psc/acme1
Using CATALINA_HOME:  /psc/pashome
Using CATALINA_TMPDIR: /psc/acme1/temp
Using JRE_HOME:        /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID:    /psc/acme1/temp/catalina.pid
```

### Notes

- TCMAN supports stopping a server instance that is not configured with a shutdown port.

On UNIX platforms stopping the running server instance is accomplished by sending a UNIX signal to the PAS process. Therefore, the administrator's process must have the UNIX permissions to signal the PAS process. On Windows platforms, the instance is identified using an OS process id that is used to stop server processes.

The following is an example a message you would see after a forced shut down with no shut down port:

```
Sep 23, 2013 4:10:47 PM org.apache.catalina.startup.Catalina stopServer
SEVERE: No shutdown port configured. Shut down server through OS signal.
Server not shut down.
Killing Tomcat with the PID: 14230
```

- Process ids are stored in the following locations:

OS	PID File Path
UNIX	\$CATALINA_BASE/temp/catalina-instance_name.pid
Windows	\$CATALINA_BASE\logs\catalina-instance_name.pid

- You can also obtain the process id of a server by running the TCMAN `env` action.

## Display server, OS, and runtime version information (version)

### Purpose

Show the Apache Tomcat runtime version and OS information for an instance.

To execute this action, the instance cannot be running

### Syntax

```
tcman.sh version [general_options]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help version` to see which general options are appropriate.

## Example

Display the server and runtime information for `acme1`, an instance of the core server installed in `/psc/pashome`:

```
$: /psc/pashome/bin/tcman.sh version -I acme1
Using CATALINA_BASE:      /psc/acme1
Using CATALINA_HOME:      /psc/pashome
Using CATALINA_TMPDIR:    /psc/acme1/temp
Using JRE_HOME:           /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/users/doc/agarbacz/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID:       /psc/acme1/temp/catalina.pid
Server version: Apache Tomcat/7.0.42
Server built:  Jul 2 2013 08:57:41
Server number: 7.0.42.0
OS Name:      Linux
OS Version:   2.6.18-164.el5
Architecture: amd64
JVM Version:  1.7.0_02-b13
JVM Vendor:   Oracle Corporation
```

## Test a server configuration (test)

### Purpose

Displays information on the configuration and environment of an instance. It also displays information about error conditions.

The `test` action starts a server (instance), loads all the configuration files, and then displays information. The instance is stopped, exiting gracefully even if there is an error condition.

To execute this action, the instance cannot be running

### Syntax

```
tcman.sh test [general_options]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help test` to see which general options are appropriate.

## Example

Run a test of the configuration of `acme1`, which is an instance of the core server installed at `/psc/pashome`:

```
$: /psc/pashome/bin/tcman.sh -I acme1 test
Using CATALINA_BASE:      /psc/acme1
Using CATALINA_HOME:      /psc/pashome
Using CATALINA_TMPDIR:    /psc/acme1/temp
Using JRE_HOME:           /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID:       /psc/acme1/temp/catalina.pid
. . .
```

## Notes

The `test` action is particularly useful for testing to verify that a server will start and run properly after you make changes to configuration and properties files.

# General actions

This section details the actions available for displaying help and server runtime environment information.

## Display help (help)

### Purpose

Display summary or detailed help for all TCMAN actions, property names, and server features.

### Syntax

```
tcman.sh help [ action | property | feature ]
```

### Parameters

*action*

Show the syntax and options of the specified action. If no action is specified, show a list of all actions and the general options.

*property*

Show the settings for specified property.

*feature*

Show if the specified feature is enabled or disabled.

## Example

Display the usage help for the `create` action:

```
$: tcman.sh help create
usage: tcman create [options] -p <http-port> [instance-opts] <new-base-path>

instance-opts:
  [-s <shutdown-port>]
  [-P <https-port>]
  [-j <ajp13-port>]
  [-W <web-apps-dir>]
  [-N <inst-alias-name>]
  [-U <file-owner> -G <file-group>]

general options:
  -u uid:pwd  pass uid and pwd for HTTP BASIC authentication
  -v          print verbose output
  -M url      override the CATALINA_BASE manager's URL with
              <{http|https}://<host>:<port>/<mgr-app>
  -B          override CATALINA_BASE environment setting
  -n          debug run action but do not execute changes
```

## Display runtime environment information (env)

### Purpose

Show details about a server's state.

### Syntax

```
tcman.sh env [general_options] [keyword]
```

### Parameters

*general\_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help env` to see which general options are appropriate.

*keyword*

Specify one or more keywords that represent the name of the state that you want to view. If no keyword is specified, then all of the state information is displayed.

Keywords include:

Keyword	Description
running	Indicate if a server is running ( 1 ) or not running ( 0 ).
mgrurl	Display the URL of the manager application.

Keyword	Description
type	Display the server type.
alias	Display the server's alias.
parent	Display the pathname of the parent of an instance.
tracking	Indicate if tracking is on (1) or off (0).
http	Display the server's http port number.
https	Display the server's https port number.
shut	Display the server's shutdown port number. A value of -1 indicates that there is no shutdown port.
pid	Display the server's process id. A hyphen (-) indicates that the server is not running.

## Example

Display all of the state information for the instance created in /psc/acme1:

```
/psc/acme1/bin/tscman.sh env
catalina home:      /psc/pashome
catalina base:      /psc/acme1
java home:          /tools/linuxx86_64/java64/jdk1.7.0_02/
jre home:
manager http port: 8501
manager https port:8601
manager shut port: 8701
manager URL:        http://localhost:8501/manager
config type:        instance
config alias:       acme1
config parent:      /psc/pashome
server running:     0
instance tracking: 1
instance file:      /psc/pashome/conf/instances.unix
server process-id: -
```



---

# Setup and Administration for ISVs

---

ISV accounts for both hosted and private cloud support white labeling functionality and give you the ability to provision and manage customer tenants directly. A customer tenant is a virtual space that securely hosts applications and data and provides a working environment for a set of users. This allows you to run your own Software as a Service (SaaS) business and take advantage of the Rollbase platform for rapid development and secure deployment of applications that meet critical business needs.

## ISV Features

For ISV pricing or to purchase, [contact Progress](#). Progress Rollbase provides the following benefits to all ISV customers:

- ISV account: Hosted cloud standard customers receive a special account that provides the ability to create and manage customer tenants. This eliminates dependencies on Progress for deployment and maintenance. Rollbase Private Cloud customers can set up any number of ISV Accounts for their own customers.
- Fifty users included: Rollbase Hosted Cloud Standard and Rollbase Private Cloud Standard ISV licenses include 50 subscribers with the base monthly fee. You will be billed for subscribers 51 and beyond. Private Cloud Enterprise customers get their first 1000 users included with the base monthly fee.
- Custom login and password retrieval pages: ISVs with dedicated websites for their companies and products can host their own branded pages for log in and forgotten passwords, making the use of Rollbase transparent to end users. Any number of custom login pages can be created and hosted externally allowing you to have different login mechanisms for each application, dedicated login pages for key customers, or a central login for all of your customers.
- Custom Application Directory: You can set up a custom branded, private application directory allowing your customers and prospects to view and install applications directly from their website.

- Customizable platform name: It is simple to replace Rollbase with your own product, platform or company name.
- Customizable URLs: All default external links within the platform that point to company-hosted pages, such as terms of service, privacy statement, etc, can be redirected to your pages. Hosted cloud ISV customers will still see [www.rollbase.com](http://www.rollbase.com) in the browser address bar. Private Cloud customers will see [www.yourdomain.com](http://www.yourdomain.com).
- Automatically provisioned free trials: As an ISV, you can offer automatically-provisioned free 30 day trials of any application through dedicated pages on your website. This includes the ability to customize and brand automated emails sent from the Rollbase system.

The topics in this section describe how to work with Progress Rollbase Hosted Cloud or Private Cloud as an ISV.

For details, see the following topics:

- [Getting Started](#)
- [System Applications](#)
- [Creating a Custom Log In Page](#)
- [Creating a Page for Users to Retrieve Passwords](#)
- [Customizing Page Title Tags](#)
- [Using a Third-Party Cloud Service for Storage](#)
- [Using the ISV Partner Application](#)
- [Pushing Application Updates to Other Tenants](#)
- [Installing Application Updates](#)
- [Version History and Rolling Back](#)

## Getting Started

Rollbase hosted standard cloud customers receive an ISV account that allows tenant and application management. When the ISV account is created, you receive a welcome email from Rollbase that allows you to log in and start creating customer tenants.

When you log in, you will want to view and modify your ISV account settings. And, you can change these settings at any time. However, note that once you have customer tenants that changes made to ISV account settings will not be applied immediately. Please contact Rollbase support if you need these to be activated within 24 hours.

To view and modify your ISV settings:

1. From the drop-down list of applications, select **Setup Home**.
2. Under the **Personal Setup** heading, click **My Settings**

In addition to the usual Rollbase user settings, you will see the ISV-specific options described in the table below. All fields are optional.

Field	Integration Name	Description
ISV System Name	isvSysName	Name of your company. This value will replace "Rollbase" throughout the platform.
ISV Home Page	isvHome	Absolute URL to your website to use instead of Rollbase.com.
		<b>Note:</b> When you set an ISV Home Page URL, Rollbase appends the privacy and terms Web page URLs to its footer. The privacy and terms Web page URLs created for your ISV Web page URL are <ISV Home Page>/privacy/ and <ISV Home Page>/terms/ respectively. Ensure that you host your privacy and terms Web page content in the aforementioned URLs. Note that, if the ISV Home page URL is not set or is not an absolute Web address, the privacy and terms pages do not appear in your Home page.
ISV Login URL	isvLoginUrl	URL to your custom login page
ISV Copyright	isvCopyright	Copyright string to use at the bottom of all pages
ISV Support URL	isvSupportUrl	URL to "Support Requests" link to use instead of Rollbase.com
ISV Title Template	isvTitleTemplate	Template to use for the <title> tag in all pages

**ISV Partner**

Changes to ISV-related parameters will be in affect after next server restart.

ISV System Name  Acme ISVISV Home Page  www.acmelsv.comISV Login URL  www.acmelsv.com/login/customLogin.htmlISV Copyright  Copyright © 2010 Acme ISV. All rights reserved.ISV Title Template  {IR} - {IO}

# System Applications

When a Rollbase Customer (i.e. tenant) is created, one or more "system" Applications is installed. A system Application must always contain the User Object, which is necessary for the functioning of every Rollbase Tenant (otherwise no users would be able to login). System Applications also include other important components which cannot be created using regular setup functions:

- The Reports list Page and Menu
- The "System Settings" Object definition
- The Location, Department and Function (LDF) Object definitions

## Creating a Custom Log In Page

A hosted or private cloud ISV account gives you the option to create and host a custom login page. This page can be hosted on another website to allow users to log into a Rollbase customer tenant.

The login page should contain forms that store the user name and password, post them to Rollbase and handle both successful and unsuccessful log in attempts.

Please contact Rollbase support or your account manager if you wish to see examples built by some existing Rollbase Hosted Standard or Private Cloud customers.

Custom HTML login pages must contain a FORM tag that includes the following:

- An `action` attribute pointing to `https://www.rollbase.com/router/servlet/Router`.
- A `loginName` parameter that will store the login name specified by a user.
- A `password` parameter that will store the password specified by a user.
- An `act` hidden parameter with a value of `login`.
- An `rt` hidden parameter points to the full URL of your custom login page (this parameter stands for "return to" and is stored as part of the user's session, so when they log out or their session expires the user is automatically taken back to your custom login page rather than `Rollbase.com`).
- An `errMsg` parameter is optional. If this HTTP parameter is set in the Login Page, URL errors, such as invalid user name, will be captured and displayed as error messages.

If a login is unsuccessful, Rollbase will redirect the user back to your custom login page and then display the error description set in the HTTP parameter `errMsg`. Progress strongly recommends that your custom login page capture and display this error message when it is non-empty.

- An `o` parameter is optional. If this HTTP parameter is set in the Login Page URL, it will be captured and stored in the `o` hidden parameter. For example, this parameter might contain the location of the first landing page.

The following example shows a simple login page:

```
<html>
<head>
```

```
<title>Custom Login Page</title>
</head>
<body>

<form action="https://www.rollbase.com/router/servlet/Router" method="post"
name="theForm">
<input type="hidden" name="act" value="login">
<input type="hidden" name="rt"
value="https://www.acmeisv.com/login/customLogin.html">

<table cellpadding=0 cellspacing=0>

<tr><td><h2>Secure Customer Login</h2></td><tr>
<tr height='10'><td></td></tr>

<tr>
<td nowrap><b>User Name:</b> </td>
<td><input type="text" maxlength="80" size="30" name="loginName"></td>
</tr>
<tr height='5'><td></td></tr>
<tr>
<td nowrap><b>Password:</b> </td>
<td><input type="password" maxlength="80" size="30" name="password"></td>
</tr>
<tr height='5'><td></td></tr>
<tr>
<td></td>
<td alignment='left'><input type="submit" value=" Login " name="btns"></td>
</tr>

</table>
</form>

</body>
</html>
```

## Creating a Page for Users to Retrieve Passwords

A hosted or private cloud ISV account gives you the option to create a page for users to retrieve forgotten passwords. Such a page must contain HTML with a FORM tag satisfying the following conditions:

- An `action` attribute pointing to  
`https://www.rollbase.com/router/login/forgetPassword2.jsp`.
- A `loginName` input parameter that stores the login name entered by a user.
- An `email` input parameter that contains the email address entered by a user (must match to login name).
- An `act` hidden parameter with a value of `forgotPassword`.
- An `rt` hidden parameter that points to the full URL of your custom login page.

The following example shows a very simple password retrieval page:

```
<html>
<head>
<title>Custom Login Page</title>
</head>
<body>
```

```
<form action="https://www.rollbase.com/router/login/forgotPassword2.jsp"
method="post" name="theForm">
<input name="act" value="forgotPassword" type="hidden">
<input type="hidden" name="rt"
value="https://www.acmeisv.com/login/customLogin.html">

<table cellpadding=0 cellspacing=0>

<tr>
  <td><h2>Forgot your password?</h2></td>
<tr>

<tr>
  <td colspan="2">In order for us to reset your password we need to confirm
your identity. Please enter your user name. <br>You will receive an email with
a new temporary password.</td>
</tr>

<tr>
  <td nowrap><b>User Name:</b> </td>
  <td ><input name="loginName" size="30" type="text"></td>
</tr>
<tr>
  <td nowrap><b>Email Address:</b> </td>
  <td ><input name="email" size="30" type="text"></td>
</tr>

<tr height='5'><td></td></tr>
<tr>
  <td></td>
  <td alignment='left'><input type="submit" value=" Submit " name="btns"></td>
</tr>

</table>
</form>
</body>
</html>
```

## Customizing Page Title Tags

A hosted or private cloud ISV account gives you the option to customize the `<title>` tag used in application pages that will be accessed by your customer tenants. Valid templates may include any text and the following merge tokens that get replaced at runtime:

- `{ !A }` - Current Application Name
- `{ !S }` - ISV System Name (see above)
- `{ !P }` - Customer's Subscription Plan Name (Solo, Professional, etc.)
- `{ !O }` - Current Object name (if any)
- `{ !R }` - Current Record name (if any)

# Using a Third-Party Cloud Service for Storage

By default, for ISV and Private Cloud users, Rollbase backs up data on a local drive. You can configure the backup location in your location drive from the **System Console**. The **Storage server** component specifies the backup location value. Starting Rollbase 3.0, you can set the data backup limit per backup by specifying the value for `storageUsageLimitForBkp` property in the `shared.properties` file. For more information on `shared.properties` file and its properties, see [shared.properties](#) on page 505.

As an alternative to local storage, you (an ISV or a Private Cloud user) can configure your customer tenants to use Amazon S3 or Microsoft Azure on a tenant by tenant basis by enabling a third-party cloud storage on the customer tenant record.

---

**Note:** Using a third party cloud service for storage can impact performance because files and images are not served directly by Rollbase but from that third party. Progress recommends that you test the performance of your selected storage provider before deploying cloud storage in production scenarios.

---

After you have enabled cloud storage, the files will be moved to your storage account in an asynchronous mode. The tenant administrator will receive an email with summary of the moved files. After that, you can delete the local files to save disk space.

All errors related to third-party cloud services are logged in the `storage.log` file. Please review this file if you're having problems with remote storage.

See the following topics for more information:

- [Using Amazon S3](#)
- [Using Microsoft Azure](#)

## Using Amazon S3

To use the Amazon S3 service for file storage, you need to have an S3 subscription and you must have at least one bucket, your Access Key ID and Secret Access Key. You can retrieve the S3 ID and S3 Key values for your AWS account by logging in and navigating to the **Security Credentials** screen.

Account

- Account Activity
- AWS Identity and Access Management
- AWS Management Console
- Consolidated Billing
- DevPay
- Manage Your Account
- Payment Method
- Personal Information
- Security Credentials**
- Usage Reports
- Billing Alerts
- Billing Preferences

Welcome Matt Robinson | Sign Out  
Account Number 1234567890123456

This page allows you to manage the root account credentials for your AWS Account. To manage IAM Users, their permissions, and security credentials, use the AWS Management Console.

Access to applications and services within AWS cloud is secure and protected in multiple ways. Accessing those applications and services requires the use of special credentials that are associated with your account. There are three types of credentials currently offered by AWS. If you know which security credentials you need, simply select one of the links below:

- Access Credentials: Your Access Keys, X.509 Certificates, and Key Pairs
- Sign-In Credentials: Your E-mail Address, Password, and AWS Multi-Factor Authentication Device
- Account Identifiers: Your AWS Account ID and Canonical User ID

If you are not sure which security credentials you should use, the link below will help you identify the credentials you need for the task you want to accomplish:

**Find out which AWS Security Credentials you need**

### Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

**Access Keys** **X.509 Certificates** **Key Pairs**

Use access keys to make secure REST or Query protocol requests to any AWS service API. We create one for you when your account is created — see your access key below.

**Your Access Keys**

Created	Access Key ID	Secret Access Key	Status
November 9, 2012	ASWp1t9p0u2z7YLAEM	Show	Active (Make Inactive)
Create a new Access Key			

For your protection, you should never share your secret key. Practice recommends frequent key rotation.

[Learn more about Access Keys](#)

## Enabling S3 Storage

To enable Amazon S3 Storage per customer tenant, follow these steps:

1. Navigate to the **ISV Partner** application.
2. From the **Customers List** on the **Customers** tab, find the customer tenant for whom you want to enable S3 storage and click **Edit**.
3. Scroll down to the **Cloud Storage** section and from the **Cloud Storage** drop-down, select **Amazon S3**.

**Note:** If you do not see the following fields when you edit a customer tenant, use the **Page Editor** to add them from the **Available Components** section.

<b>Cloud Storage</b>	
Cloud Storage	Amazon S3
S3 ID/Account Name	3WGQOWVT6H5GWGQ
S3 Bucket	rollbase
S3 Key/Account Key	XGjkrWLClrOCAi5bNn/l

4. Enter the appropriate values in the following fields:
  - S3 Bucket:** the S3 bucket to use for this customer tenant.
  - S3 ID:** the **Access Key ID** assigned to your AWS account.
  - S3 Key:** the Secret Access Key associated with the **Access Key ID**.

5. If you want existing files to be moved from local storage to Amazon, check the box **Move local files to Cloud storage**. Otherwise, only files stored in the future will be stored in the cloud.
6. For debugging purposes only, uncheck **Secure** to disable SSL. For production systems, you should leave this box unchecked.

## Using Microsoft Azure

To use the Microsoft Azure service you need to have an Azure account and you will need to specify the following information:

- Name of your Azure account. Optionally you can add "/" followed by customer-specific text prefix which will be added to container names used by customer. Prefixes allow storage for several customer tenants to use the same Azure account.
- The **Account Key**, which plays the role of a password. Click the **Manage Keys** link at the bottom of the **Azure Management Portal** page to obtain this key.

Note: Azure account and prefix names can only contain lower-case letters, digits, and hyphens "--" (but not two hyphens in sequence). The length of the prefix cannot exceed 16 characters. The system will automatically format specified prefix to match these Azure limitations.

The following examples illustrate how the Azure storage area relates to the specified **Account Name**:

Account Name	Azure Name	Azure Storage
rollbase	Rollbase	data template
rollbase/test	Rollbase	testdata testtemplate
rollbase/TEST_#1	Rollbase	test-1data test-1template

### Enabling Azure Storage

To enable Amazon S3 Storage per customer tenant, follow these steps:

1. Navigate to the **ISV Partner** application.
2. From the **Customers List** on the **Customers** tab, find the customer tenant for whom you want to enable S3 storage and click **Edit**.
3. Scroll down to the **Cloud Storage** section and from the **Cloud Storage** drop-down, select **Microsoft Azure**.
4. Enter the appropriate values in the following fields:
  - **S3 Bucket**: leave this field empty, it does not apply.
  - **S3 ID/Account Name**: the name of your Azure account.
  - **S3 Key/Account Key**: the **Account Key** from your Azure account.

5. If you want existing files to be moved from local storage to Azure, check the box **Move local files to Cloud storage**. Otherwise, only files stored in the future will be stored in the cloud.
6. For debugging purposes only, you can disable SSL. For production systems, you should leave this box unchecked.

## Using the ISV Partner Application

When logging into an ISV account, you will automatically be taken to the **ISV Partner** application that includes tabs for managing the following objects:

- **Customers:** Allows you to create and manage customer tenants.
- **Published Applications:** Allows you to manage applications published by customer tenants to your **Application Directory**. Here you can edit published application information such as name and description and choose whether or not they should appear by approving them for installation.
- **Support:** Allows you to manage and respond to support tickets submitted by your customers

### Sharing User Object Fields with Customer Tenants

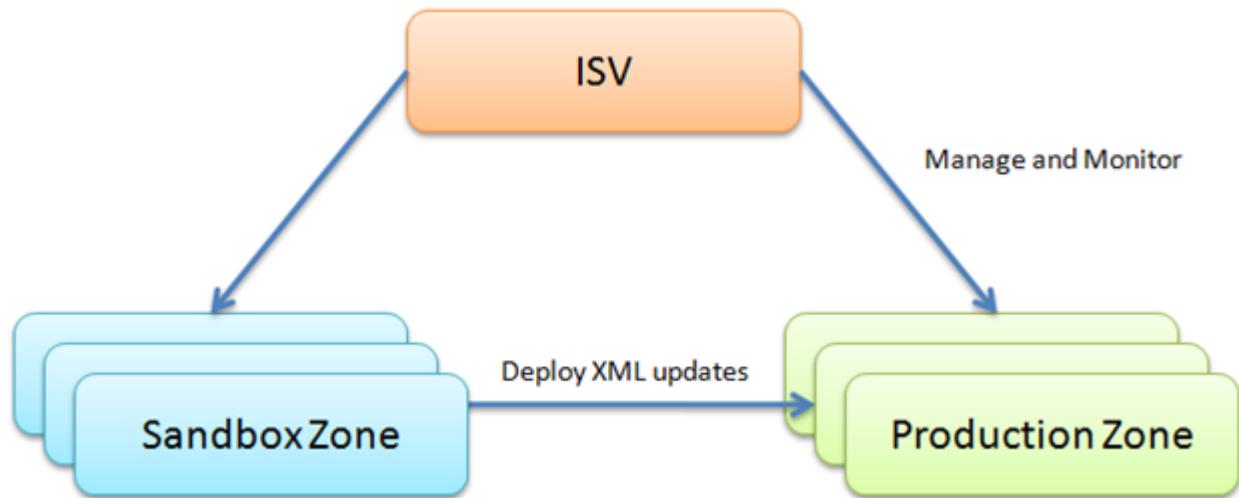
Several fields defined on the **User** object in the Master system are designed to share important information with ISV-managed Customers. See the table in [Getting Started](#).

Private Cloud customers can define any fields to be shared between an ISV user and related customers. In the **User** object in the Master system, check **Make this field available for related ISV Customers** checkbox on the field create or field edit page for each field you want to expose. Once this is enabled, in any template or formula within that customer tenant you can use a merge field to retrieve the value of that Master system user field.

Warning: Due to the current `User` caching mechanism, updates to shared fields may not be available to customers until after the next Rollbase maintenance event (or Tomcat restart for Private Cloud customers).

### Using a Sandbox for Development

We recommend that all ISV customers create at least one dedicated customer tenant for development purposes (i.e. a sandbox) that applications will be published from.



Use sandbox customer tenants to develop and test your applications. Then use the XML publishing mechanism described in [Publishing and Distributing Applications](#) on page 407 to publish your application and deploy/update each application in your production Customers.

## Creating and Managing Customer Tenants

The **ISV Partner** application contains **Customer** records to define customer tenants. To add a customer tenant, select the **Customer** tab, and click **New Customer**. Complete as much information as possible and provide data for the first administrative user. This will initiate the tenant creation process and the administrative user will receive a welcome email with login information and a temporary password when the tenant creation process is completed.

To prevent users from modifying applications, create an account for yourself in each of your customer tenants and assign the **Administrator** role (with all customization capabilities) to yourself rather than to your users.

To manage customer tenants, from the **Customers List**, click the customer name. The available actions include the following:

- The **Login** button logs you in to a customer tenant as a super-admin, an invisible user with full access.
- From the **More actions** drop-down, the **Login As** option allows you to log in to a customer tenant as a particular user.
- The **View Logs** button allows you to view system logs that are useful for troubleshooting.
- From the **More actions** drop-down, the **Reindex** option allows you to re-create the full-text search index for that customer.

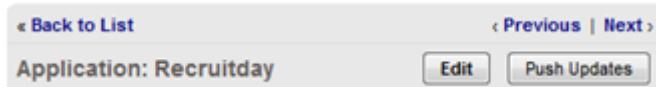
## Pushing Application Updates to Other Tenants

Publishers who have a hosted ISV account, or Private Cloud customers with sufficient permissions, can push application updates into any tenant where that particular application has already been installed.

**Note:** Try to choose a period of zero or low usage for pushing application updates in order to minimize the impact they have on user sessions.

Here's how it works:

1. Publish your application to the Application Directory as version 1.
2. Make sure your application is approved and available for installation (edit the Published Application record accordingly).
3. Install this application into one or more target tenants.
4. Publish updates to your application as needed. Each update will increment your app's version #.
5. Once you are ready to push updates to all of your tenants who have this app installed, go to the Published Application view page for that app in the Master system and click the "Push Updates" button.
6. On next page check "Override Changes" box if you want changes in your application to override any customization made in target Tenants (this option is always checked for fully managed applications).
7. Updates will be pushed into all target Tenants who have lower version # of that application installed.
8. Pushing updates will not override changes made by administrators on target Tenants.
9. If your application is fully locked, elements deleted from the original application (fields, triggers, etc.) will be automatically deleted from target Tenants. If your application is not locked, no elements will be deleted in target Tenants.
10. Updates will be scheduled for installation in an asynchronous fashion. After an update is completed in a particular tenant, that tenant will be reloaded in memory and all users must establish new sessions (for this reason we recommend scheduling a time to push updates during off hours for your users).



## Installing Application Updates

After installing an application you may need to update it periodically to the most recent version from within the tenant (rather than pushing updates to all tenants from the Master system). This can be done one of two ways:

- If you installed the application from the Applications Directory, click **Check for Updates** in the application's view page. If a newer version exists, you will be prompted to install it from the Applications Directory.
- If you installed an application directly from XML, obtain the newer version and upload it in using the same "Install or Update Application from XML" page (follow the same steps you used to install this application to begin with).

In both instances, you can specify whether you want Rollbase to avoid overwriting existing unlocked components with changes in the latest version of the application by unchecking the box next to them. Locked applications and locked components will always receive these updates, regardless of the state of this checkbox. The **Override Changes** checkbox also controls overwriting, see [Installing and Updating from the Application Directory](#) on page 113

## Version History and Rolling Back

When an application publisher uploads a new version of a published application to the **Application Directory**, Rollbase preserves the current version of the application XML as a related **Application History** record. Application XML from these history records can be viewed at any time. Rollbase Administrators can also roll back a published application to one of the older versions. The roll back action will do the following:

- Move the current application XML into a new **Application History** record
- Replace the current application XML with that of the selected older version
- Increment the current **Published Application** version #

The roll back action simply creates a new version out of an older version of the application's XML, and makes the selected version current. This action does not actually push updates to any tenants, but it does everything to prepare you for doing this.

For example: If the current version # of your app is 3 and you choose to roll back to version 2, the result will be version 4 which is identical to version 2. There are several reasons for treating the rolled back version as a new app version #, primarily so that all tenants who have already installed that app are automatically aware that there is a new version available and so that the "push updates" process rolls out the correct version of the app. Once a roll back is done, the application XML can then be pushed to Customers using the standard "Push Updates" button (see next section).

Application History			
Action	Version	User	Date
<a href="#">View</a>   <a href="#">Roll Back</a>	10	Pavel Vorobiev	10/04/2012 07:45 PM
<a href="#">View</a>   <a href="#">Roll Back</a>	9		10/03/2012 08:42 PM
<a href="#">View</a>   <a href="#">Roll Back</a>	8		10/03/2012 08:40 PM
<a href="#">View</a>   <a href="#">Roll Back</a>	7		09/21/2012 10:54 AM

If you do not see the **Application History** section as shown above when viewing a published application, edit that page and create a new section, then drag in the **Application History** component from the **Available Components** section in the sidebar.



---

# Reference

---

Topics in this section describe Rollbase APIs, built-in CSSs, and field types.

For details, see the following topics:

- [Server-side API](#)
- [Client-side AJAX API](#)
- [Code Generator](#)
- [Metadata API and XML Reference](#)
- [Rollbase REST Methods](#)
- [Rollbase SOAP Methods](#)
- [Rollbase CSS Styles](#)
- [Field Types](#)

## Server-side API

This section describes server-side APIs that can be used in formulas. Access Control permissions apply to server-side query and trigger APIs.

Access permissions are checked for:

- The Current user (Portal User Role for Portals)
- The System Role "Server API"

If neither the current user nor the Server API role has appropriate permissions to access the API, the API call fails.

For more information about security and access permissions, see [Security and Access Control](#) on page 375.

**Table 6: Access Required for Query and Trigger API Calls**

API	Access Required
selectQuery()	VIEW all records of given object type
selectValue()	VIEW all records of given object type
selectNumber()	VIEW all records of given object type
getCount()	VIEW all records of given object type
selectCustomerQuery()	VIEW customer records on the master server
selectCustomerValue()	VIEW customer records on the master server
selectCustomerNumber()	VIEW customer records on the master server
getFieldValue()	VIEW current records
getNumFieldValue()	VIEW current records
getBinaryData()	VIEW current records
getTextData()	VIEW current records
getRelatedFields()	VIEW current records
runTemplate()	VIEW current records
createRecord()	CREATE record of given object type
setFieldValue()	EDIT current record
setBinaryFieldValue()	EDIT current record
setTextFieldValue()	EDIT current record
updateRecord()	EDIT current record
runTrigger()	EDIT current record
createActivityLog()	EDIT current record

API	Access Required
attach()	EDIT two records with relationship
detach()	EDIT two records with relationship
deleteRecord()	DELETE current record

## API Error Messages

If parameters passed to API calls are incorrect or inconsistent, the Rollbase scripting engine will generate error messages. The following table lists common error messages and suggested remedies.

Message	Description
You do not have permission to perform this action	Neither the current user nor the Server API Role has permission to perform requested action.
Object definition XXX not found	Use a list of objects and make sure that the passed object integration name is correct.
XXX with id 123456 not found	Ensure that the passed numeric ID belongs to existing records. In most cases use the template token for the current record ID, <code>record {!id}</code> , or a related record <code>{!R123456}</code> .
Field YYY not found	Ensure that the field integration name is correct.
Relationship definition RRR not found	Ensure that the relationship integration name is correct.
Trigger TTT not found	Ensure that the trigger integration name is correct.
User with id 3456768 not found	In most cases, use the token ID for the current user <code>{!#CURR_USER.id}</code>
Object Definition ID 13218 is different from required 13277	To retrieve a record, pass pair of object name and record ID. If these values do not match, an error will result. Ensure that the object name and record ID match.

## Query API

The Rollbase Query API is available for all server-side formulas. To access these methods use the `rbv_api` system object. The order of query parameters is set and cannot be changed.

Queries support the following syntax:

```
SELECT expression FROM object_name {WHERE expression} {GROUP BY expression} {ORDER BY expression}
```

The **SELECT** statement consists of the following parts:

- **SELECT** lists columns or expressions to be selected (mandatory)
- Use field integration names as SQL column names. You can use expressions such as `COUNT(1)`. Selecting all columns with star (\*) is not supported.
- **FROM** clause must consist of exactly one and only one object name (mandatory).
- **WHERE** clause includes a valid SQL expression to narrow the selection (optional). Use field integration names as SQL column names.
- **GROUP BY** clause includes an expression (typically a valid field name) to group selection (optional).
- **ORDER BY** clause includes a valid SQL expression to order selection (optional).

## Query Limitations

Only data fields with stored input values (text, decimal, etc.) can be used in a **SELECT** expression. Dependent fields, such as formulas, cannot be used in query methods. A relationship (i.e. Lookup) field can be used, but will only take the ID of the first related record--not an array of all related records. Related fields can be used if they point to a data field.

The limitation preventing formulas from being used in the Query API can be easily bypassed in simple cases.

Consider this Formula: `{!amount} * {!price}`

Though you cannot use this Formula in a query directly, you can create a query such as:

```
SELECT SUM(amount*price)  
      FROM order WHERE ...
```

For conditions involving date fields, you can use special tokens:

- **TODAY** for the current time
- **WEEK** for 12 AM of last Sunday
- **MONTH** for 12 AM of 1st day of the current month
- **QUARTER** for 12 AM of 1st day of the current quarter
- **YEAR** for 12 AM of 1st day of the current year
- **CURR\_USER** for id of the currently logged in user

You can also use built-in functions:

- `#YEAR(date)` returns the date's year as an integer
- `#MONTH(date)` returns the date's month as an integer in the range of 1 - 12
- `#DAY(date)` returns the date's day of the month as an integer
- `#ISO(literal_string)` returns the date or date/time in ISO format, which can be used in a query. Examples: `#ISO(2013-09-20T)`, `#ISO(2013-09-20T20:05:01Z)`
- `#IF(expr, val1, val2)` returns `val1` if `expr` is evaluated to true, `val2` otherwise.

The following query selects records created in the current year (starting from midnight January 1st) by current user:

```
SELECT name, id FROM order WHERE
  createdAt>=YEAR && createdBy=CURR_USER
```

Query methods do not support arithmetic operations with data tokens like view filters do. For instance, MONTH-1 does not represent the first day of the previous month. You should use a query with parameters instead. If a query includes a Date or Date/Time column, an instance of the JavaScript Date class is returned. You can use any JavaScript Date class methods. See [selectValue\(\)](#) for a code example.

You can also use single-quoted integration codes for picklists and status fields in a query WHERE clause. For example:

```
SELECT count(1) FROM order WHERE status='Q'
```

The following example reads records of where the user role integration code equals SALES:

```
SELECT loginName, role,
  email FROM USER WHERE role = 'SALES'
```

Object and field integration names are case-sensitive, while other components of an SQL query are not. When an integration code is used to reference pick list item, status, or role in the query, the corresponding item must actually exist and be unique. This means that two picklists in the query object cannot use the same integration code. Otherwise, the integration code will not be resolved and will likely cause a query error.

**Retrieving Picklist Integration Codes** If you query values of picklist fields you will get numeric values corresponding to the ID of a selected item. If you wish to receive the integration code of the selected item instead, add the #code suffix to the name of the picklist field. Add the #value suffix to extract the display name of the selected item. The same syntax can be used for multi-select picklists, status fields, radio Buttons etc. For example:

- `SELECT my_picklist FROM invoice WHERE id={!id}:` returns the numeric ID of selected item
- `SELECT my_picklist#code FROM invoice WHERE id={!id}:` returns the integration code of selected item
- `SELECT my_picklist#value FROM invoice WHERE id={!id}:` returns the display name of selected item

Examples of valid queries for the USER Object:

- **Select fields for users whose name starts with 'M':** `SELECT id, name, updatedAt, updatedBy FROM USER WHERE name LIKE 'M%' ORDER BY name`
- **Count number of users whose name starts with 'M':** `SELECT count(1) FROM USER WHERE name LIKE 'M%'`
- **Count number of users created or updated in current quarter:** `SELECT count(1) FROM USER WHERE updatedAt>=QUARTER`
- **Count number of approval records for current record (identified by {!id} token):** `SELECT count(1) FROM $APPROVAL WHERE approvedRecord={!id}`
- **Summarize amount\*price expression for all records per category (picklist):** `SELECT sum(amount*price), category FROM sales GROUP BY category`

- Extract records created between two dates:

```
var d1 = new Date('2/2/2014');
var d2 = new Date('4/2/2014');
var arr = rbv_api.selectQuery(
  "SELECT id, name FROM a1 WHERE createdAt BETWEEN ? AND ?",
  100, d1, d2);
rbv_api.printArr(arr);
```

Example of an HTML report that displays the total number of users in the system:

```
<html><h2>
  Total users: #EVAL[ rbv_api.selectValue("SELECT COUNT(1) FROM USER") ]
</h2></html>
```

Although UI views do not make distinction between NULL values and 0 for numeric fields, the Query API treats NULL values and 0 differently. To mimic UI behavior you need to explicitly filter NULL values. For example:

```
SELECT count(1) FROM invoice WHERE amount=0 OR amount IS NULL
```

Access control applies to query methods. See [Security and Access Control](#) for more details. Passwords cannot be retrieved through the Query API due to security precautions. When applying these statements to lookup fields, the `selectQuery()` and `selectValue()` methods return the first related ID. If you need to retrieve all related IDs, use the `getRelatedIds()` method.

Some security precautions related to queries:

- 5000 characters or less
- Cannot include separators: ; \n /
- Cannot include reserved words: ALTER, BEGIN, CALL, CASCADE, DELETE, DROP, DATABASE, GRANT, EXECUTE, INSERT, REVOKE, RENAME, SHUTDOWN, SCHEMA, UPDATE, LOGIN\_NAME, PASSWORD

You can also use queries with parameters (see the method descriptions). Modifying the previous example to use parameter results in these statements:

```
count(1) FROM USER WHERE name LIKE ?SELECT
```

Then, supply the SQL query parameter 'M%'

You can perform JavaScript calculations and then pass results to a SQL query with parameters. This technique has a number of advantages over embedding variables directly into the text of a SQL query. To select ID of related record you can use query similar to this:

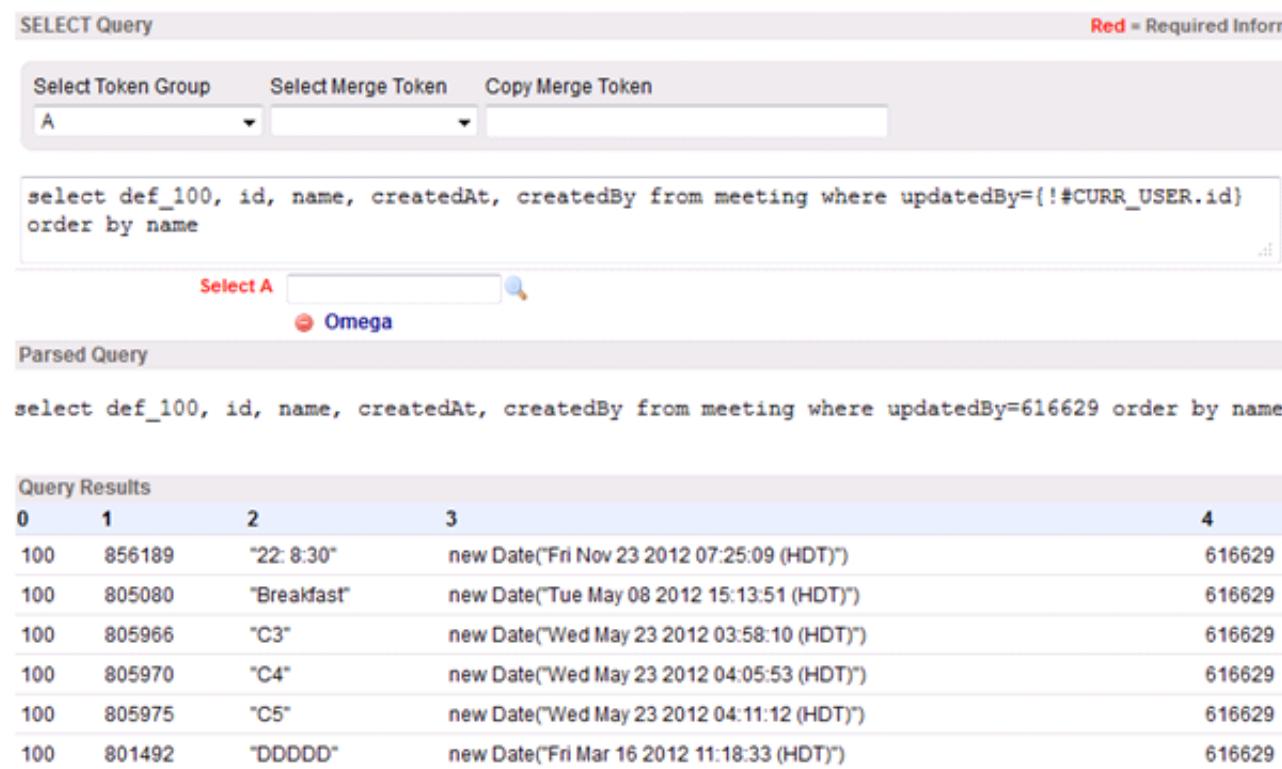
```
select R1234 from
  invoice where id={!id}
```

However please keep in mind that this query will only work for single relationships (1:1 and N:1). For multiple relationships (1:N and N:N) result of this query is undetermined. You should use `getRelatedIds()` or a template loop.

## Using the UI to Design a Query

To design and test your query you can use specially designed helper page. Click the **Test Query** button below formula text area to pop up that helper page. Enter your query starting with the `SELECT` keyword. Use merge tokens corresponding to all available fields and objects, as well as available SQL operands and built-in functions. To test your query, select a record of the current object type and click **Test**. The system will display the parsed query and results (up to 10 rows) or error message. When you finish your testing - copy the resulting SQL query into the body of your formula as the first parameter to one of the query methods.

The following shows an example query:



The screenshot shows a user interface for a SELECT query. At the top, there are three buttons: 'Select Token Group', 'Select Merge Token', and 'Copy Merge Token'. Below these buttons is a dropdown menu set to 'A'. The main area contains a text input box with the following SQL query:

```
select def_100, id, name, createdAt, createdBy from meeting where updatedBy={!#CURR_USER.id}
order by name
```

Below the input box, there is a 'Select A' dropdown with the value 'Omega' and a magnifying glass icon. The text 'Parsed Query' is displayed above the resulting SQL query:

```
select def_100, id, name, createdAt, createdBy from meeting where updatedBy=616629 order by name
```

The bottom section is titled 'Query Results' and displays a table with six columns. The columns are labeled 0, 1, 2, 3, and 4. The data is as follows:

0	1	2	3	4
100	856189	"22:8:30"	new Date("Fri Nov 23 2012 07:25:09 (HDT)")	616629
100	805080	"Breakfast"	new Date("Tue May 08 2012 15:13:51 (HDT)")	616629
100	805966	"C3"	new Date("Wed May 23 2012 03:58:10 (HDT)")	616629
100	805970	"C4"	new Date("Wed May 23 2012 04:05:53 (HDT)")	616629
100	805975	"C5"	new Date("Wed May 23 2012 04:11:12 (HDT)")	616629
100	801492	"DDDDDD"	new Date("Fri Mar 16 2012 11:18:33 (HDT)")	616629

## rbv\_api.selectQuery()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Runs an SQL SELECT query and returns results as a 2D-array. For Private Cloud customers, the maximum number of rows to return can be configured: see [MaxReqsInQuery parameter in the shared.properties file](#).

### Syntax

```
rbv_api.selectQuery (query, maxRows, arg1, arg2...)
```

### Parameters

*query*

SQL SELECT query. See [Query API](#) on page 561 for examples and limitations.

*maxRows*

Maximum number of rows to retrieve (1-20,000 range)

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query result in a 2-D array

## Example

The following example uses `selectQuery()` to obtain Line Item object records. The WHERE clause with the value `R8011457=?` retrieves only records related to the current order. The actual ID is passed as parameter, which is more efficient because it embeds the value directly into the query. The `#code` suffix for the column `category` fetches the integration name instead of the ID.

```
var arr = rbv_api.selectQuery( "select name, amount, price,
    category#code from line_item where R8011457=?", 100, {!id});
var buff = "Name, Amount, Price<br>";
for (var i=0; i<arr.length; i++)
{ buff += arr[i][0]+", "+arr[i][1]+",
    "+arr[i][2] +", "+arr[i][3]+"<br>";
}
return buff;
```

## rbv\_api.selectQuery2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Similar to `selectQuery()`, `selectQuery2()` runs a query and returns the results as a 2D-array. The extra parameter, `rowFrom`, defines a starting point for results. This makes it possible to make several calls for objects that contain thousands of rows of data.

## Syntax

`rbv_api.selectQuery2 (query, rowFrom, maxRows, arg1, arg2...)`

## Parameters

*query*

SQL SELECT query. See [Query API](#) on page 561 for examples and limitations.

*rowFrom*

Number of row to start output (0 based)

*maxRows*

Maximum number of rows to retrieve (1-20,000 range)

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query result in 2-D array

## rbv\_api.selectValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Runs an SQL SELECT query and returns a single value. This simplified version of `rbv_api.selectQuery()` is useful for calculating sum and performing other operations on single values.

### Syntax

```
rbv_api.selectValue (query, arg1, arg2...)
```

### Parameters

*query*

SQL SELECT query. See examples below and see [Query API](#) on page 561 for limitations.

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query result as a single value

### Example

The following example calculates the most recent update date for a set of related records. It is used in a formula field of type Date:

```
rbv_api.selectValue("SELECT updatedAt FROM related_Object  
WHERE R474176={!id} order by updatedAt desc");
```

The following example performs the same calculation, but uses an `obj_id` parameter with the template token `{!id}` to obtain the ID of the current record. This formula field will run the SELECT query and bring back most recent value for the `updatedAt` field as a Date.

```
var obj_id = {!id};  
var d = rbv_api.selectValue("SELECT updatedAt FROM related_Object  
WHERE R474176=? order by updatedAt desc", obj_id);
```

The following example calculates the number of records updated in the previous month (relative to the current date). Please note that the `Verbose` flag is set for debugging and query parameters are used:

```
rbv_api.setVerbose(true);
var d1=new Date(rbv_api.firstDayOfMonth(0));
var d2=new Date(rbv_api.firstDayOfMonth(-1));

return rbv_api.selectNumber("SELECT COUNT(1)
    FROM my_object where updatedAt>=? and updatedAt<?", d2, d1);
```

## rbv\_api.selectNumber()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Runs an SQL SELECT query and returns a single decimal value. It is similar to the `selectValue()` method.

### Syntax

```
rbv_api.selectNumber (query, arg1, arg2...)
```

### Parameters

*query*

SQL SELECT query. See examples below and see [Query API](#) on page 561 for limitations.

*args*

Variable number of parameters passed to query (optional)

### Return Value

Query result as a decimal number

### Example

The following example selects the value in the Total field from the most recent Order record:

```
return rbv_api.selectNumber("SELECT total FROM order ORDER BY updatedAt DESC");
```

## rbv\_api.selectCustomerQuery()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Similar to [selectQuery2\(\)](#), but runs on the specified customer.

Do not use this API for views.

The caller must have VIEW permissions to access the customer record with given ID. This method throws an error if the target customer does not have the queried object installed. The `selectCustomerQuery()`, `selectCustomerValue()`, and `selectCustomerNumber()` methods might require cache loading and may be slow and are only available if called from a Master Server.

## Syntax

`rbv_api.selectCustomerQuery (custId, query, rowFrom, maxRows, arg1, arg2...)`

## Parameters

*query*

SQL SELECT query. See [Query API](#) on page 561 for examples and limitations.

*rowFrom*

Number of row to start output (0 based)

*maxRows*

Maximum number of rows to retrieve (1-20,000 range)

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query results as a 2-D array

## **rbv\_api.selectCustomerValue()**



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Similar to [selectValue\(\)](#) returns a single value, but runs on the specified customer.

The caller must have VIEW permissions to access the customer record with given ID. This method throws an error if the target customer does not have the queried object installed. The `selectCustomerQuery()`, `selectCustomerValue()`, and `selectCustomerNumber()` methods might require cache loading and may be slow and are only available if called from a Master Server.

## Syntax

`rbv_api.selectCustomerValue (custId, query, arg1, arg2...)`

## Parameters

*custId*

ID of the customer record to query

*query*

SQL SELECT query. See [Query API](#) on page 561 for examples and limitations.

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query result as a single value

## **rbv\_api.selectCustomerNumber()**



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Same as `selectNumber()`, but runs on the customer with the specified ID.

The caller must have VIEW permissions to access the customer record with given ID. This method throws an error if the target customer does not have the queried object installed. The `selectCustomerQuery()`, `selectCustomerValue()`, and `selectCustomerNumber()` methods might require cache loading and may be slow and are only available if called from a Master Server.

## Syntax

`rbv_api.selectCustomerNumber (custId, query, arg1, arg2...)`

## Parameters

*custId*

ID of the customer record to query

*query*

SQL SELECT query. See an example below and see [Query API](#) on page 561 for examples and limitations.

*args*

Variable number of parameters passed to query (optional)

## Return Value

Query result as a number

## Example

This example fetches number of records from a User object for a customer with the ID represented by the `{!id}` token:

```
var count = rbv_api.selectCustomerNumber({!id}, "SELECT COUNT(1) FROM USER");
```

## rbv\_api.getCount()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the number of records in the selected View. Optionally, this method can filter by field name and value.

### Syntax

```
rbv_api.getCount(viewId, filterName, filterValue)
```

### Parameters

*viewID*

Original ID of view

*filterName*

Optional field integration name to filter results

*filterValue*

Optional field integration value to filter results

### Return Value

Number of records

## Example

The following example returns the total number of records in view 123456 where the Status field equals Open (any filters imposed by the view apply as well):

```
var count = rbv_api.getCount(123456, "status", "Open");
```

## rbv\_api.getRelatedIds()

### Purpose

For native Rollbase objects, this method returns an array of related IDs for a specified relationship and object ID.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [rbv\\_api.getRelatedIds2\(\)](#) on page 572.

---

## Syntax

```
rbv_api.getRelatedIds (relName, id)
```

## Parameters

*relName*

Integration name of relationship

*id*

Object ID from one side of the relationship

## Return Value

An array of related IDs

## Example

```
var arr = rbv_api.getRelatedIds("R1321", {!id});
for (var k=0; k<arr.length; k++) {
  var id = arr[k];
  rbv_api.println("id="+id);
}
```

# rbv\_api.getRelatedIds2()

## Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), returns an array of related IDs for a specified relationship and object ID.

---

**Note:** For native Rollbase objects, you can use the method, [rbv\\_api.getRelatedIds\(\)](#) on page 571.

---

## Syntax

```
rbv_api.getRelatedIds2(relName, objName, objId)
```

## Parameters

*relName*

Integration name of the relationship.

*ObjName*

Integration name of the object.

*objId*

Object ID from one side of the relationship.

## Return Value

An array of related IDs

## Example

```
var arr = rbv_api.getRelatedIds2("R1321", "ex_group", "{!#UID}");
rbv_api.printArr(arr);
```

# rbv\_api.getRelatedFields()

## Purpose

For native Rollbase objects, returns an array of field values from related records for a given relationship and Object ID.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [rbv\\_api.getRelatedFields2\(\)](#) on page 574.

---

## Syntax

```
rbv_api.getRelatedFields(relName, id, fieldName)
```

## Parameters

*relName*

Integration name of relationship

*id*

Object ID from one side of the relationship

*fieldName*

Integration name for field in related object

## Return Value

Array of field values from related records or NULL if no related records exist.

## Example

```
var values = rbv_api.getRelatedFields("R1321", {!id}, "status");
for (var k=0; k<values.length; k++) {
    var status = values[k];
    rbv_api.println("status="+status);
}
```

## rbv\_api.getRelatedFields2()

### Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), returns an array of field values from related records for a given relationship, and Object ID.

---

**Note:** For native Rollbase objects, you can use the method, [rbv\\_api.getRelatedFields\(\)](#) on page 573.

---

### Syntax

```
rbv_api.getRelatedFields2(relName, objName, objId, fieldName)
```

### Parameters

*relName*

Integration name of relationship.

*objName*

Integration name of Object.

*objId*

Object ID from one side of the relationship.

*fieldName*

Integration name for field in related object.

### Return Value

Array of field values from related records or NULL if no related records exist.

### Example

```
var values = rbv_api.getRelatedFields2("R1321", "ex_group", "{!#UID}",  
"status");  
rbv_api.printArr(arr);
```

## Object Script API

The methods described in this section are for use in [Object Script triggers](#) and support a range of programmatic data manipulation. Use of Object Script methods requires advanced skills.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Object Script Example

In this example, we need to create values for a **Document Number** field, based on sequential IDs where the leading letters represent a particular document type. The types must be mixed without breaking the sequence. For example, as objects of different document types are created, the Document Number field might contain values such as the following:

- AAA0001
- AAA0002
- BBBG001
- AAA003

The Auto-Number field cannot be used because more than one sequence is involved. Instead, to fulfill this requirement, try the following solution:

- Create a picklist field, `doc_prefix`, with values AAA, BBB etc.
- Create an integer field, `seq_num`, but do not add this field to any page or view.
- Use the following Object Script methods in an **After Create** trigger to populate the **Document Number** field (read-only on pages):

```
// Find next sequential number
var nextNum = rbv_api.selectNumber("select MAX(seq_num)
    from documents where doc_prefix='{!!doc_prefix#value}''") + 1;

// Format document number
var nextStr = nextNum.toString(); while (nextStr.length < 4)
    nextStr = '0'+nextStr;
var docNumber = '{!!doc_prefix#value}'+nextStr;

// Record document number and sequential number
rbv_api.setFieldValue("documents", {!id}, "name", docNumber);
rbv_api.setFieldValue("documents", {!id}, "seq_num", nextNum);
```

## rbv\_api.getFieldValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This object script method returns the value of a field from an existing object record. You can access fields using template tokens instead of this method for better performance.

To get integration codes for enumerated values such as picklists and status, append the `#code` suffix to the `fieldName` parameter. For example, a `fieldName` value of `status` returns a status ID, `status#code` returns the status code, `status#value` returns the status display name.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

---

**Note:** Because `getFieldValue()` acts on existing data, it cannot be used in a validation script, which checks data before storing it.

---

## Syntax

```
rbv_api.getFieldValue(objName, objId, fieldName)
```

## Parameters

`objName`

Integration name of object

`objId`

ID of object record

`fieldName`

Integration name of field to retrieve value from

## Return Value

Value of requested field

## Example

For regular Rollbase objects:

```
var x = rbv_api.getFieldValue("order", {!id}, "description");
```

For objects imported from an external database or OpenEdge Service:

```
var x = rbv_api.getFieldValue("order", '{!#UID}', "description");
```

## rbv\_api.getNumFieldValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This Object Script method returns the numeric value of a field from an existing object record. This method is similar to `getFieldValue()`, but ensures that the returned value is a number or 0 if the field has a NULL value.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

```
rbv_api.getNumFieldValue(objName, objId, fieldName)
```

## Parameters

Parameter	Description
objName	Integration name of object
objId	ID of object record
fieldName	Integration name of field

## Return Value

Value of requested field

## rbv\_api.getBinaryData()

### Purpose

This Object Script method returns uploaded file as a base-64 encoded string. This method only works with native Rollbase objects.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.getBinaryData(objName, objId, fieldName)
```

## Parameters

Parameter	Description
objName	Integration name of object
objId	ID of object record
fieldName	Integration name of file upload field to retrieve value from

## Return Value

A base-64 encoded string or null

## rbv\_api.getTextData()

### Purpose

This Object Script method returns uploaded file as a plain-text string. Use only with plain-text uploaded files, such as HTML or CSV. This method only works with native Rollbase objects.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

**Syntax**`rbv_api.getTextData(objName, objId, fieldName)`**Parameters**

Parameter	Description
objName	Integration name of object
objId	ID of object record
fieldName	Integration name of File Upload field to retrieve value from

**Return Value**A plain-text string or `Null`**rbv\_api.isFieldEmpty()**

**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

**Purpose**

This Object Script method checks whether value of particular field is null or empty without bringing the actual value into the formula.

**Note:** Because `isFieldEmpty()` acts on existing data, it cannot be used in a validation script, which checks data before storing it.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

**Syntax**`rbv_api.isFieldEmpty(objName, objId, fieldName)`**Parameters**

objName

Integration name of object

objId

ID of object record

fieldName

Integration name of file upload field to retrieve value from

## Return Value

`true` if value of a field from an existing object record is null or empty, `false` otherwise

## Example

```
if (rbv_api.isEmpty("order", {!id}, "description") {  
    do something  
}
```

## rbv\_api.setFieldValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This Object Script method sets the value of a field for an existing object record. For picklists and status fields, use integration names, since they will be valid after publishing and installing your application, unlike numeric ids.

When called outside of a transaction scope, for instance, in a JavaScript report, the `setFieldValue()` method changes the field value temporarily. The `setFieldValue()` method does not invoke triggers before or after the update occurs. The reason is that this method can often be used repeatedly to update several fields and invoking triggers on each update may result in undesired behavior.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

```
rbv_api.setFieldValue(objName, objId, fieldName, newValue)
```

## Parameters

`objName`

Integration name of object

`objId`

ID of object record

`fieldName`

Integration name of field to set value in

`newValue`

Value to be set

## Return Value

None

## Example

The following code sets a picklist (single-selection) or radio-buttons value:

```
rbv_api.setFieldValue("order", {!id}, "size", "S");
```

The following sets a picklist (multi-selection) or group of checkboxes value:

```
rbv_api.setFieldValue("order", {!id}, "size",  
    Array("S", "M"));
```

An alternative use is with a comma-separated string of integration codes or numeric ids:

```
rbv_api.setFieldValue("order", {!id}, "size", "S,M");
```

## rbv\_api.setBinaryFieldValue()

### Purpose

This Object Script method sets the value of a file upload field using a base-64 encoded string. This method only works with native Rollbase objects.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.setBinaryFieldValue(objName, objId, fieldName, encodedValue,  
contentType, fileName)
```

### Parameters

objName

Integration name of object

objId

ID of object record to modify

fieldName

Integration name of the field to update

encodedValue

Base-64 encoded content of value to be set

contentType

MIME content type

fileName

Name of file to be stored

### Return Value

None

## rbv\_api.setTextFieldValue()

### Purpose

This Object Script method sets the value of a text field to the name of a file to be stored. This method only works with native Rollbase objects.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.setTextFieldValue(objName, objId, fieldName, textValue,  
contentType, fileName)
```

### Parameters

objName

Integration name of object

objId

ID of object record to modify

fieldName

Integration name of the field to update

textValue

Plain text content of value to be set

contentType

MIME content type

fileName

Name of file to be stored

### Return Value

None

## Example

```
var xmlBody = "<test>TEST</test>"  
rbv_api.setTextFieldValue("invoice", {!id}, "xmlField", xmlBody,  
"text/xml", "invoice.xml");
```

## rbv\_api.createRecord()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

This Object Script method creates a new object record from the values passed in. The Object Script API invokes triggers on create, on update and on delete the same way as the Rollbase user interface does. The ID of a newly created record is available for other triggers in update chain through the shared value, `newID_objectName`, that can be retrieved using `getSharedValue()`.

**Note:** This methods observes required and unique field properties. An attempt to create or update a record without setting required fields or by violating uniqueness of field values will cause an error.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

```
rbv_api.createRecord(objName, arr)
```

## Parameters

objName

Integration name of the type of object to create

arr

JavaScript array of field name/value pairs (required fields must be supplied)

## Return Value

Object ID of the newly created record

## Example

The following creates a related record:

```
var x = new Array();  
x["amount"]=1000;  
x["R477842"]={!id};  
x["name"]="API Created";  
var newId = rbv_api.createRecord("item", x);  
rbv_api.print("Created record with id: "+newId);
```

The following creates a Communication Log record:

```
var a = new Array();
a["name"] = "Next Action Changed";
a["body"] = "Test body string";
a["commParent"] = {!id};
rbv_api.createRecord("COMMLOG", a);
```

## rbv\_api.updateRecord()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This Object Script method updates a record. Only supply values for the fields you want to change, others fields will remain unchanged. The Object Script API invokes triggers on create, on update and on delete the same way as the Rollbase user interface does. The ID of a newly created record is available for other triggers in update chain through the shared value, `newID_objectName`, that can be retrieved using [getSharedValue\(\)](#).

---

**Note:** This methods observes required and unique field properties. An attempt to create or update a record without setting required fields or by violating uniqueness of field values will cause an error.

---

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.updateRecord(objName, objId, arr)
```

### Parameters

`objName`

Integration name of the type of object to create

`arr`

JavaScript array of field name/value pairs (required fields must be supplied)

### Return Value

None

### Example

The following example loops through related records and updates them:

```
var x;
{!#LOOP_BEGIN.R477842}
x = new Array();
x["amount"]={!R477842.amount}+100;
```

```
rbv_api.updateRecord("item", {!R477842.id}, x);
{!#LOOP_END.R477842}
```

## rbv\_api.attach()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This Object Script method creates a relationship between two records and returns `true` if a new relationship has been successfully created.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.attach(relName, objName1, objId1, objName2, objId2)
```

### Parameters

*relName*

Integration name of the relationship

*objName1*

Integraton name of the first object

*objId1*

ID of the first Object record

*objName2*

Integration name of the second object

*objId2*

ID of the second object record

### Return Value

`true`, if association succeeded

## rbv\_api.detach()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This Object Script method removes a relationship between two records.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

```
rbv_api.detach(relName, objName1, objId1, objName2, objId2)
```

## Parameters

*relName*

Integration name of the relationship

*objName1*

Integraton name of the first object

*objId1*

ID of the first Object record

*objName2*

Integration name of the second object

*objId2*

ID of the second object record

## Return Value

None

## rbv\_api.runTrigger()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This Object Script method runs a trigger on the specified object. This method will invoke a trigger regardless of timing options, such as On After Update. This API requires EDIT permission on the object record. For HTTP calls, consider using [sendHttpGet\(\)](#) or [setHttpPost\(\)](#) instead.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

```
rbv_api.runTrigger(objName, objId, triggerOrigId, checkValidation)
```

## Parameters

objName

Integration name of the object

objId

ID of the record on which to run the trigger

triggerOrigId

Integration name (string) or original ID (integer) of the trigger

checkValidation

true or false. If set to true, check validation formula before running trigger, otherwise ignore validation formula.

## Return Value

None

## **rbv\_api.deleteRecord()**



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

This Object Script method deletes the specified record.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

## Syntax

`rbv_api.deleteRecord(objName, objId)`

## Parameters

objName

Integration name of object

objId

ID of record to delete

## Return Value

None

## Example

The following example deletes a record:

```
rbv_api.deleteRecord("item", {!R477842.id});
```

## rbv\_api.setCreator()

### Purpose

This Object Script trigger method sets the **Created By** field value for the selected record. The current user must have a **User Role** of **Administrator** to successfully use this method. This method only works with native Rollbase objects.

Formulas used in [Object Script triggers](#) can include other API methods and regular Rollbase template functionality, such as loops. Object Script methods have no effect if called outside of update triggers.

### Syntax

```
rbv_api.setCreator(objName, objId, creatorName, creatorId)
```

### Parameters

*objName*

Integration name of object

*objId*

ID of record to delete

*creatorName*

Integration name of the user or portal user to be set as creator

*creatorId*

ID of the creator record

### Return Value

None

## Example

```
rbv_api.setCreator("item", {!id}, "USER", 123456);
```

## User Selection API

The methods in this section give the caller access to information about users. This includes whether an application is installed for a particular user, and who might have viewed or flagged particular records.

## rbv\_api.isViewed()

### Purpose

Accesses the `viewed` attribute of a record for a particular user. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.isViewed(objName, objId, userId)
```

### Parameters

*objName*

Integration name of object

*objId*

ID of the object record

*userId*

*ID of the user who might have viewed the record*

### Return Value

`true` if a record was viewed

## rbv\_api.setViewed()

### Purpose

Sets the `viewed` attribute for a specified record and user. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.setViewed(objName, objId, userId, isViewed)
```

### Parameters

*objName*

Integration name of object

*objId*

ID of object record

*userId*

*ID of the user who might have viewed the record*

*isViewed*  
true or false

### Return Value

None

## rbv\_api.isFlagged()

### Purpose

Determines whether a particular user flagged a record. Tip: Use the token `{!#CURR_USER.id}` to retrieve the ID of the currently logged-in user. This method only works with native Rollbase objects.

### Syntax

`rbv_api.isFlagged(objName, objId, userId)`

### Parameters

*objName*

Integration name of object

*objId*

ID of object record

*userId*

ID of the user who might have flagged the record

### Return Value

true if a record was flagged

## rbv\_api.setFlagged()

### Purpose

Sets the flagged attribute for a specified record and user. This method only works with native Rollbase objects.

### Syntax

`rbv_api.setFlagged(objName, objId, userId, isFlagged)`

### Parameters

*objName*

Integration name of object

*objId*

ID of object record

*userId*

ID of the user who might have flagged the record

*isFlagged*

true or false

## Return Value

None

## rbv\_api.getSelectedIds()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Gets an array of record ids selected by current user using check boxes in list view.

## Syntax

`rbv_api.getSelectedIds(objName)`

## Parameters

*objName*

Integration name of object

## Return Value

Array of record IDs

## Example

```
var arr = rbv_api.getSelectedIds("order");
var buff = '';
if (arr!=null) {
  for (var k=0; k<arr.length; k++)
    buff += arr[k]+', ';
}
return buff;
```

## rbv\_api.isAppInstalled()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns `true` if application with the given ID is installed for the current customer, `false` otherwise.

### Syntax

```
rbv_api.isAppInstalled(appId)
```

### Parameters

*appId*

Original ID of application

### Return Value

`true` or `false`

## Miscellaneous Methods

The methods in this section can be used in formulas.

## rbv\_api.getIdByCode()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the ID of a status or a picklist item with the given integration code.

### Syntax

```
rbv_api.getIdByCode (objName, fieldName, code)
```

### Parameters

*objName*

Integration name for object

*fieldName*

Integration name of workflow status or picklist

code

Integration name of status or picklist item

### Return Value

ID of item or -1

### Example

```
var itemId = rbv_api.getIdByCode("order", "type", "S");
```

## rbv\_api.getCodeById()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Returns the code of a status or a picklist item with the given ID.

### Syntax

```
rbv_api.getCodeById(objName, fieldName, id)
```

### Parameters

objName

Integration name of object

fieldName

Integration name of workflow status or picklist

### Return Value

Code of item or null

## rbv\_api.getValueById()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Returns the value of a status or a picklist item with the given ID.

### Syntax

```
rbv_api.getValueById(objName, fieldName, id)
```

## Parameters

objName

Integration name for object

fieldName

Integration name of workflow status or picklist

id

ID of status or picklist item

## Return Value

Text value of item or null

## Example

The following returns the value of an order type.

```
var itemValue = rbv_api.getValueById("order", "type", {!type#id});
```

# rbv\_api.runReport()

## Purpose

Runs a report and returns it as a base-64 encoded string. This method only works with native Rollbase objects.

## Syntax

```
rbv_api.runReport(reportId, filterName, filterValue)
```

## Parameters

reportId

The report ID.

filterName

An optional field that specifies the name of the field to filter.

filterValue

Specifies a value for the field. Only records with this value will be added to the report.

## Return Value

Base-64 encoded string.

## Example

The following example reads the order report and stores the results in the file upload field:

```
var reportData = rbv_api.runReport(23086, null, null);
rbv_api.setBinaryFieldValue("report", {!id}, "file",
reportData, "application/pdf", "report.pdf");
```

## rbv\_api.runTemplate()

### Purpose

Runs a document template on a specified object record and returns it as a base-64 encoded string. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.runTemplate(objName, ObjId, templateId)
```

### Parameters

*objName*

The object Integration name.

*objId*

ID of object record.

*templateId*

The document template ID.

### Return Value

The template as a base-46 encoded String.

## Date, Time, and Currency API

The methods in this section allow you to get and format date, number, and currency values.

## rbv\_api.getCurrentDate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns a string corresponding to the current date, relative to the current user's time zone. Use `new Date()` to create a new instance of a JavaScript Date.

**Syntax**

```
rbv_api.getCurrentDate()
```

**Return Value**

Current date as a string

## rbv\_api.firstDayOfMonth()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

**Purpose**

Returns a value of 12AM on the first day of the current month plus or minus the number of months specified.

**Syntax**

```
rbv_api.firstDayOfMonth(offset)
```

**Parameters**

*offset*

Number of months to offset the current month (positive, negative, or zero)

**Return Value**

Date as a string.

**Example**

```
var d = new Date(rbv_api.getCurrentDate());
rbv_api.println(d);
var d1 = new Date(rbv_api.firstDayOfMonth(0));
rbv_api.println(d1);
var d2 = new Date(rbv_api.firstDayOfMonth(-1));
rbv_api.println(d2);
```

Output:

```
Sat Feb 26 2012 13:00:02 GMT-0800 (PST)
Tue Feb 01 2012 00:00:00 GMT-0800 (PST)
Sat Jan 01 2012 00:00:00 GMT-0800 (PST)
```

## rbv\_api.firstDayOfQuarter()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns a value of 12AM on the first day of the current quarter plus or minus the number of quarters specified.

## Syntax

```
rbv_api.firstDayOfQuarter(offset)
```

## Parameters

*offset*

Number of quarters to offset the current quarter (positive, negative, or zero)

## Return Value

Date as a string.

## rbv\_api.firstDayOfYear()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns a value of 12AM on the first day of the current year, plus or minus the number of years specified.

## Syntax

```
rbv_api.firstDayOfYear(offset)
```

## Parameters

*offset*

Number of years to offset the current year (positive, negative, or zero)

## Return Value

Date as a string

## Example

The following example calculates first day of the current calendar year and uses Query API to sum invoice records after that date:

```
var d = new Date(rbv_api.firstDayOfYear(0));
var sum = rbv_api.selectNumber("SELECT SUM(amount) FROM invoice WHERE
due_date>=?", d);
```

## rbv\_api.formatDate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Formats a JavaScript Date class instance as specified in the `pattern` parameter.

### Syntax

```
rbv_api.formatDate(date, pattern)
```

### Parameters

*date*

instance of JavaScript Date class

*pattern*

A pattern specified using the conventions defined in the Java `SimpleDateFormat` class

### Return Value

Date formatted as a string

### Example

```
var d = rbv_api.selectValue("select updatedAt from orders where id=?", {!id});
var x = rbv_api.formatDate(d, "yyyy-MM-dd");
rbv_api.println("x="+x);

Output:
x=2012-03-12
```

## rbv\_api.formatNumber()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Formats a decimal number to a string according to the specified parameters.

### Syntax

```
rbv_api.formatNumber(number, decPlaces, decSeparator, thSeparator)
```

## Parameters

*number*

Number to be formatted

*decPlaces*

Number of decimal places to be used

*decSeparator*

Separator to use for decimal places

*thSeparator*

Character separator to use for thousands

## Return Value

Number as a formatted string.

## Example

```
var x = rbv_api.formatNumber(7000123.45, 2, ",", ".") ;  
rbv_api.println("x="+x) ;
```

Output:  
x=7.000.123,45

## rbv\_api.formatCurrency()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Formats a currency number as a string according to the specified parameters.

## Syntax

```
rbv_api.formatCurrency(number, currSymbol, decPlaces, decSeparator,  
thSeparator)
```

## Parameters

*number*

Number to be formatted

*currSymbol*

Currency symbol to use in formatted string

*decPlaces*

Number of decimal places to be used

*decSeparator*

Separator to use for decimal places

*thSeparator*

Character separator to use for thousands

## Return Value

Number as a formatted string.

## Example

Example:

```
var x = rbv_api.formatCurrency(7000123.45, "$ ", 2, ".", " ");  
rbv_api.println("x="+x);
```

Output:

```
x=$ 7 000 123.45
```

## rbv\_api.getExchangeRate()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns an exchange rate between two currencies on a given date.

## Syntax

```
rbv_api.getExchangeRate(srcCode, destCode, date)
```

## Parameters

*srcCode*

Currency code to convert from

*destCode*

Currency code to convert to

*date*

Date for exchange rate (Defaults to current date)

## Return Value

Rate as a decimal number

## Example

```
var rate = rbv_api.getExchangeRate("EUR", "USD", new Date());
```

# PDF Processing API

The methods in this section simplify processing of PDF documents on the Rollbase platform. For Private Cloud installations, these methods are only available if PDF Kit software is installed.

## rbv\_api.getPDFProperty()

### Purpose

Returns the specified property of the PDF document specified as a base-64 encoded string. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.getPDFProperty(encodedValue, property)
```

### Parameters

*encodedValue*

base-64 encoded string representing PDF document

*Property*

One of the following: TITLE, AUTHOR, CREATIONDATE, CREATOR, KEYWORDS, MODDATE, ENCRYPTED, ERROR

### Return Value

Property of PDF document as a string

## Example

```
var str = rbv_api.getBinaryData("document", {!id}, "file");
var author = rbv_api.getPDFProperty(str, 'AUTHOR');
var encrypted = rbv_api.getPDFProperty(str, 'ENCRYPTED');
```

## rbv\_api.concatPDF()

### Purpose

Concatenates two or more PDF documents specified as base-64 encoded strings. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.concatPDF(encodedValue1, encodedValue2, ...)
```

## Parameters

*encodedValue1*

base-64 encoded string representing 1st PDF document

*encodedValue2, ...*

base-64 encoded string representing PDF document(s) to concatenate

## Return Value

Merged PDF document as a base-64 encoded string

## Example

```
var pdf1 = rbv_api.getBinaryData("document", {!id}, "pdf_file1");
var pdf2 = rbv_api.getBinaryData("document", {!id}, "pdf_file2");
var pdf3 = rbv_api.concatPDF(pdf1, pdf2);
rbv_api.setBinaryFieldValue("document", {!id}, "pdf_file3", pdf3,
    "application/pdf", "merged.pdf");
```

# Hosted File API

Rollbase hosted files include files such as images, movies, and stylesheets for a portal. The methods in this section allow you to interact with hosted files.

## rbv\_api.getHostedAsText()

### Purpose

Returns the content of a hosted file as a string. This method only works for text-based files such as those with extensions of .xml, .html, and .txt. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.getHostedAsText(origId)
```

## Parameters

*origId*

Original ID of hosted file

## Return Value

Content of hosted file as a string or NULL if file doesn't exist

## Example

### **rbv\_api.getHostedAsBinary()**

#### **Purpose**

Returns the content of a hosted file as a base-64 encoded string. This method only works with native Rollbase objects.

#### **Syntax**

```
rbv_api.getHostedAsBinary(origId)
```

#### **Parameters**

*origId*

Original ID of hosted file

#### **Return Value**

The content of a hosted file as a base-64 encoded string or NULL (if file does not exist).

## HTTP API

The functions in this section can be used to send HTTP requests from server-side logic.

### **rbv\_api.sendHttpGet()**



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

#### **Purpose**

Sends HTTP GET requests to the specified URL and returns the HTTP response body.

#### **Syntax**

```
rbv_api.sendHttpGet(url)
```

#### **Parameters**

*url*

URL for a GET request

#### **Return Value**

HTTP response

## Example

```
var info = rbv_api.sendHttpGet(  
    "http://www.rollbase.com/master/system/ri.jsp");  
rbv_api.println(info);  
  
Output:  
4.08|2013-06-01T
```

## rbv\_api.sendHttpPost()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Sends multipart HTTP POST requests to the specified URL and returns body of HTTP response.

## Syntax

```
rbv_api.sendHttpPost(url, params, headers, chunked)
```

## Parameters

*url*

URL containing POST request

*params*

Array of POST parameters in name/value pairs

*headers*

Optional array of HTTP headers in name value pairs

*chunked*

If set to `true` or omitted, HTTP POST "chunked" parameter is used

## Return Value

Body of HTTP response

## Example

The following example logs in to a remote server:

```
var url = "http://my_server/login";  
var params = {"loginName":"my_name", "password":"my_password"};  
var resp = rbv_api.sendHttpPost(url, params, null);  
rbv_api.println(resp);
```

## rbv\_api.sendJSONRequest()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Sends a JSON request to the specified URL and returns the HTTP response body.

### Syntax

```
rbv_api.sendJSONRequest(url, data, method, contentType, username, password, headers)
```

### Parameters

*url*

URL to send an HTTP request

*data*

Text string or a JSON object to send as a body of the request

*method*

HTTP method, one of GET (default), POST, PUT or DELETE

*contentType*

MIME content type of data. For example:

application/json; charset=UTF-8 (default)

*username*

Login name for BASIC authentication, by default null

*password*

Password for BASIC authentication, by default null

*headers*

Optional array of HTTP headers in name/value pairs, by default null

### Return Value

Body of HTTP response

## rbv\_api.getHTTPParameter()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This method returns the value of an HTTP parameter sent to the server during a UI-based update operation from an object field or a custom HTML component.

### Syntax

```
rbv_api.getHTTPParameter(name)
```

### Parameters

*name*

Name of input HTML box of a UI form HTML element

### Return Value

HTTP parameter or null

## XML Processing API

The methods in this section allow you to parse XML strings. For instance, a call to an external server might return an XML string.

## rbv\_api.parseXML()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Parses an XML string and returns an instance of `org.w3c.dom.Element` corresponding to the XML root. JavaScript can use all methods of this Java object. See <http://www.w3.org/2003/01/dom2-javadoc/org/w3c/dom/Element.html> for more info.

### Syntax

```
rbv_api.parseXML(xmlString)
```

### Parameters

*xmlString*

An XML document as a string

## Return Value

Root element of an XML document

## Example

```
var xml = "<a><b>BBB</b><c>CCC</c></a>";
var root = rbv_api.parseXML(xml);
var nodes = root.getChildNodes();
for (var k=0; k<nodes.length; k++) {
    var child = nodes.item(k);
    rbv_api.println("Tag="+child.getTagName());
    var nodes2 = child.getChildNodes();
    for (var n=0; n<nodes2.length; n++) {
        var child2 = nodes2.item(n);
        var x = child2.getNodeValue();
        rbv_api.println("Value="+x);
    }
}
Output:
Tag=b
Value=BBB
Tag=c
Value=CCC
```

## rbv\_api.evalXpath()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Parses an XML string and evaluates an XPath expression. See <http://www.w3schools.com/xpath/> for more information on XPath.

## Syntax

`rbv_api.evalXpath(xmlString, xpathExpr)`

## Parameters

`xmlString`

XML document as a string

`xpathExpr`

XPath expression to evaluate

## Return Value

Result of XPath evaluation

## Example

The following example evaluates a three-node XPath.

```
var xmlString =
    "<a><b>node one</b><c>node two</c><b>node three</b></a>";
var xpathExpr = "/a/b/text()";
var res = rbv_api.evalXPath(xmlString, xpathExpr);
for (var k=0; k<res.length; k++)
    rbv_api.println("Result: "+res.item(k));

Output:
Result: [#text: node one]
Result: [#text: node three]
```

# JSON Processing API

The methods in this section allow you to parse strings into JSON objects or to convert JSON objects into strings.

## rbv\_api.jsonToString()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Converts a JSON object into a text string.

### Syntax

```
rbv_api.jsonToString(json)
```

### Parameters

*json*

A JSON object

### Return Value

A text string

## rbv\_api.stringToJson()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This method parses a text string into a JSON object

**Syntax**

```
rbv_api.stringToJson(str)
```

**Parameters**

*str*

A string value

**Return Value**

A text string

**Example**

```
var x = {"a": "AAAA", "b": 100};
var y = rbv_api.jsonToString(x);
rbv_api.println("y=" + y);

var z = rbv_api.stringToJson(y);
rbv_api.println("z.a=" + z.a);
rbv_api.println("z.b=" + z.b);

Output:
y={"b": 100, "a": "AAAA"}
z.a=AAAA
z.b=100
```

## Trigger API

The methods in this section are only for use with triggers. They allow you to share information among triggers. For example, the `setSharedValue()` and `getSharedValue()` allow storing results of calculations from one trigger and making them available for reuse in another trigger(s).

The remaining methods in this section can be used to determine:

- The environment in which your trigger is running: UI, Portal, API, or Delayed.
- What operation is currently being performed: create, update, delete, or something else.

Data can be stored only using these trigger methods while the current transaction to create, update, or delete a record is running. If you need another type of sharing, or application-level sharing rather than record-level, consider using company-wide settings, see [Using Company-wide Settings](#) on page 424.

### rbv\_api.setSharedValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

**Purpose**

This method must be called within a trigger. It stores a value and makes it available for subsequent triggers through the `getSharedValue()` method.

Data can be stored only using these trigger methods while the current transaction to create, update, or delete a record is running. If you need another type of sharing, or application-level sharing rather than record-level, consider using company-wide settings, see [Using Company-wide Settings](#) on page 424.

## Syntax

```
rbv_api.setSharedValue(name, value)
```

## Parameters

*name*

Symbolic name for the stored value

*value*

stored value

## Return Value

None

## rbv\_api.getSharedValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This method must be called within a trigger. It returns a value stored by the [setSharedValue\(\)](#) method.

Data can be stored only using these trigger methods while the current transaction to create, update, or delete a record is running. If you need another type of sharing, or application-level sharing rather than record-level, consider using company-wide settings, see [Using Company-wide Settings](#) on page 424.

## Syntax

```
rbv_api.getSharedValue(name)
```

## Parameters

*name*

Symbolic name for the stored value

## Return Value

Shared value or null

## Example

Trigger 1:

```
var x = rbv_api.selectNumber(...);
// Store queried value for subsequent re-use
rbv_api.setSharedValue("x", x);
```

Trigger 2:

```
// Retrieve previously stored value
var x = rbv_api.getSharedValue("x");
```

## rbv\_api.isUI()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Returns `true` if the trigger was called or a formula containing this method was invoked from a regular UI page.

**Note:** This method returns `false` if called outside of a trigger

## Syntax

```
rbv_api.isUI()
```

## Parameters

*None*

N/A

## Return Value

`true` or `false`

## Example

```
if (rbv_api.isUI()) rbv_api.print("Called from UI page");
```

## rbv\_api.isPortal()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Returns `true` if a trigger is called or a formula is invoked from a portal UI page.

---

**Note:** This method returns `false` if called outside of a trigger

---

## Syntax

```
rbv_api.isPortal()
```

## Parameters

*None*

N/A

## Return Value

`true` or `false`

## Example

To receive notification when a new lead record is created from portal page, but not from UI page, create an [Email notification trigger](#) and set the condition formula to (no need to add return keyword):

```
rbv_api.isPortal()
```

## rbv\_api.isMobile()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns `true` if a trigger is called from a Rollbase Mobile-Web page.

---

**Note:** This method returns `false` if called outside of a trigger

---

## Syntax

```
rbv_api.isMobile()
```

## Parameters

*None*

N/A

## Return Value

`true` or `false`

## rbv\_api.isAPI()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns `true` if a trigger is called from a regular Web Services API or REST.

---

**Note:** This method returns `false` if called outside of a trigger

---

### Syntax

`rbv_api.isAPI()`

### Parameters

*None*

N/A

### Return Value

`true` or `false`

## rbv\_api.isDelayed()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns `true` if this trigger is delayed and running when no user is logged in.

---

**Note:** This method returns `false` if called outside of a trigger

---

### Syntax

`rbv_api.isDelayed()`

### Parameters

*None*

N/A

### Return Value

`true` or `false`

## rbv\_api.isImport()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns `true` if this trigger is running while importing records from spreadsheet.

---

**Note:** This method returns `false` if called outside of a trigger

---

### Syntax

```
rbv_api.isImport()
```

### Parameters

*None*

N/A

### Return Value

`true` or `false`

## rbv\_api.isCreate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns `true` if this trigger is called while creating a new record.

---

**Note:** This method returns `false` if called outside of a trigger

---

### Syntax

```
rbv_api.isCreate()
```

### Parameters

*None*

N/A

### Return Value

`true` or `false`

## Example

```
if (rbv_api.isCreate()) rbv_api.print("Creating a new record");
```

## rbv\_api.isUpdate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Returns `true` if trigger is called while updating an existing record.

**Note:** This method returns `false` if called outside of a trigger

### Syntax

```
rbv_api.isUpdate()
```

### Parameters

*None*

N/A

### Return Value

`true` or `false`

## Example

```
if (rbv_api.isCreate()) rbv_api.print("Creating a new record");
```

## rbv\_api.isDelete()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Returns `true` if trigger is called while deleting an existing record.

**Note:** This method returns `false` if called outside of a trigger

### Syntax

```
rbv_api.isDelete()
```

## Parameters

*None*

N/A

## Return Value

`true` or `false`

## Example

```
if (rbv_api.isCreate()) rbv_api.print("Creating a new record");
```

# Debugging API

The methods in this section apply to debugging of server-side logic.

## `rbv_api.print()`



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Prints text messages in the formula and trigger debugging windows. End-users will not see the messages at runtime.

### Syntax

```
rbv_api.print (text)
```

### Parameters

*text*

Message to print in the debugging window

## Return Value

*None*

## Example

**Original Formula**

```
var x = '{!id}';  
rbv_api.print("Hello world id="+x);  
return 100;
```

**Parsed Formula**

```
var rbv_api = new Packages.com.rb.core.services.  
  
function wrapper() {  
var x = '480924';  
rbv_api.print("Hello world id="+x);  
return 100;  
}  
wrapper();
```

Formula return type: **Decimal**

**Result**

```
Hello world id=480924  
100
```

## rbv\_api.println()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Prints text messages in the formula debugging window, adding an end-of-line "\n" symbol after the output. End-users will not see the messages at runtime.

The `print()` and `println()` methods can also be used in [JavaScript reports](#) to generate output.

### Syntax

```
rbv_api.println (text)
```

### Parameters

*text*

message to be output

## Return Value

None

## rbv\_api.printARR()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Prints the contents of a JavaScript array in the format key => value.

## Syntax

```
rbv_api.printArr(arr)
```

## Parameters

*arr*

JavaScript array to print

## Return Value

None

## rbv\_api.setVerbose()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Sets the verbose flag for debugging formulas and triggers. Methods such as `selectQuery()`, `createRecord()`, `updateRecord()`, will generate additional debugging output if this flag is set to `true`.

## Syntax

```
rbv_api.setVerbose (flag)
```

## Parameters

*flag*

`true` or `false`

## Return Value

None

## rbv\_api.isVerbose()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the verbose flag that can be set using the `setVerbose()` method.

### Syntax

```
rbv_api.isVerbose()
```

### Parameters

NA

Not applicable

### Return Value

`true` or `false`

## rbv\_api.inArgs()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the position (0-based index) of the first argument in a variable length list of remaining arguments.

### Syntax

```
rbv_api.inArgs(x, arg0, arg1, ...)
```

### Parameters

`x`

variable

`args`

One or more arguments

### Return Value

`true` or `false`

## Example

```
rbv_api.inArgs("Z", "X", "Y", "Z") returns 2
rbv_api.inArgs("A", "X", "Y", "Z", "1") returns -1 (i.e. not found)
```

# Log API

The methods discussed in this section enables you to log information in a log file.

## rbv\_api.createActivityLog()

### Purpose

Creates an activity log record on an existing object record. This method only works with native Rollbase objects.

### Syntax

```
rbv_api.createActivityLog (objName, objId, text)
```

### Parameters

objName

Integraton name of object

objId

ID of record

text

Text of Activity Log record

### Return Value

None

## rbv\_api.log()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Creates an log entry in the specified system log file.

**Note:** The `log()` method can be used for debugging delayed triggers, SOAP calls, and REST calls where UI-based debugging is unavailable.

---

**Syntax**

```
rbv_api.Log (logName, text)
```

**Parameters**

logName

The name of one of the system log files—webapi, rest, jobs, debug—where your log entry will be recorded.

text

The text entry to be logged.

**Return Value**

None

**Example**

```
var x = rbv_api.selectValue(...);  
rbv_api.log("debug", "x="+x);
```

## Email API

This section describes APIs used to read email messages from external email servers using IMAP or POP3 protocols. Only secure (SSL) connection is supported.

### **rbv\_api.openIMAP()**

**Purpose**

Opens connection to an external email server (such as Gmail) using IMAP protocol. It must be called once before reading data from IMAP server.

**Syntax**

```
rbv_api.openIMAP(hostName, port, userName, password, mailProps)
```

**Parameters**

hostName

The name of the external mail server.

port

The port number for an SSL connection on the external mail server (typically 993).

userName

The user's name (typically user's email address) on the external mail server.

password

The user's password on the external mail server.

mailProps

The optional array of properties to be set on the external mail server. This may be null.

## Return Value

None

## Example

```
rbv_api.openIMAP("imap.gmail.com", 993, "address@gmail.com", password, null);
```

# rbv\_api.openPOP3()

## Purpose

Opens connection to an external email server (such as Microsoft Exchange) using POP3 protocol. It must be called once before reading data from POP3 server.

## Syntax

```
rbv_api.openPOP3(hostName, port, userName, password, mailProps)
```

## Parameters

hostName

The name of the external mail server.

port

The port number for an SSL connection on the external mail server (typically 995).

userName

The user's name (typically user's email address) on the external mail server.

password

The user's password on the external mail server.

mailProps

The optional array of properties to be set on the external mail server. This may be null.

## Return Value

None

## Example

```
rbv_api.openPOP3("mail.mydomain.com", 995, "address@mydomain.com", password,  
null);
```

## rbv\_api.getMailMessageCount()

### Purpose

Returns the number of messages in the INBOX folder for the logged in user.

### Syntax

```
rbv_api.getMailMessageCount()
```

### Return Value

Number of email messages.

## Example

```
rbv_api.openPOP3("mail.mydomain.com", 995, "address@mydomain.com", password,  
null);  
var x = rbv_api.getMailMessageCount();  
rbv_api.closeMailSession();
```

## rbv\_api.getNewMailMessageCount()

### Note:

This method is no longer available.

## rbv\_api.getMailMessage()

### Purpose

Returns the specified number of email messages from the INBOX folder of the logged in user.

## Syntax

```
rbv_api.getMailMessage (msgNo)
```

## Parameters

msgNo

The number of email messages.

## Return Value

Object representing message. The following properties are available:

- `to`: address “to”
- `cc`: address “CC”
- `from`: address “from”
- `subject`: subject of email message
- `body`: body of email message as HTML
- `text`: body of email message as plain text

## Example

```
rbv_api.openPOP3("mail.mydomain.com", 995, "address@mydomain.com", password, null);
var m = rbv_api.getMailMessage(100);
var x = m.get("subject");
rbv_api.closeMailSession();
```

# rbv\_api.getMailMessages()

## Purpose

Returns email messages based on specified start and end indexes (1-based) from the INBOX folder of the logged in user.

## Syntax

```
rbv_api.openIMAP (start, end)
```

## Parameters

start

The start index (1-based) of message in the INBOX folder.

end

The end index (1-based) of message in the INBOX folder.

## Return Value

Array of objects representing messages. See available properties in [rbv\\_api.getMailMessage\(\)](#) on page 622.

## Example

```
rbv_api.openPOP3("mail.mydomain.com", 995, "address@mydomain.com", password,  
null);  
var arr = rbv_api.getMailMessages(100, 110);  
for (var k=0; k<arr.length; k++) {  
    var m = arr[k];  
    rbv_api.println(m.get('subject'));  
}  
rbv_api.closeMailSession();
```

## **rbv\_api.closeMailSession()**

### **Purpose**

Closes connection to a connected external email server.

### **Syntax**

```
rbv_api.closeMailSession()
```

### **Return Value**

None

# **Client-side AJAX API**

Topics in this section describe APIs that can be used to modify the browser-side experience. Access Control permissions apply to client-side AJAX APIs.

The access permissions required to execute AJAX API are:

- The Current user (Portal User Role for Portals)
- The System Role "AJAX API"

If neither the current user nor the AJAX API role has appropriate permissions to access the API, the API call fails.

For more information about security and access permissions, see [Security and Access Control](#) on page 375.

## **Queries**

The Rollbase AJAX Query API allows developers to run SQL SELECT queries from application and portal pages using AJAX.

The process of performing a SELECT query using the Client-Side Query API is similar to the server-side API described (see [Server-side API](#) on page 559).

AJAX Query API functions typically require a callback function as a parameter. This is due to the asynchronous nature of AJAX communications. You should process results inside that callback function rather than try to use global JavaScript variables. For complex processing consider using other functions inside callback, possibly included in the hosted files. For information on hosted files, see [Hosted Files](#) on page 267.

Consider an example of invalid API usage:

```
var globalcount;
rbf_selectNumber("SELECT COUNT(1) FROM USER",
  function(count) { globalcount = count; } );
alert("Counter="+globalcount);
```

The following code will display "Counter=undefined" since the processing of query result starts before that result is fetched. However this example will work as expected:

```
rbf_selectNumber("SELECT COUNT(1) FROM USER",
  function(count) { alert("Counter="+count); } );
```

Several Query API return results as JSON arrays or single values. JSON return values include those shown in the following table.

Data Type	JSON Value
NULL (no value)	null
Number	Number
Boolean value	true or false
String	String
Relationship (lookup)	Array of related ids
Date or Date/time	Instance of Javascript Date class

## rbf\_createRecord()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

This method creates a new record without changing the UI presentation. If an error occurred during operation error notification procedure will be invoked.

The create operation on the selected object must be allowed for the current user or API User role, see [Built-in Security Levels](#) on page 377

### Syntax

```
rbf_createRecord(objName, fieldMap, useIds, callback)
```

### Parameters

*objName*

The integration name of the selected Object definition

*fieldMap*

An array of named parameters of the form: name = field name, value = field value)

*useIds*

If `true`, the API will accept numeric IDs; if `false` (default) the API will return integration codes for status fields and picklists

*callback*

A callback function that will receive array of fetched records as first parameter

## rbf\_deleteRecord()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This method deletes an existing backend record without changing UI presentation. If error occurred during operation error notification procedure will be invoked (see [rbf\\_setErrorsCallback](#)).

The delete operation on the selected record must be allowed for the current user or API User role, see [Built-in Security Levels](#) on page 377.

### Syntax

```
rbf_deleteRecord(objName, id, callback)
```

### Parameters

*objName*

The integration name of the selected Object definition

*id*

ID of the selected Object record

*callback*

A function to be executed when the operation is successful. It does not receive any parameters. This is an optional parameter.

## rbf\_getCount()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function retrieves total number of records in a view. It brings info back to the page using an asynchronous AJAX mechanism.

## Syntax

```
rbf_getCount(viewId, callback)
```

### Parameters

*viewID*

The original ID of View

*callback*

A callback function that will receive number of records as first parameter

## rbf\_getCount2()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

To filter, based on a filter condition, and retrieve records in the records view page.

## Syntax

```
rbf_getCount2(viewId, filterName, filterValue, callback)
```

### Parameters

*viewId*

The original ID of the view

*filterName*

Name of the field to filter output using the "equals" option (replaces the filter set in the View, if any).

*filtervalue*

Value of the field to filter output using the "equals" option.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

## Example

```
rbf_getCount2(18257, "starts with", "test", callback)
```

## rbf\_getFields()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

This function retrieves the values of specified fields for a selected Rollbase record. It brings info back to the page using asynchronous AJAX mechanism.

### Syntax

```
rbf_getFields(objName, id, fields, callback)
```

### Parameters

*objName*

The integration name of selected Object definition

*id*

ID of selected Object record

*fields*

A comma-separated list of field names for values that are requested

*callback*

A callback function that will receive parameters: *objName*, *id*, an array of received values, which can be accessed using field's integration name as a key (see example below).

## Example

The following example reads two fields from current record (presumably not displayed on the page) and makes them available for computations:

```
function my_callback(objName, objId, values) {
    var amount = values['amount'];
    var price = values['price'];
    // Do something...
}

rbf_getFields("order", id, "amount,price", my_callback);
```

## rbf\_getPage()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function fetches a set of records in a View, including (optionally) dependent records. It brings info back to the page using asynchronous AJAX mechanism. The amount of processing and time required to get complete results can vary widely, depending on whether it fetches related records and the number of rows you specify per page.

### Syntax

```
rbf_getPage(viewId, startRow, rowsPerPage, composite, objNames, fieldList, callback)
```

### Parameters

*viewId*

The original ID of View

*startRow*

The row to start fetching records from (0 by default)

*rowsPerPage*

The maximum number of row to fetch in one call (user-specified value by default)

*composite*

The depth of recursion of dependent records to fetch (0 by default)

*objNames*

A comma-separated list of dependent object names to fetch (use "\*" for all objects - default value)

*fieldNames*

A comma-separated list of field names to fetch (use "\*" for all fields - default value)

A callback function that will receive array of fetched records as first parameter

### Example

The following example fetches set of rows and processes results in a loop:

```
function my_callback(arr) {
    for (var k=0; k<arr.length; k++) {
        var amount = arr[k]['amount'];
        var price = arr[k] ['price'];
```

```
    // Do something...
}
}

rbf_getPage(123456, 0, 100, null, null, "amount,price", my_callback);
```

## rbf\_getPage2()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Similar to [rbf\\_getPage\(\)](#) on page 629, this function fetches a set of records in a view, including (optionally) dependent records, but allows you to apply a filter to the results. It brings info back to the page using asynchronous AJAX mechanism. The amount of processing and time required to get complete results can vary widely, depending on whether it fetches related records and the number of rows you specify per page.

### Syntax

```
rbf_getPage2(viewId, startRow, rowsPerPage, composite, objNames,
fieldList, filterName, filterValue, callback)
```

### Parameters

*viewId*

The original ID of View

*startRow*

The row to start fetching records from (0 by default)

*rowsPerPage*

The maximum number of row to fetch in one call (user-specified value by default)

*composite*

The depth of recursion of dependent records to fetch (0 by default)

*objNames*

A comma-separated list of dependent object names to fetch (use "\*" for all objects - default value)

*fieldNames*

A comma-separated list of field names to fetch (use "\*" for all fields - default value)

*filterName*

The name of field used to filter output records

*filterValue*

A value used for filtering in the the same format as for spreadsheet import

A callback function that will receive array of fetched records as first parameter

## rbf\_getRelatedFields()

### Purpose

For native Rollbase objects, this function retrieves the values of selected field for records related for a selected Rollbase record.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [rbf\\_getRelatedFields2\(\)](#) on page 632.

---

### Syntax

```
rbf_getRelatedFields(relName, id, fieldName, callback)
```

### Parameters

*relName*

The integration name of selected Relationship definition

*id*

The ID of selected Object record

*fieldName*

The integration name of Field to retrieve (for related records)

*callback*

A callback function that will receive parameters: *relName*, *id*, an array of received values (one value per related record).

### Example

The following example reads field "amount" from related records and makes these values available for computations:

```
function my_callback(relName, objId, values) {
    for (var i=0; i<values.length; i++) {
        var amount = values[i];
        // Do something...
    }
}

rbf_getRelatedFields("R12345", id, "amount ", my_callback);
```

## rbf\_getRelatedFields2()

### Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this function retrieves the values of selected field for records related for a selected Rollbase record.

---

**Note:** For native Rollbase objects, you can use the method, [rbf\\_getRelatedFields\(\)](#) on page 631.

---

### Syntax

```
rbf_getRelatedFields2(relName, objName, id, fieldName, callback)
```

### Parameters

*relName*

The integration name of selected Relationship definition

*objName*

The integration name of selected Object definition

*id*

The ID of selected Object record

*fieldName*

The integration name of Field to retrieve (for related records)

*callback*

A callback function that will receive parameters: *relName*, *id*, an array of received values (one value per related record).

### Example

The following example reads field "amount" from related records and makes these values available for computations:

```
function my_callback(relName, objId, values) {
    for (var i=0; i<values.length; i++) {
        var amount = values[i];
        // Do something...
    }
}

rbf_getRelatedFields2("R12345", "obj1", id, "amount ", my_callback);
```

## rbf\_getRelatedIds()

### Purpose

For native Rollbase objects, this function retrieves the ids of records related for a selected Rollbase record.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [rbf\\_getRelatedIds2\(\)](#) on page 633.

---

### Syntax

```
rbf_getRelatedIds(relName, id, callback)
```

### Parameters

*relName*

Integration name of selected Relationship definition

*id*

ID of selected Object record

*callback*

A callback function that will receive parameters: *relName*, *id*, an array of related ids.

## rbf\_getRelatedIds2()

### Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this function retrieves the ids of records related for a selected Rollbase record.

---

**Note:** For native Rollbase objects, you can use the method, [rbf\\_getRelatedIds\(\)](#) on page 633.

---

### Syntax

```
rbf_getRelatedIds2(relName, objName, id, callback)
```

### Parameters

*relName*

Integration name of selected Relationship definition

*objName*

The integration name of selected Object definition

*id*

ID of selected Object record

*callback*A callback function that will receive parameters: `relName`, `id`, an array of related ids.

## Example

```
function my_callback(relName, objId, values) {  
    for (var i=0; i<values.length; i++) {  
        var amount = values[i];  
        // Do something...  
    }  
}  
  
rbf_getRelatedIds2("R12345", "obj1", id, my_callback);
```

## rbf\_runTrigger()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function sends AJAX request to run a trigger on selected record. The edit operation on The selected record must be allowed for the current user or API User role, see [Built-in Security Levels](#) on page 377.

## Syntax

```
rbf_runTrigger(objName, id, triggerID, checkCondition, callback)
```

## Parameters

*objName*

The integration name of the selected object definition

*id*

ID of the selected object record

*triggerID*

The integration name (string) or ID (number) of trigger to run

*checkCondition*

If true, check trigger's condition formula before running trigger, otherwise ignore condition formula.

*callback*

A function to be called when AJAX request returns (optional). Will receive single parameter: true or false depending on whether trigger has actually ran.

## Example

If you want the system to send email when portal user clicks on particular link on your portal page do the following:

1. Prepare an email template.
2. Prepare a trigger to send email. This example uses the integration name `sendEmail`
3. Make sure the Edit operation is allowed for either Portal User or API User.
4. Prepare the following link:

```
<a href="#" onclick='rbf_runTrigger("myVisitor", { !#CURR_VISIT.id }, "sendEmail", false);'>Send Email</a>
```

For OpenEdge or any external objects use the `{ !#UID }` token in quotes:

```
<a href="#" onclick='rbf_runTrigger("oeObject", { !#UID }, "sendEmail", false);'>Send Email</a>
```

## rbf\_selectNumber()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function runs a SQL SELECT query on the server and returns a single decimal number to the callback function. This is a simplified version of `rbf_selectValue` suitable for calculating totals etc.

## Syntax

```
rbf_selectNumber(query, callback)
```

## Parameters

*query*

SQL SELECT query.

*callback*

A callback function that will receive a single value returned by the query.

## rbf\_selectQuery()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function runs a SQL SELECT query on the server and returns results as a 2D-array to the callback function.

### Syntax

```
rbf_selectQuery(query, maxRows, callback)
```

### Parameters

*query*

SQL SELECT query. See Query API section below for examples and limitations.

*maxRows*

Maximum number of rows to retrieve (1-20,000 range)

*callback*

A callback function that will receive a 2D JSON array with query results as parameter.

### Example

The following example reads two fields from record defined by template token `{!id}`:

```
function my_callback(values) {
    var amount = values[0][0];
    var price = values[0][1];
    // Do something...
}

rbf_selectQuery("select amount, price from order where id={!id}", 1,
my_callback);
```

## rbf\_selectQuery2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function runs a SQL SELECT query on the server and returns results as a 2D-array to the callback function.

## Syntax

```
rbf_selectQuery(query, rowFrom, maxRows, callback)
```

## Parameters

*query*

SQL SELECT query. See Query API section below for examples and limitations.

*rowFrom*

Number of row to start output (0 based)

*maxRows*

Maximum number of rows to retrieve (1-20,000 range)

*callback*

A callback function that will receive a 2D JSON array with query results as parameter.

## Example

The following example reads two fields from record defined by template token `{!id}`:

```
function my_callback(values) {
    var amount = values[0][0];
    var price  = values[0][1];
    // Do something...
}

rbf_selectQuery("select amount, price from order where id={!id}", 1,
my_callback);
```

## rbf\_selectValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function runs a SQL SELECT query on the server and returns a single value to the callback function.

## Syntax

```
rbf_selectValue(query, callback)
```

## Parameters

*query*

SQL SELECT query.

*callback*

A callback function that will receive a single JSON value brought by the query.

## rbf\_setField()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function sets the value of specified field for a selected Rollbase record and modifies the actual record value without changing the UI presentation. Compare with rbf\_setFieldValue API which sets UI presentation but does not immediately change backend value.

If an error occurred during update operation, an error notification procedure will be invoked . The edit operation on selected record must be allowed for the current user or API User role.

### Syntax

```
rbf_setField(objName, id, fieldName, fieldValue, useIds)
```

### Parameters

*objName*

The integration name of the selected Object definition

*id*

ID of the selected object record

*fieldName*

The name of the field for which the value will be set

*fieldValue*

The new value for the field

*useIds*

If true, the API will accept numeric IDs; if false (default) the API will return integration codes for status fields and picklists

## rbf\_updateRecord()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This method modifies record values of a Rollbase object without changing UI presentation. If error occurred during operation error notification procedure will be invoked (see [rbf\\_setErrorsCallback](#)).

---

**Note:** The edit operation on the selected record must be allowed for the current user or API User role, see [Built-in Security Levels](#) on page 377

---

## Syntax

```
rbf_updateRecord(objName, id, fieldMap, useIds, callback)
```

### Parameters

*objName*

The integration name of the selected Object definition

*id*

ID of the selected Object record

*fieldMap*

An array of named parameters of the form: name = field name, value = field value

*useIds*

If `true`, the API will accept numeric IDs; if `false` (default) the API will return integration codes for status fields and picklists

*callback*

A function to be executed when the operation is successful. It does not receive any parameters. This is an optional parameter.

# Field Manipulation

The functions in this section allow you to access and modify record fields.

## rbf\_getFieldContent()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function retrieves the HTML content of a read-only field on the current page.

### Syntax

```
rbf_getFieldContent(fieldname)
```

## Parameters

*fieldName*

The integration name of field

## rbf\_getFieldValue()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function retrieves the value of a field on current page with a given integration name. It will return null if the field does not exist or is not included on the current page.

## Syntax

`rbf_getFieldValue(fieldName)`

## Parameters

*fieldName*

The integration name of field

## rbf\_getPicklistCode()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns an integration code or the text name of a single-selection picklist's or radio buttons currently selected option. This function works in **View** and **Edit** pages.

## Syntax

`rbf_getPicklistCode(fieldName)`

## Parameters

*fieldName*

The integration name of field

## rbf\_getPicklistCodes()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function returns a list of comma-separated integration codes corresponding to the currently selected options in a multi-select picklist field or group of checkboxes field. This function works in View and Edit mode.

### Syntax

```
rbf_getPicklistCode (fieldName)
```

*fieldName*

The integration name of field

## rbf\_setFieldContent()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function sets the HTML content for a formula or template field on current page. It will have no effect if the field does not exist on the current page.

### Syntax

```
rbf_setFieldContent (fieldname, value)
```

### Parameters

*fieldName*

The integration name of field

*value*

The new value to set

## rbf\_setFieldValue()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function sets the value of a field on current page with a given integration name. It will run the field's `onchange` event handler if there is event handling code attached to that.

### Syntax

```
rbf_setFieldValue(fieldname, value)
```

#### Parameters

*fieldName*

The integration name of field to set

*value*

Value to set

## rbf\_setFieldDisabled()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function disables or enables input field on current page. It will have no effect if the field does not exist on the current page.

**Note:** According to the HTML specification, disabled fields do not send data back to server when HTML form is submitted. So if you wish disabled to send data (probably set by some client-side script) you can re-enable input field in page's `onSubmit` event handler.

---

### Syntax

```
rbf_setFieldDisabled(fieldname, isDisabled)
```

#### Parameters

*fieldName*

The integration name of field

*isDisabled*

The new value for "disabled" field's property

## rbf\_setPicklistCode()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function sets the value of a picklist field or radio buttons if the code attribute of the particular option is equal to the *optCode* parameter. It will also invoke `onchange` (`onclick`) event handling code associated with this field.

### Syntax

```
rbf_setPicklistCode(fieldName, optCode)
```

### Parameters

*fieldName*

The integration name of field

*optCode*

This value will be compared to the integration code of a particular picklist option

## Data Formatting

The methods in this section allow you to manipulate numbers, dates, and currency.

## rbf\_getDate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function creates a JavaScript Date Object from a string. It returns null for an empty string.

### Syntax

```
rbf_getDate(value)
```

## Parameters

*value*

The string to convert to a Date Object

## rbf\_getDigits()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Extracts digits from string value and ignores all other characters.

## Syntax

`rbf_getDigits(value)`

## Parameters

*value*

The string from which to extract digits

## rbf\_getFloat()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Converts a given value from a string into a float number. It ignores symbols (e.g., \$).

## Syntax

`rbf_getFloat(value)`

## Parameters

*value*

The string to convert

## rbf\_formatCurrency()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Formats a given number into a currency string.

### Syntax

```
rbf_formatCurrency(value, currSymbol, showZero, decPlaces, thsSep)
```

### Parameters

*value*

The number to format

*currSymbol*

currency symbol (\$ by default)

*showZero*

If true, empty and NaN values are treated as "\$0.00", otherwise as empty string ""

*decPlaces*

The number of decimal places (2 by default)

*thsSep*

The thousands separator (, by default)

## rbf\_formatDate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function formats a JavaScript Date Object into a string suitable for storing in Date input fields.

### Syntax

```
rbf_formatDate(d, format)
```

## Parameters

*d*

Javascript Date Object

*format*

The format to apply to date such as dd/mm/yyyy, mm.dd.yy, etc.

## rbf\_formatUsingMask()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Formats a string containing digits as specified in the *mask* parameter.

## Syntax

```
rbf_formatUsingMask(value, mask)
```

## Parameters

*value*

The string to format

*mask*

The input mask, which uses # as a placeholder for a digit

## Example

```
var x = rbf_formatUsingMask("123456789", "###-##-####");
```

Result: 123-45-6789

## rbf\_getInt()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

This function converts a given value from a string into an integer number. It ignores symbols (e.g., \$).

## Syntax

```
rbf_getInt (value)
```

## Parameters

*value*

The string to convert

## Grid Control Examples and API

Grid controls allow users creating or editing a record to add, delete, and edit all records for a particular relationship. For example, the screen shot below shows a grid control that displays the furnishings associated with a particular room:

Quantity	Furniture	Description	Size
24	Bistro Chairs	Plastic molded	
2	Glass Buffet	Modern design, <del>countertop height</del>	50" X 24"
6	Glass Tables	Seats 4	
1	Revolving Dessert Case	Refrigerated, with 6 glass shelves	36" square

The JavaScript API functions described in this section can be used to manipulate the rows and cells of an existing Grid Control. At least one grid control must exist on a page before you can configure grids or use the grid APIs. See [Using Grid Controls to Manage Multiple Records](#) on page 227 for information on adding, configuring and using grid controls. All Grid API names have the suffix `2` to distinguish them from the deprecated API.

Field-level HTML event handlers do not apply to fields in a grid control since there may be any number of instances of the same field in a grid. Therefore, Rollbase provides special grid event handling options. Grid Controls provide three types of event handlers:

- `onCreate` — invoked when a new row is added
- `onUpdate` — invoked when one of the controls in the current row is modified
- `onDelete` — invoked when a row is deleted

These handlers work similarly to event handlers attached to individual HTML input fields. However, the symbol `@@` in grid event handlers will be replaced by the current row number and the symbol, `##`, will be replaced by the `gridNo` parameter used by Rollbase Client-side API.

Rollbase allows multiple grid controls on page. Therefore to distinguish between the grids on a page, all grid-related methods take the 0-based sequential number of the grid as first parameter. To find the grid number, navigate to the page containing the grid, click **Configure Grid**, and navigate to the last page of the wizard, as shown below:

## Check Out: Configure Grid of Related Titles

Fields Properties				
Field Label	Data Type	Integration Name	Required In Grid	Read Only
Author	Text	author	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Title	Record Name	name	<input checked="" type="checkbox"/>	<input type="checkbox"/>

---

**Initial Rows Sorting**

Select sorting criteria for existing records. This does not imply dynamic sorting for updated or newly created records.

Sort By	<input style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;" type="button" value="Title"/>	<input style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;" type="button" value="Title"/>
---------	---	---

---

**JavaScript Event Handlers**

Tip: Using this feature you can assign JavaScript event handling code to be executed when rows of this grid are created, updated, or deleted. Use @@ symbol as a placeholder for row's number and ## symbol as a placeholder for Grid number parameter user by API. [Learn more about JavaScript event handlers...](#)

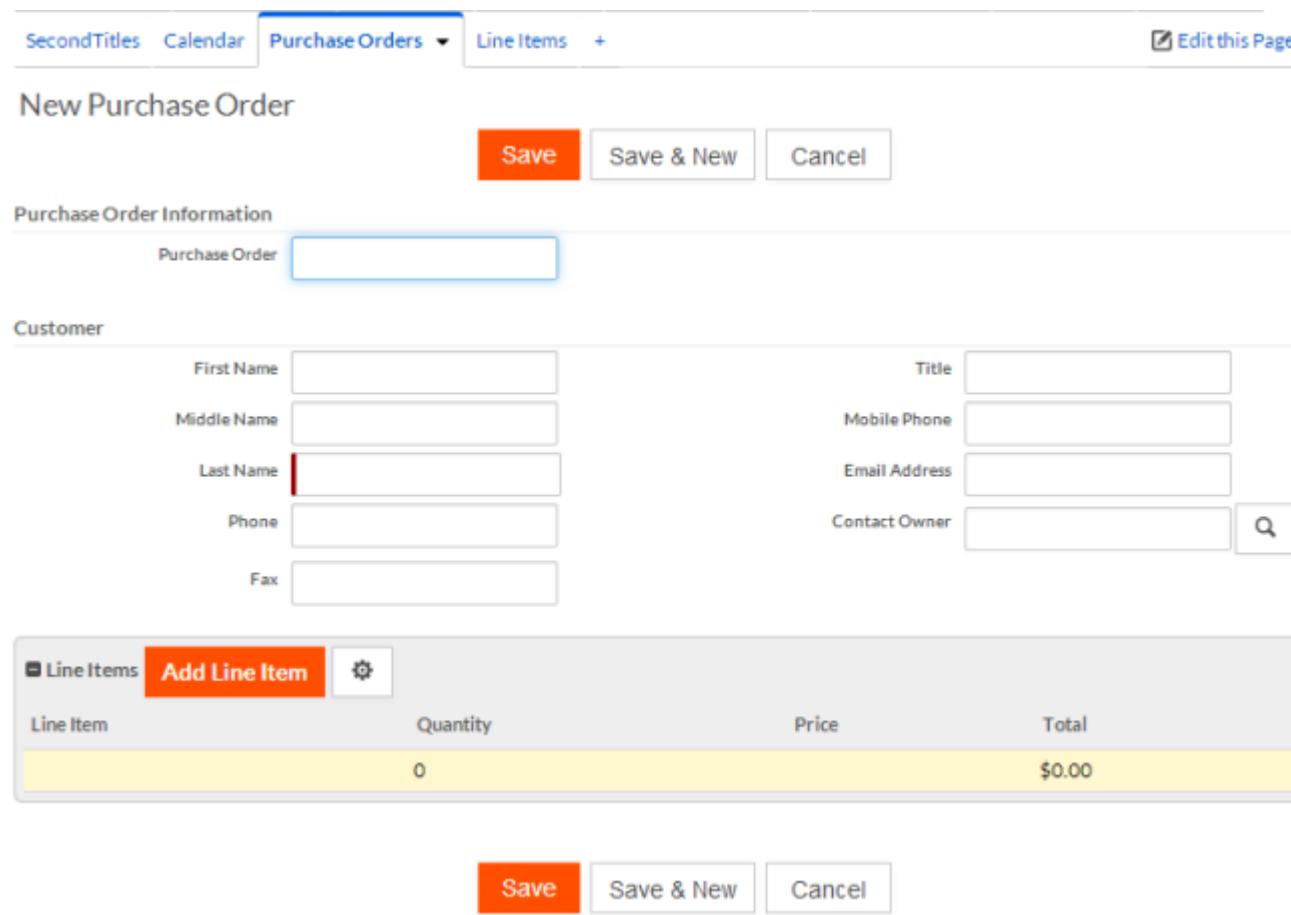
Grid: 0	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
Number	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
onCreate =	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
" "	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
onUpdate =	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
" "	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
onDelete =	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>
" "	<input style="width: 100%; border: 1px solid #ccc; height: 20px;" type="text"/>

## Simple Grid Calculation Example

You can add JavaScript that dynamically updates totals as a user adds or modifies grid records. This example uses a Purchase Order object that has a 1-to-many relationship with a Line Item object. The Line Item object has three fields:

1. An integer field with the integration name: quantity
2. A currency field with the integration name: price
3. A formula field with the integration name: total, a formula body: `{!quantity}*{!price}`, and a return type of currency.

The **New Purchase Order** page with a grid added will look like the following screen. Users adding line items must save the parent record before they can view the total quantity and price.

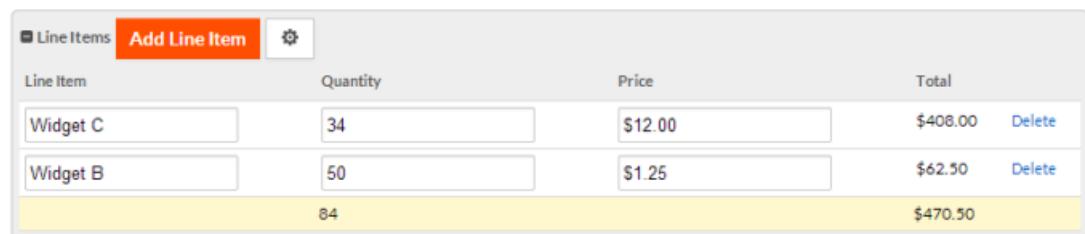


The screenshot shows the 'New Purchase Order' page. At the top, there are tabs for 'Second Titles', 'Calendar', 'Purchase Orders' (which is selected and highlighted in blue), and 'Line Items'. To the right of the tabs are buttons for 'Save', 'Save & New', and 'Cancel'. Below the tabs, the page title 'New Purchase Order' is displayed. Under 'Purchase Order Information', there is a 'Purchase Order' field. The 'Customer' section contains fields for First Name, Middle Name, Last Name, Phone, Fax, Title, Mobile Phone, Email Address, and Contact Owner. Below this is a grid control with the following structure:

Line Item	Quantity	Price	Total
	0		\$0.00

At the bottom of the grid are buttons for 'Save', 'Save & New', and 'Cancel'.

By adding an event handler, a script component, and JavaScript to perform the calculations, the grid totals will dynamically update when the user tabs out of the calculated fields, as shown below:



The screenshot shows the 'New Purchase Order' page with the grid control. The grid now displays the following data:

Line Item	Quantity	Price	Total
Widget C	34	\$12.00	\$408.00
Widget B	50	\$1.25	\$62.50
	84		\$470.50

To recreate this example, follow these steps:

1. Use the **Page Editor** to add a Grid Control to the **Edit** and **New** pages for Purchase Order objects. See [Adding a Grid Control with the Page Editor](#) on page 228 for detailed procedures.
2. Configure the grids as follows:
  - a. Create a Purchase Order record. On the **New** page, click **Configure Grid**.
  - b. Select the relationship **Purchase Order-Line Item**.
  - c. Select these fields to be displayed in the grid: **Quantity**, **Price**, and **Total**.

- d. For **Grid Totals**, select the **Quantity** and **Total** columns and select the **Total** operation for both.
  - e. In the JavaScript Event Handlers section **onUpdate** field, add the following call (the @@ symbol will be replaced at runtime with the current row number):

```
rbf_showGridRow(@@)
```
  - f. Click **Save & Synchronize**.
  - g. Select the **Edit** page.
  - h. Click **Save** and the changes will be applied to both pages.
3. Add the JavaScript for calculating totals:
  - a. Click **Design This Page**.
  - b. From the **Create** section of the left pane, drag a **New Section** component onto the page below the **Grid Control**.
  - c. Select the new section and from the **Properties** section in the left pane remove the **Section Title**. Since the Javascript will not display on the page, you don't want the title to show either.
  - d. Drag a new **Script Component** from the left pane and drop it in the **New Section**.
  - e. Click **Edit**.
  - f. In the editor, add the following Javascript:

```
<script>
    function rbf_showGridRow(rowIndex) {
        var a = rbf_getFloat(rbf_getGridValue2(0, 'quantity', rowIndex));
        var p = rbf_getFloat(rbf_getGridValue2(0, 'price', rowIndex));
        var t = a*p; rbf_setGridContent2(0, 'total', rowIndex,
        rbf_formatCurrency(t));
    }
</script>
```
  - g. Click **Save & Synchronize**.
  - h. Select the **Edit** page.
  - i. Click **Save** and the changes will be applied to both pages.
4. Test by creating a new record and adding line items. Note that when you tab out of the quantity and price fields, the totals update automatically.

## Advanced Grid Example

Consider a more complex use case than that shown in [Simple Grid Calculation Example](#) on page 648. Suppose that you want the user to select an item price from a price list instead of inputting it directly. To the data model described in [Simple Grid Calculation Example](#) on page 648 you could add a relationship (R8011504) with a Catalog Item object.

The JavaScript to handle this use case would be as follows:

```
<script>
var m_rowIndex = -1;

function rbf_findPrice(rowIndex) {
    m_rowIndex = -1;
    var catId = rbf_getInt(rbf_getGridValue2(0, 'R8011504', rowIndex));
    var p = rbf_getFloat(rbf_getGridValue2(0, 'price', rowIndex));
```

```
if (catId>0 && p<=0) {
  rbf_getFields("catalog_item", catId, "msrp", rbf_callback);
  mRowIndex = rowIndex;
}
else {
  rbf_showGridRow(rowIndex);
}
}

function rbf_showGridRow(rowIndex) {
  var a = rbf_getFloat(rbf_getGridValue2(0, 'quantity', rowIndex));
  var p = rbf_getFloat(rbf_getGridValue2(0, 'price', rowIndex));
  var t = a*p;
  rbf_setGridContent2(0, 'total', rowIndex, rbf_formatCurrency(t));
}

function rbf_callback(objName, objId, values) {
  var p = values["msrp"];
  if (mRowIndex >= 0) {
    rbf_setGridValue2('price', mRowIndex, p);
    rbf_showGridRow(mRowIndex);
  }
}
</script>
```

Set `rbf_findPrice(@@)` as an `onUpdate` event handler in the Grid.

The functions perform the following:

- If a related Catalog Item is selected but the Price field is not set yet, the `rbf_findPrice()` function sends an AJAX request using `rbf_getFields()` to determine the price of the selected item and return it. Otherwise `rbf_showGridRow()` calculates the Line Item totals.
- The `rbf_callback` function processes the information fetched by the AJAX response. It finds the 'msrp' value from response and copies it to the 'price' field, and then updates the grid row using `rbf_showGridRow()`.

## rbf\_addGridRow()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Adds a new empty row to a grid.

### Syntax

`rbf_addGridRow(gridNo)`

### Parameters

*gridNo*

The 0-based number of Grid Control on page

## rbf\_delGridRow()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Deletes row with given index from a grid.

### Syntax

```
rbf_delGridRow(gridNo, rowIndex)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

*rowIndex*

The row number

## rbf\_getGridField()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Gets the name of the field that was last changed in the Grid.

### Syntax

```
rbf_getGridField(gridNo)
```

### Parameters

Parameter

Description

*gridNo*

The 0-based number of Grid Control on page

## rbf\_getGridPicklistCode



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Gets the integration code of a Grid Control picklist field with the given integration name at the specified row number. Returns null if the Grid Control does not exist on the current Page.

### Syntax

```
rbf_getGridPicklistCode(gridNo, fieldName, rowIndex);
```

### Parameters

*gridNo*

The 0-based number of the Grid Control on page

*fieldName*

The integration name of the input field

*rowIndex*

The row number

### Return Value

Null if the Grid Control does not exist on the page.

## rbf\_getGridPicklistControl()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns an integration code of the picklist field in the specified **Grid Control** at the specified row number. Returns null if the **Grid Control** does not exist on the current page. This method is similar to [rbf\\_getPicklistCode\(\)](#) on page 640.

### Syntax

```
rbf_getPicklistCode(gridNo, fieldName, rowIndex)
```

## Parameters

*gridNo*

The 0-based **Grid Control** number

*fieldName*

The integration name of the field

*rowIndex*

The row number

## Return Value

Null if the **Grid Control** does not exist on the current page.

## rbf\_getGridValue2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Gets the value of a single field in a Grid Control with the given integration name at the specified row number. Returns null if a field or Grid Control does not exist on the current the Page.

## Syntax

`rbf_getGridValue2(gridNo, fieldName, rowIndex)`

## Parameters

*gridNo*

The 0-based number of Grid Control on page

*fieldName*

The integration name of the input field

*rowIndex*

The row number

## rbf\_getMaxRowIndex2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Returns the maximum value for the rowIndex in a grid. This may be different from the total number of rows in the grid.

---

**Note:** A `rowIndex` is assigned to each grid row when it is first created or rendered. Deleting a row does not affect (decrement) indexes of other rows.

---

## Syntax

```
rbf_getMaxRowIndex2 (gridNo)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

## Example

This function can be used to iterate through all grid rows:

```
var n = rbf_getMaxRowIndex2(0);
for (var rowIndex=0; rowIndex<n; rowIndex++) {
    var x = rbf_getGridValue2(0, "amount", rowIndex);
```

## rbf\_setGridContent2()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Sets the value of a picklist field or radio buttons in grid control. It will also invoke onchange (onclick) event handling code associated with this field. Has no effect if a field or Grid Control does not exist on the current Page.

## Syntax

```
rbf_setGridContent2 (gridNo, fieldName, rowIndex, value)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

*fieldName*

The integration name of formula or template field

*rowIndex*

The row number

*value*

The HTML content to be assigned to the cell

## rbf\_setGridPicklistCode()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Sets integration code of picklist field in Grid Control with the given integration name at the specified row number. Has no effect if a field or Grid Control does not exist on the current Page. This function is similar to [rbf\\_setPicklistCode\(\)](#) on page 643.

### Syntax

```
rbf_setGridPicklistCode(gridNo, fieldName, rowIndex, optCode)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

*fieldName*

The integration name of the input field

*rowIndex*

The row number

*optCode*

Value will be compared to the integration code of a particular picklist option

## rbf\_setGridValue2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Sets value of single editable field in Grid Control with the given integration name at the specified row number. Has no effect if a field or Grid Control does not exist on the current Page.

## Syntax

```
rbf_setGridValue2(gridNo, fieldName, rowIndex, value)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

*fieldName*

The integration name of the input field

*rowIndex*

The row number

*value*

HTML content to set in field

## rbf\_showGridTotals()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Calculate and display totals for grid.

### Syntax

```
rbf_showGridTotals(gridNo)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

## rbf\_sumGridColumn2()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Retrieves the sum of column values in a grid.

## Syntax

```
rbf_sumGridColumn2(gridNo, fieldName)
```

### Parameters

*gridNo*

The 0-based number of Grid Control on page

*fieldName*

The integration name of formula or numeric input field

## Miscellaneous

Methods in this section allow you to obtain information such as integration codes and IDs, determine or set the state of checkboxes, and set a callback for error conditions.

## rbf\_getCodeById()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the integration code of a picklist item, status, or template with the given ID.

### Syntax

```
rbf_getCodeById(objName, fieldName, id, callback)
```

### Parameters

*objName*

The object definition integration name

*fieldName*

The integration name of picklist field

*id*

ID of picklist item

*callback*

The callback function that will receive code of picklist item as first parameter (null if not found).

## rbf\_getExchangeRate()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns an exchange rate between two currencies on a given date.

### Syntax

```
rbf_getExchangeRate(srcCode, destCode, date, callback)
```

### Parameters

*srcCode*

The currency code to convert from

*destCode*

The currency code to convert to

*date*

The date for exchange rate (current date by default)

*callback*

The callback function that will receive exchange rate as first parameter (null if not found)

### Example

```
rbf_getExchangeRate("EUR", "USD", null, function(rate) { alert("EUR/USD="+rate);});
```

## rbf\_getIdByCode()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns the ID of a picklist item, status, or template with the given code.

### Syntax

```
rbf_getIdByCode(objName, fieldName, code, callback)
```

## Parameters

*objName*

Object definition integration name

*fieldName*

The integration name of picklist field

*code*

The integration name of picklist item

*callback*

The callback function that will receive the ID of picklist item as first parameter (-1 if not found).

## rbf\_isChecked()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns true if a checkbox with a given integration name is checked; false otherwise.

### Syntax

`rbf_isChecked(fieldName)`

### Parameters

*fieldName*

The integration name of checkbox field

## rbf\_isEmpty()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Returns true if argument is null or empty string.

### Syntax

`rbf_isEmpty(arg)`

## Parameters

*arg*

The argument to be tested

## Example

```
rbf_isEmpty() -> true
rbf_isEmpty(-1.23) -> false
rbf_isEmpty(0) -> true
rbf_isEmpty('') -> true
rbf_isEmpty(false) -> true
rbf_isEmpty("A") -> false
rbf_isEmpty("0") -> false
```

## rbf\_isSelected()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Returns true if record with given id is selected (check box checked) in list view.

## Syntax

```
rbf_isSelected(recordId)
```

## Parameters

Parameter

Description

*recordId*

The numeric ID of record

## rbf\_isZero()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function can be useful in validation scripts, it returns true if the argument is any of the following:

- null
- the number zero
- a value that cannot be converted into a number

## Syntax

```
rbf_isZero(arg)
```

*arg*

The argument to be tested.

## Example

```
rbf_isZero() -> true
    rbf_isZero(-1.23) -> false
    rbf_isZero(0) -> true
    rbf_isZero('') -> true
    rbf_isZero(false) -> true
    rbf_isZero("A") -> true
    rbf_isZero("0") -> true
```

## rbf\_setChecked()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Sets the state of the checkbox field with the given integration name to checked (`true`) or unchecked (`false`). It will invoke any existing `onchange` event handlers for that field.

## Syntax

```
rbf_setChecked(fieldName, isChecked)
```

## Parameters

*fieldName*

The integration name of the checkbox field

*isChecked*

`true` or `false`

## rbf\_setErrorsCallback()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Assign a callback function to be used when processing API errors, such as errors in SQL queries, etc. By default the API uses the `rbf_showInfoMessage()` method to display error messages, but this behavior can be customized.

### Syntax

```
rbf_setErrorsCallback(callback)
```

### Parameters

*callback*

The callback function, which will receive an error message as a first parameter and API name (which caused the error) as second parameter

### Example

```
var callback = function(errMsg, apiName) {  
    alert("ERROR: "+errMsg+" in: "+apiName);  
};  
rbf_setErrorsCallback(callback);
```

## rbf\_setLookupFilter()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

This function sets a filter on the selection of related records for a lookup field. This method only applies to **Selector** type lookup fields and cannot be used for dependent lookups. This method can be used for dynamic filtering of available lookup choices.

### Syntax

```
rbf_setLookupFilter(fieldName, filterName, filterValue);
```

### Parameters

*fieldName*

Integration name of the lookup field

*filterName*

Name of the related object field used for filtering

*filterValue*

Value of fields to display

## Return Value

### Example

The call in the following example ensures that the selector window opening on the R12345 lookup field only displays records where the value of the `club_member` field equals true.

```
rbf_setLookupFilter('R12345', 'club_member', 'true');
```

## rbf\_setSelected()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Selects/unselects record with given ID in list view.

### Syntax

```
rbf_setSelected(recordId, selected)
```

### Parameters

Parameter

Description

*recordId*

The numeric ID of record

*selected*

true or false

### Example

```
var x = rbf_isSelected(recordId);
rbf_setSelected(recordId, !x);
```

## rbf\_startServerDebugging()

### Purpose

Starts server-side debugging. This API is used for debugging purposes.

### Syntax

```
rbf_startServerDebugging(callback)
```

### Parameters

*callback*

The function that is executed when server debugging operation is started. This doesn't receive any parameters.

### Permissions

It is only available for Administrative users.

### Example

```
rbf_startServerDebugging(function() {  
    alert("Start debugging server-side API");  
});
```

## rbf\_stopServerDebugging()

### Purpose

Stops server-side debugging process started by [rbf\\_startServerDebugging\(\)](#) on page 665. This API is used for displaying the results of debugging.

### Syntax

```
rbf_stopServerDebugging(callback)
```

### Parameters

*callback*

The function that is executed when server debugging operation is completed. This receives a string parameter that contains a summary of debug operations.

---

**Note:** Outputs from `rbv_api.println()` and similar functions are included and displayed as the results of debugging.

---

### Permissions

It is only available for Administrative users.

## Example

```
rbf_startServerDebugging(function() {  
    alert("Start debugging server-side API");  
});
```

# AJAX Metadata API

The server-side AJAX APIs provides methods to retrieve, create, update, and delete the following Rollbase metadata elements:

- Application
- Object
- Field
- Relationship

---

**Note:** All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session to use these methods by logging in with credentials for an administrative user.

---

To use the metadata API you must reference `metadata.js` on your web page. Use the **Page Editor** and add the following HTML code-snippet to your Script component:

```
<script src='..../js/metadata.js' type='text/javascript' charset='utf-8'></script>
```

---

**Note:** This code-snippet is available through the **Helper** user interface.

---

For more information about XML metadata definitions for Rollbase applications, see [Metadata XML Reference](#) on page 687.

## rbf\_createApplicationDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Creates an application from an XML document.

### Syntax

```
rbf_createApplicationDef(xmlString, callback, errorCallback)
```

## Parameters

*xmlString*

A string containing the XML document describing the application to be created.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML elements in an Application, see [Application XML Elements](#) on page 688.

## rbf\_createFieldDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Creates a field definition from an XML document.

### Syntax

`rbf_createFieldDef(xmlString, callback, errorCallback)`

## Parameters

*xmlString*

A string containing the XML document describing the field to be created.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a field, see [Field XML Definition](#) on page 695.

## rbf\_createObjectDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Creates a new object definition, including any specified fields and relationships, from an XML document.

### Syntax

```
rbf_createObjectDef(xmlString, callback, errorCallback)
```

### Parameters

*xmlString*

A string containing the XML document describing the object to be created.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in an object, see [Object XML Definition](#) on page 690.

## rbf\_createRelationshipDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Creates a new relationship from an XML document.

### Syntax

```
rbf_createRelationshipDef(xmlString, callback, errorCallback)
```

## Parameters

*xmlString*

A string containing the XML document describing the relationship to be created.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a relationship, see [Relationship XML Definition](#) on page 703.

## rbf\_deleteApplicationDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Deletes an application definition and all of its component definitions.

## Syntax

`rbf_deleteApplicationDef(appId, callback, errorCallback)`

## Parameters

*appId*

The original application ID.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML elements in an Application, see [Application XML Elements](#) on page 688.

---

## rbf\_deleteFieldDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Deletes a field definition from the specified object.

### Syntax

```
rbf_deleteFieldDef(objName, fieldName, callback, errorCallback)
```

### Parameters

*objName*

The object Integration name.

*fieldName*

The field integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a field, see [Field XML Definition](#) on page 695.

## rbf\_deleteObjectDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Deletes an object definition, including its fields and relationships.

## Syntax

```
rbf_deleteObjectDef(objName, callback, errorCallback)
```

## Parameters

*objName*

The object Integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in an object, see [Object XML Definition](#) on page 690.

## rbf\_deleteRelationshipDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Deletes a relationship definition.

## Syntax

```
rbf_deleteRelationshipDef(relName, callback, errorCallback)
```

## Parameters

*relName*

The relationship integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a relationship, see [Relationship XML Definition](#) on page 703.

## rbf\_getApplicationDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Retrieves an XML description of a Rollbase application.

### Syntax

```
rbf_getApplicationDef(appId, callback, errorCallback)
```

### Parameters

*appId*

The original application ID.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML elements in an application, see [Application XML Elements](#) on page 688.

## rbf\_getFieldDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Retrieves a field definition as an XML document.

## Syntax

```
rbf_getFieldDef(objname, fieldName, callback, errorCallback)
```

## Parameters

*objName*

The object Integration name.

*fieldName*

The field integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a field, see [Field XML Definition](#) on page 695.

## rbf\_getObjectDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Retrieves the specified object definition and returns it as an XML document.

## Syntax

```
rbf_getObjectDef(objName, callback, errorCallback)
```

## Parameters

*objName*

The object Integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in an object, see [Object XML Definition](#) on page 690.

## rbf\_getRelationshipDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Retrieves a relationship definition as an XML document.

### Syntax

```
rbf_getRelationshipDef(relName, callback, errorCallback)
```

### Parameters

*relName*

The relationship integration name.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a relationship, see [Relationship XML Definition](#) on page 703.

## rbf\_updateApplicationDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

Updates an application from an XML document.

## Syntax

```
rbf_updateApplicationDef(xmlString, callback, errorCallback)
```

## Parameters

*xmlString*

A string containing the XML document describing the application to be updated.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML elements in an Application, see [Application XML Elements](#) on page 688.

## rbf\_updateFieldDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Updates a field definition from an XML document.

## Syntax

```
rbf_updateFieldDef(xmlString, callback, errorCallback)
```

## Parameters

*xmlString*

A string containing the XML document describing the field to be updated.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a field, see [Field XML Definition](#) on page 695.

## rbf\_updateObjectDef()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Updates an object definition from an XML document.

### Syntax

```
rbf_updateObjectDef(xmlString, callback, errorCallback)
```

### Parameters

*xmlString*

A string containing the XML document describing the object to be updated.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in an object, see [Object XML Definition](#) on page 690.

## rbf\_updateRelationshipDef()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Updates a relationship definition from an XML document.

### Syntax

```
rbf_updateRelationshipDef(xmlString, callback, errorCallback)
```

### Parameters

*xmlString*

A string containing the XML document describing the relationship to be updated.

*callback*

A function to process the specified operation and receive its result, in the XML format, as the first argument.

*errorCallback*

An optional parameter to receive an error message as the first argument. If you do not specify this parameter in the method call, the callback defined by `ref_setErrorsCallback(callback)` is invoked to receive the error message.

For information about the XML definitions in a relationship, see [Relationship XML Definition](#) on page 703.

## Display Functions

The Rollbase JavaScript API functions in this section can be useful in customizing UI presentation of Rollbase pages.

## rbf\_activatePageTab()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Activate page-level tab with given index (only for pages where tabs are enabled).

## Syntax

```
rbf_activatePageTab(tabIndex)
```

### Parameters

*tabIndex*

The 0-based index of page level tab to be activated

## rbf\_getSectionIdByTitle()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

This function finds ID of section with given title or null if section not found. This ID can be used in rbf\_setSectionCollapse and rbf\_showOrHideSection API.

Important: For multi-lingual customers sectionTitle parameter must use customer's base language, not user selected language.

## Syntax

```
rbf_getSectionIdByTitle(sectionTitle)
```

### Parameters

*sectionTitle*

Title of section on current page

## rbf\_getViewSelector()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

This function returns the view ID used by a lookup field for both auto-complete and pop-up selectors.

## Syntax

```
rbf_getViewSelector(fieldName)
```

## Parameters

*fieldName*

The integration name of lookup field

## rbf\_growl()

### Purpose

Displays a growl-type floating notification in the upper right side of the screen.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Syntax

```
rbf_growl(title, message, type);
```

### Parameters

*title*

Optional title to display. If none is specified, a default title will display.

*message*

The notification. If null or empty, the notification will not display.

*type*

One of info, success, error, or warn. If not specified, defaults to warn.

## rbf\_growlError()

### Purpose

Displays a growl-type floating notification in the upper right side of the screen. End-users have to close the notification to dismiss it.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Syntax

```
rbf_growlError(title, message);
```

## Parameters

### *title*

Optional title to display. If none is specified, a default title will display.

### *message*

The notification. If null or empty, the notification will not display.

## **rbf\_hideGrowl()**

### Purpose

Hides a notification if one is being displayed.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Syntax

```
rbf_hideGrowl();
```

## **rbf\_hideInfoMessage()**



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

This function hides the status message at the top of the page. This method implementation calls [rbf\\_hideGrowl\(\)](#) on page 680.

### Syntax

```
rbf_hideInfoMessage();
```

## **rbf\_growlInfo()**

### Purpose

Displays a growl-type floating notification in the upper right side of the screen. The notification closes automatically after approximately 3.5 seconds.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Syntax

```
rbf_growlInfo(title, message);
```

## Parameters

*title*

Optional title to display. If none is specified, a default title will display.

*message*

The notification. If null or empty, the notification will not display.

## rbf\_growlSuccess()

### Purpose

Displays a growl-type floating success notification in the upper right side of the screen. The notification closes automatically after approximately 3.5 seconds.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Syntax

## Parameters

*title*

Optional title to display. If none is specified, a default title will display.

*message*

The notification. If null or empty, the notification will not display.

## Return Value

## Example

Introduce example here.

## rbf\_growlWarning()

### Purpose

Displays a growl-type floating notification in the upper right side of the screen. End-users have to close the notification to dismiss it.



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Syntax

```
rbf_growlWarning(title, message);
```

### Parameters

*title*

Optional title to display. If none is specified, a default title will display.

*message*

The notification. If null or empty, the notification will not display.

## rbf\_showInfoMessage()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

### Purpose

Displays a status message on the top of the page, which will automatically disappear after 40 seconds. This method implementation calls [rbf\\_growlInfo\(\)](#) on page 680

### Syntax

```
rbf_showInfoMessage(message, isError)
```

### Parameters

*message*

The message text

*isError*

Value of `true` if message represents an error (errors are shown in dark red text rather than black)

## rbf\_showMessage()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Displays a status message on the top of the page, which will automatically disappear after 40 seconds. This method implementation calls [rbf\\_growl\(\)](#) on page 679.

### Syntax

```
rbf_showMsg (message, type);
```

### Parameters

*message*

The message to display.

*type*

The message type, one of: `success`, `info`, `warn`, or `error`

### Return Value

?

### Example

Introduce example here.

?

## rbf\_showOrHidePageTab()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

### Purpose

Show or hide page-level tab with given index (only for pages where tabs are enabled).

### Syntax

```
rbf_showOrHidePageTab(tabIndex, showTab)
```

## Parameters

*tabIndex*

The 0-based index of page level tab to be activated

*showTab*

If `true`, show tab, if `false`, hide tab

## rbf\_showOrHideSection()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

This function hides or shows a Page's section. It will have no effect if the section does not exist.

The ID for any section can be found by selecting that section while editing the page in the Page Editor. Highlight the Page section by clicking on its header and use the "Original ID" parameter shown in the Properties box. You can also use [rbf\\_getSectionIdByTitle\(\)](#) on page 678.



## Syntax

```
rbf_showOrHideSection(sectionId, showSection)
```

## Parameters

*sectionId*

The original ID of page's section (can be found in **Page Editor**)

*showSection*

If `true`, show section, if `false`, hide section

## rbf\_setSectionCollapse()



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

## Purpose

This function collapses or expands the Page's section. It will have no effect if the section does not exist or is non-collapsible.

## Syntax

```
rbf_setSectionCollapse(sectionId, collapsed)
```

### Parameters

*sectionId*

Original ID of Page's section (can be found in Page Editor)

*collapsed*

If true, collapse section, if false, expand section

## rbf\_setViewSelector()

---



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

This function sets the view ID used by a lookup field for both auto-complete and pop-up selectors.

Important: This function cannot be used on dependent lookups. This function only apply for "Selector" style lookup fields.

## Syntax

```
rbf_setViewSelector(fieldName, selectorViewId)
```

### Parameters

*fieldName*

The integration name of lookup field

*selectorViewId*

ID of view to use

# Code Generator

To access Rollbase APIs from external computer systems, customers often need to write computer code that uses Object names, Field names, and some other customer-specific parameters. Writing such code can be a tedious and error-prone task, since it might require many copy-and-paste operations.

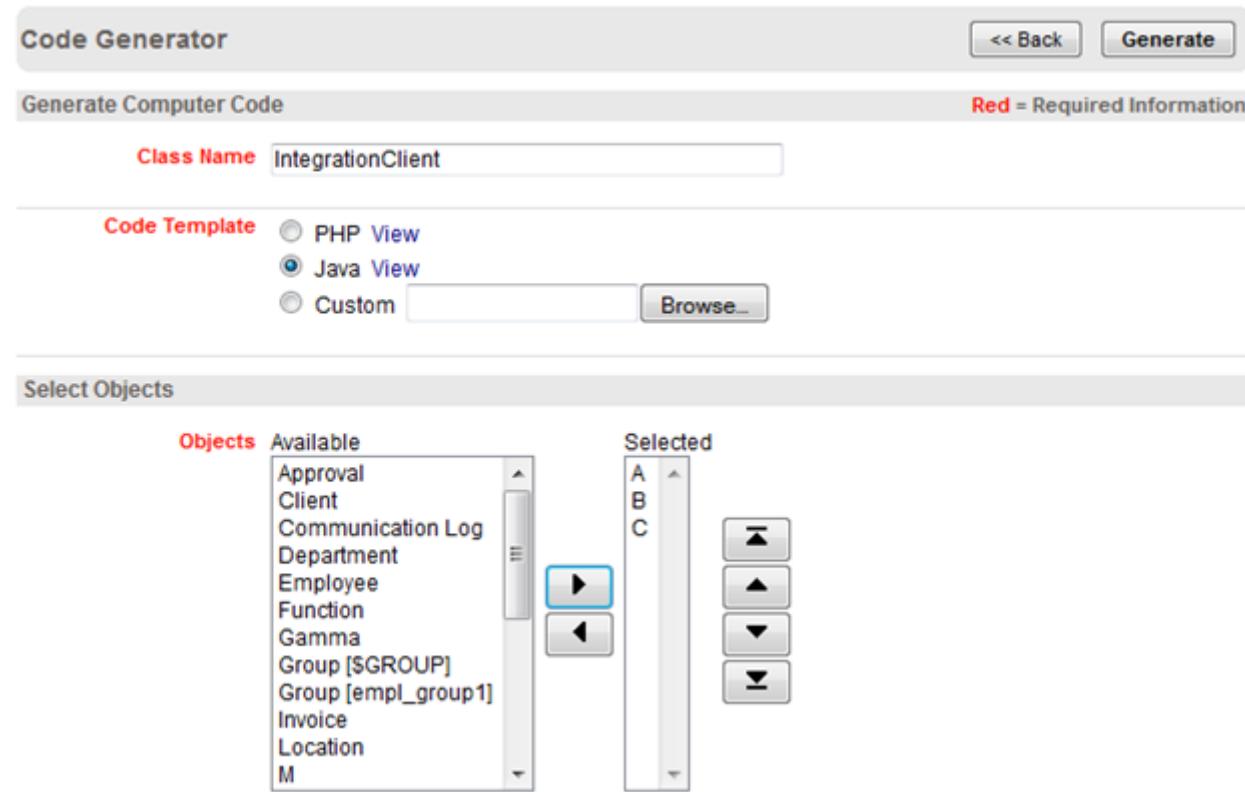
To simplify this task, Rollbase provides a Code Generator, available from the **Administration Setup** page. The Code Generator parses and fetches a text template that uses the metadata tokens described below.

Template Token	Description
{ !#SYSTEM_NAME }	Name of the Rollbase installation
{ !#HOST_NAME }	Host name used by the Rollbase installation
{ !#CURR_CUSTM.name }	Customer name
{ !#CURR_CUSTM.id }	Customer ID
{ !#CLASS_NAME }	Class name (free text input from the <b>Code Generator</b> page)
{ !#OBJ_LOOP_BEGIN }	Marks the beginning of a loop through selected Objects. Text within the loop is replicated for each selected Object.
{ !#OBJ_LOOP_END }	Marks the end of an Object loop.
{ !#object.name }	Integration name of an Object in a loop.
{ !#object.display }	Display name of an Object in a loop.
{ !#FIELD_LOOP_BEGIN }	Marks the beginning of loop through fields of the current Object. Text within the loop is replicated for each field (excluding read-only fields).
{ !#FIELD_LOOP_END }	Marks the end of a Fields loop.
{ !#field.name }	Integration name of a Field in a loop.
{ !#field.display }	Display name of a Field in a loop.

## Using the Code Generator

Follow these steps to use the Code Generator:

1. Provide the Class name to be used as a { !#CLASS\_NAME } token in your code.
2. Select the template:
  - a) Java client for the Rollbase REST API
  - b) PHP client for the Rollbase REST API
  - c) Custom template (upload)
3. Select Objects to be used in the generated code.



4. Click **Generate** to download the resulting text document.

## Metadata API and XML Reference

The SOAP and REST methods in this section provide a way to manipulate the following Rollbase metadata elements: application, object, field, and relationship definitions. These calls either generate output in XML format or take input in XML format. The XML schema for these APIs is available at <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

## Metadata XML Reference

The XML definition of a Rollbase application includes the application element and object, field, and relationship nodes as described in the following topics:

- [Application XML Elements](#) on page 688
- [Object XML Definition](#) on page 690
- [Field XML Definition](#) on page 695
- [Relationship XML Definition](#) on page 703

## Application XML Elements

The following sections describe the application-level XML elements that define application metadata:

- [Application Node](#) on page 688
- [ApplicationProps Node](#) on page 689
- [Application Definition XML Example](#) on page 690

### Application Node

Element Name	XML Element Type	Data Type	Description
DisplayName	Node	String	The application display name.
Description	Node	String	The application description.
Props	Node	Complex	The application properties. See the description for <a href="#">ApplicationProps</a> .
DependentDefs	Node	String	A set of comma-separated objects, which must be installed prior to installing this application.
DataObjectDefs	Node	Complex	A set of object definitions. See the description for <a href="#">DataObjectDef</a> .
id	Attr	int	The application ID.
origId	Attr	int	The original application ID.
orderNo	Attr	int	The order in which this application appears in the list of applications.
isSystem	Attr	Boolean	Specifies whether this is a system application.

Element Name	XML Element Type	Data Type	Description
version	Attr	int	The application version.
terminateOnError	Attr	Boolean	Specifies whether to terminate the application when an error occurs.

## ApplicationProps Node

The following elements exist in the `ApplicationProps` element, a child of the `Application` node.

Element Name	XML Element Type	Data Type	Description
pubManaged	Node	int	One of the following values: 0: unlocked (not managed), 1: fully locked, 2: partially locked
isDeployed	Node	boolean	If true, application is deployed and non-administrative users who have the appropriate permissions will be able to use the application. If false, only administrators will have access.
helpField	Node	Boolean	Specifies whether to enable help pop-ups for object fields in this application.
isDeployed	Node	boolean	If true, application is deployed and non-administrative users who have the appropriate permissions will be able to use the application. If false, only administrators will have access.
dependentDefs	Node	String	Comma-separated, installer-generated list of dependent objects.

Element Name	XML Element Type	Data Type	Description
isHidden	Node	Boolean	Specifies whether this application is hidden and does not appear in the drop-down list of applications.
useLegacyHeaderFooter	Node	Boolean	Specifies whether to use the legacy header and footer in the UI.
showMenus	Node	Boolean	Specifies whether to show second-level menus in a row underneath application tabs.
isMobile	Node	Boolean	Specifies whether to enable Mobile-Web functionality for this application.

## Application Definition XML Example

The following shows an XML example of an Application node:

```

<Application id="7558" origId="7558" orderNo="3" isSystem="false" version="1"
  >
  <DisplayName>ABC</DisplayName>
  <Props>
    <pubManaged>0</pubManaged>
    <helpField>false</helpField>
    <isDeployed>true</isDeployed>
    <dependentDefs>,</dependentDefs>
    <isHidden>false</isHidden>
    <useLegacyHeaderFooter>false</useLegacyHeaderFooter>
    <showMenus>false</showMenus>
  </Props>
  <dependentDefs>USER</dependentDefs>
  <DataObjectDefs>
    <!-- List of Objects -->
  </DataObjectDefs>
</Application>

```

## Object XML Definition

The following nodes and elements describe an object definition:

- [DataObjectDef Node](#) on page 691
- [ObjectProps Node](#) on page 692
- [Object Definition XML Example](#) on page 695

## DataObjectDef Node

The following table describes the `DataObjectDef` XML node.

Element Name	XML Element Type	Data Type	Description
SingularName	Node	String	The singular name for the object.
PluralName	Node	String	The plural name for the object.
NameTemplate	Node	String	The template to use when creating names for new objects of this type.
Description	Node	String	Description of this object.
Attributes	Node	String	A comma-separated list of object attributes. See the <code>fieldGroups.xml</code> file for attribute names.
Props	Node	Complex	A set of properties for this object. See the <code>ObjectProps</code> node description.
DataFieldDefs	Node	Complex	A set of fields for this object. See the <code>DataFieldDef</code> node description.
RelationshipDefs	Node	Complex	A set of relationships for this object. See the <code>RelationshipDef</code> node description.
id	Attr	int	The object ID.
origId	Attr	int	The original ID of the object.
objDefName	Attr	String	The object integration name (required).
isSystem	Attr	Boolean	Specifies whether this is a system object.
isAuditable	Attr	Boolean	Specifies whether audit is enabled for this object.

Element Name	XML Element Type	Data Type	Description
isViewable	Attr	Boolean	Specifies whether to track record views per user.
isFlaggable	Attr	Boolean	Specifies whether to track record flagging per user.
isDependent	Attr	Boolean	Specifies whether the object is a dependent object.
isDeployed	Attr	Boolean	Specifies whether the object is deployed.
addComponents	Attr	Boolean	Specifies whether the default components, such as pages and views will be added to new the object.
createTab	Attr	Boolean	Specifies whether a tab will be created for a new object.
attachApp	Attr	int	OriginalID of application to which the new object and tab will be attached.

## ObjectProps Node

The elements in the following table are children of the `ObjectProps` node.

Element Name	XML Element Type	Data Type	Description
auditView	Node	Boolean	Specifies whether to allow viewing of audit records.
auditCreate	Node	Boolean	Specifies whether to allow creation of audit records.
auditEdit	Node	Boolean	Specifies whether to allow edit of audit records.
auditDelete	Node	Boolean	Specifies whether to allow deletion of audit records.

Element Name	XML Element Type	Data Type	Description
dataSetName	Node	String	For OpenEdge objects, the data set name.
defProcess	Node	int	Specifies the ID of the default workflow process.
deleteFormula	Node	String	Specifies a formula to calculate delete permissions for this record.
editFormula	Node	String	Specifies a formula to calculate edit permissions for this record.
enableReports	Node	Boolean	Specifies whether to enable reports for this object.
googleSynch	Node	Boolean	For objects with an <code>Event</code> attribute, specifies whether to synchronize with the Google calendar.
isCloud	Node	Boolean	Specifies whether the object is an external object accessed through DataDirect Cloud.
isLegacy	Node	Boolean	Specifies whether this is an external object.
isManaged	Node	Boolean	Specifies whether this object is managed by the application publisher and cannot be modified.
isOpenEdge	Node	Boolean	Specifies whether the object is an OpenEdge ABL object.
isReadOnly	Node	Boolean	For external objects, specifies whether the object is read-only.

Element Name	XML Element Type	Data Type	Description
loginName	Node	String	Login name for OpenEdge and external objects that will be accessed through DataDirect Cloud.
pkColumn	Node	String	Primary key(s) for OpenEdge and external objects.
queryInsert	Node	String	A Base-64 encoded INSERT query for external objects.
queryDelete	Node	String	A Base-64 encoded DELETE query for external objects.
queryNewID	Node	String	A Base-64 encoded query to fetch a new record ID external objects.
querySelect	Node	String	A Base-64 encoded SELECT query for external objects.
queryUpdate	Node	String	A Base-64 encoded UPDATE query for external objects.
password	Node	String	The password for OpenEdge objects. This property is encrypted and cannot be set with an API call.
requireUserCreds	Node	Boolean	Specifies whether to require individual user credentials for OpenEdge objects.
resourceURI	Node	String	The URI to an OpenEdge resource.
showTags	Node	Boolean	Specifies whether to show <b>Search Tag</b> options.

## Object Definition XML Example

The following shows an example `DataObjectDef` node:

```

<DataObjectDef id="10359" origId="14184" objDefName="b" isSystem="no"
  isAuditable="no"
    isViewable="no" isFlaggable="no" isDependent="no" isDeployed="yes">
  <Props>
    <isManaged>false</isManaged>
    <defProcess>-1</defProcess>
    <enableReports>true</enableReports>
    <googMap>false</googMap>
    <auditView>false</auditView>
    <showTags>false</showTags>
    <googSynch>false</googSynch>
  </Props>
  <SingularName>B</SingularName>
  <PluralName>Bs</PluralName>
  <DataFieldDefs>
    <DataFieldDef id="19869" origId="19869" objName="b" fieldName="sumbur"
      dataName="FieldDummy"
        uiClass="FormulaField" isRequired="no" isReadOnly="yes" isTextIndexable="no"
        isSystem="no"
          isAuditable="no" maxLength="0">
      <DisplayLabel>Sumbur</DisplayLabel>
      <Props>
        <returnType>1</returnType>
        <decimalPlaces>0</decimalPlaces>
        <hideLabel>false</hideLabel>
        <formatIndex>0</formatIndex>
        <formulaB64>eyFpZH0rMg==</formulaB64>
      </Props>
    </DataFieldDef>
    <!-- More fields here -->
  </DataFieldDefs>
  <RelationshipDefs>
    <RelationshipDef id="19891" origId="19891" relName="R19877" objDef1="b"
      objDef2="a1"
        singularName1="B" pluralName1="Bs" singularName2="A" pluralName2="As"
        isMultiple1="yes"
        isMultiple2="yes" orphanControl1="no" orphanControl2="no" cloneRelated1="no"
          cloneRelated2="no" isSystem="no" isPkFk="no">
    </RelationshipDef>
  </RelationshipDefs>
</DataObjectDef>

```

## Field XML Definition

The following topics describe the XML elements that describe a field:

- [DataFieldDef Node](#) on page 696
- [FieldProps Node](#) on page 698
- [Field Definition Example](#) on page 702
- [ListItem Node](#) on page 702
- [List XML example](#) on page 703

## DataFieldDef Node

Element Name	XML Element Type	Data Type	Description
DisplayLabel	Node	String	The display label for this field.
Description	Node	String	Description of this field.
Props	Node	Complex	A set of field properties. See <a href="#">FieldProps Node</a> on page 698 for more information.
ValidationScript	Node	String	A Base-64 encoded formula to validate this field.
ListItems	Node	Complex	For SelectList and similar fields, a set of picklist items.
imageData	Node	String	For SharedImage fields only, specifies a Base-64 encoded image.
id	Attr	int	The ID for this field.
origID	Attr	int	The original ID of this field.
objDefName	Attr	String	The integration name of the object that contains this field.
columnName	Attr	String	For external objects only, the database column name. This value is required.
fieldName	Attr	String	The integration name of this field. This value is required and must be unique per object.
groupName	Attr	String	The group name for this field.
dataName	Attr	String	Name of the data type, which is ignored for new fields.

Element Name	XML Element Type	Data Type	Description
uiClass	Attr	String	Specifies the UI type for this field. Required value, one of: TextInput, TextArea, DateInput, DateTimeInput, IntInput, CurrencyInput, DecimalInput, CheckBox, URLInput, EmailInput, PhoneInput, SelectList, SelectListMultiple, RadioButtons, CheckBoxesGroup, AutoNumber, FileInput, ImageInput, SharedImage, FormulaField, ExpressionField, IntegrationLink, TemplateField.
isRequired	Attr	Boolean	Specifies whether a value must be entered for this field on all pages.
isReadOnly	Attr	Boolean	Specifies whether the field is read-only on all pages in which it appears.
isTextIndexable	Attr	Boolean	Specifies whether this field will be indexed for full-text search.
isAuditable	Attr	Boolean	Specifies whether audit changes will be tracked for this field.

Element Name	XML Element Type	Data Type	Description
maxLength	Attr	Int	Specifies the maximum number of characters allowed as input for text-based fields
addToPages	Attr	Boolean	Specifies whether an API call to add a field definition should also add the newly created field to object pages.

## FieldProps Node

The following table describes elements of a `FieldProps` node:

Element Name	XML Element Type	Data Type	Description
autoNumFormat	Node	String	For AutoNum fields, the format. For example, <code>PO# {YYYY}-{0000000}</code> will be converted to a value such as <code>PO# 2007-0000001</code>
autoNumStart	Node	String	For AutoNum fields, the number from which to start.
cols	Node	Int	For TextArea fields, the number of columns.
decimalPlaces	Node	Int	For DecimalInput fields, the number of decimal places.
defValue	Node	String	Specifies a default value to be displayed for this field when new records are created.
fileMaxSize	Node	Int	For FileInput fields, specifies the maximum size, in kilobytes, for uploaded files.
formatIndex	Node	Int	For CurrencyInput fields, specifies a currency format index.

Element Name	XML Element Type	Data Type	Description
formula	Node	String	For formula fields, specifies a Base-64 encoded formula.
hideLabel	Node	Boolean	Specifies whether to hide the field display label on user interface pages.
inputMask	Node	String	For <code>TextInput</code> fields, specifies an input mask.
inlineEdit	Node	Boolean	Specifies whether this field can be edited on object view and edit pages.
isLocalized	Node	Boolean	For text fields, specifies whether to support input in a different language.
isShared	Node	Boolean	For <code>SelectInput</code> fields, specifies whether the values can be shared with other fields that provide multiple choices.
isUnique	Node	Boolean	Specifies whether this field allows the same value in more than one record: a value of true, means duplicates will not be allowed.
isWarning	Node	Boolean	Specifies whether validation errors should be considered as warnings, which can be ignored by the user.
isvShare	Node	Boolean	For fields in an ISV account on the Master Server, specifies whether this field can be viewed and updated, if applicable, by ISV tenants.

Element Name	XML Element Type	Data Type	Description
linkTempl	Node	String	For IntegrationLink fields, a Base-64 encoded template for the integration link.
lowerLabel	Node	String	For CheckBox fields, specifies an additional label to render on the right side of the checkbox.
maxValue	Node	Int	For numeric fields, the maximum value allowed.
minValue	Node	Int	For numeric fields, the minimum value allowed.
publicAccess	Node	Boolean	For FileInput and ImageInput fields, specifies whether to allow public access to uploaded resource without checking permissions.
returnType	Node	Int	For FormulaField fields, specifies the return type. Available values are: <ul style="list-style-type: none"> <li>• decimal number</li> <li>• currency</li> <li>• integer</li> <li>• string</li> <li>• boolean</li> <li>• date</li> <li>• date/time (adjusted to user's time zone)</li> <li>• date/time (not adjusted to user's time zone)</li> </ul> .
rows	Node	Int	For TextArea fields, the number of rows.

Element Name	XML Element Type	Data Type	Description
separator	Node	String	For DecimalInput fields, use this as a separator.
shiftCurrDate	Node	Int	For DateInput fields, add this number of days to the current date.
size	Node	Int	Specifies the size in number of characters for an HTML input box
sortABC	Node	Boolean	For fields that allow selections, such as SelectList, specifies whether to sort picklist values in alphabetic order.
source	Node	String	For fields that allow selections, such as SelectList, specifies the source name for picklist values.
template	Node	String	For TemplateField fields, specifies a Base-64 encoded template.
useCurrDate	Node	Boolean	For DateInput fields, specifies whether to use the current date as the default value.
useFullDate	Node	Boolean	For DateInput fields, specifies whether to use the full text of the date on view pages.
useRichEditor	Node	Boolean	For TextArea fields, specifies whether a Rich Text Editor will be available to enter text.
valFalse	Node	String	Clears the value for TRUE (CheckBox fields, External Objects)

Element Name	XML Element Type	Data Type	Description
valTrue	Node	String	Sets the value to TRUE (CheckBox fields, External Objects)
vertical	Node	Boolean	For fields that have multiple lines, such as checkboxes and radio buttons: if true, align controls vertically; if false, align horizontally.
viewWidth	Node	String	The width for this field in a view, in pixels or %.

## Field Definition Example

The following example shows how to use a `DataFieldDef` node to specify a field definition.

```

<DataFieldDef objDefName="a1" columnName="STR0" fieldName="my_text2"
  dataName="FieldString" uiClass="TextInput" isRequired="yes" isReadOnly="no"
  isTextIndexable="yes" isSystem="no" isAuditable="yes" maxLength="0">
  <DisplayLabel>My Text</DisplayLabel>
  <Props>
    <isLocalized>false</isLocalized>
    <isUnique>true</isUnique>
  </Props>
</DataFieldDef>

```

## ListItem Node

Element Name	XML Element Type	Data Type	Description
id	Attr	int	The item ID.
origId	Attr	int	The original ID of the list item.
orderNo	Attr	int	Required sequential order in the list for this item.
source	Attr	String	Required name to distinguish this group of items.
name	Attr	String	Specifies the display name for this list (required).

Element Name	XML Element Type	Data Type	Description
code	Attr	String	Specifies the integration code for this list.
mainItemId	Attr	int	For dependent picklists, specifies the ID of the main item.
isDefault	Attr	Boolean	Specifies whether this list item is the default for new records.

## List XML example

The following example shows a `ListItem` node.

```
<ListItem id="20535" origId="20535" orderNo="5" source="20529"
          name="ZZZ" code="z" mainItemId="-1" isDefault="no"/>
```

## Relationship XML Definition

A relationship specifies the cardinality and behavior of two associated objects in a direction. The source of the relationship, such as `invoice`, is object 1 and the associated objects, such as `line item`, is object two. A corresponding relationship in the other direction would be defined with `line item` as object 1 and `invoice` as object 2. The following topics describe the XML elements that define a relationship:

- [RelationshipDef Node](#) on page 703
- [Example of a relationship definition](#) on page 705

### RelationshipDef Node

The following table describes the child elements of a `RelationshipDef` node:

Element Name	XML Element Type	Data Type	Description
id	Attr	int	The relationship ID.
origId	Attr	int	The original ID of the relationship
relName	Attr	String	The required integration name for this relationship.
objDef1	Attr	String	A required value that specifies the integration name of the first object.

Element Name	XML Element Type	Data Type	Description
objDef2	Attr	String	A required value that specifies the integration name of the second object.
singularName1	Attr	String	A required value that specifies the singular name for the first object.
singularName2	Attr	String	A required value that specifies the singular name for the second object.
pluralName1	Attr	String	A required value that specifies the plural name for the first object.
pluralName2	Attr	String	A required value that specifies the plural name for the second object.
fkobjDef	Attr	String	For external relationships, specifies the integration name of the foreign key object.
pkColumn	Attr	String	For external relationships, specifies the name of the database column that contains the Primary Key.
fkColumn	Attr	String	For external relationships, specifies name of the database column that contains the Foreign Key.
isMultiple1	Attr	Boolean	Specifies whether to allow multiple related records for the first object (Many to <i>N</i> ).
isMultiple2	Attr	Boolean	Specifies whether to allow multiple related records for the second object ( <i>N</i> to Many).

Element Name	XML Element Type	Data Type	Description
orphanControl1	Attr	Boolean	Specifies whether to control orphan records for the first object. For example, with a relationship between obj1 and obj2, setting this element to True will delete obj1 records when the related obj2 record is deleted.
orphanControl2	Attr	Boolean	Specifies whether to control orphan records for the second object. For example, with a relationship between obj1 and obj2, setting this element to True will delete obj2 records when the related obj1 record is deleted.
cloneRelated1	Attr	Boolean	Specifies whether to clone related records for the first object.
cloneRelated2	Attr	Boolean	Specifies whether to clone related records for the second object.
isSystem	Attr	Boolean	Specifies whether this relationship is a system relationship.
isPkFk	Attr	Boolean	For external objects, specifies whether this is a primary key-foreign key relationship.
addToPages	Attr	Boolean	Specifies whether a relationship created by an API call should be added in a new lookup field on user interface pages.

## Example of a relationship definition

The following example shows a RelationshipDef node:

```
<RelationshipDef id="19877" origId="19877" relName="R19877" objDef1="b"
objDef2="a1" singularName1="B" pluralName1="Bs" singularName2="A"
pluralName2="As" isMultiple1="yes" isMultiple2="yes" orphanControl1="no"
```

```
orphanControl2="no" cloneRelated1="no" cloneRelated2="no" isSystem="no"
isPkFk="no">
</RelationshipDef>
```

## SOAP Metadata Methods

This section describes the SOAP metadata methods.

### **createApplicationDef()**

#### **Purpose**

Creates a new application definition and its components from an XML document. This method will create any objects described in the document, create their corresponding tabs and menus, and attach both objects and menus to the newly created application.

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

#### **Syntax**

```
createApplicationDef(string sessionId, string xmlString);
```

#### **Parameters**

*sessionId*

A string containing the session ID obtained at log in.

*XMLString*

A string containing an XML application node description that includes any objects to be created.

#### **Output**

Status text:

Application ABC has been created

#### **Permissions Required**

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

#### **Example**

The following example passes in the XML required to create an application.

```
String infoMessage = binding.createApplicationDef(sessionId, xmlString);
```

## createObjectDef()

### Purpose

Creates a new object definition and its components from an XML document.

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

### Syntax

```
createObjectDef(string sessionId, string xmlString)
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the object in an `DataObjectDef` node, along with fields and relationships.

### Output

Status text: Object ABC has been created

### Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

### Example

This example shows how to call `createObjectDef()`:

```
String infoMessage = binding.createObjectDef(sessionId, xmlString);
```

## createFieldDef()

### Purpose

Creates a new field definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the `DataFieldDef` XML node and an example of its use.

### Syntax

```
createFieldDef(string sessionId, string xmlString)
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the field in an `DataFieldDef` node.

## Output

Status text: Field *ABC* has been created

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

This example shows how to call `createFieldDef()`:

```
String infoMessage = binding.createFieldDef(sessionId, xmlString);
```

# createRelationshipDef()

## Purpose

Creates a new relationship definition from an XML document.

See [Relationship XML Definition](#) on page 703 for a description of the `RelationshipDef` XML node and an example of its use.

## Syntax

```
createRelationshipDef(string sessionId, string xmlString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the field in an `RelationshipDef` node.

## Output

Status text: Relationship *R123456* has been created

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example creates a relationship definition:

```
String infoMessage = binding.createRelationshipDef(sessionId, xmlString);
```

# deleteApplicationDef

## Purpose

Permanently deletes an existing application and its components, including objects that are only assigned to this application and all corresponding records.

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

## Syntax

```
deleteApplicationDef(string sessionId, string appId);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*appId*

The original ID of the application to retrieve.

## Output

Status text:

Application ABC has been deleted

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following examples shows how to delete application 123456.

```
String infoMessage = binding.deleteApplicationDef(sessionId, 123456);
```

## deleteFieldDef()

### Purpose

Permanently deletes an existing field definition from the specified object, along with any existing values for that field.

See [Field XML Definition](#) on page 695 for a description of the `DataFieldDef` XML node and an example of its use.

### Syntax

```
deleteFieldDef(string sessionId, string objDefName, string fieldName);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing the integration name for the object containing the field

*fieldName*

String containing the integration name for the field definition to delete

### Output

Status text: Field ABC has been deleted

### Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example deletes the `firstName` field from `lead` objects:

```
String infoMessage = binding.deleteFieldDef(sessionId, "lead", "firstName");
```

## deleteObjectDef()

### Purpose

Permanently deletes an existing object definition, its components, and its records.

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

### Syntax

```
deleteObjectDef(string sessionId, string objDefName);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing the integration name for the object definition to delete

### Output

Status text: Object *ABC* has been deleted

### Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

### Example

The following example deletes the definition of a lead object, its records, and components.

```
String infoMessage = binding.deleteObjectDef(sessionId, "lead");
```

## deleteRelationshipDef()

### Purpose

Permanently deletes an existing relationship definition.

See [Field XML Definition](#) on page 695 for a description of the `RelationshipDef` XML node and an example of its use.

### Syntax

```
deleteObjectDef(string sessionId, string relName);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*relName*

String containing the integration name for the relationship definition to delete

## Output

Status text: Field *R123456* has been deleted

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example deletes a relationship definition:

```
String infoMessage = binding.deleteRelationshipDef(sessionId, "R123456");
```

# getApplicationDef

## Purpose

Retrieves a full description of a Rollbase application definition as an XML document. The XML schema for this document can be found at: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

## Syntax

```
getApplicationDef(string sessionId, long appId);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*appId*

The original ID of the application to retrieve.

## Output

XML document with the application definition in an Application node.

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example retrieves an XML description of MyApplication.

```
String xmlData = binding.getApplicationDef(sessionId, "MyApplication");
```

## getFieldDef()

### Purpose

Retrieves a description of a Field definition as an XML document in a sub-node of the DataObjectDef node.

See [Field XML Definition](#) on page 695 for a description of the DataFieldDef XML node and an example of its use.

### Syntax

```
getFieldDef(string sessionId, string objDefName, string fieldDefName);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing the integration name of the object definition that contains the field

*fieldDefName*

String containing the integration name of the field definition to retrieve

### Output

An XML document with the field definition in a DataFieldDef node.

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example retrieves the field definition of `firstName`:

```
String xmlData = binding.getFieldDef(sessionId, "lead", "firstName");
```

## getObjectDef()

### Purpose

Retrieves a full description of an object definition as an XML document. The XML schema for this document can be found at: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

### Syntax

```
getObjectDef(string sessionId, string objDefName);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the integration name for the object definition to retrieve

### Output

An XML document containing the object definition description in a `DataObjectDef` node.

### Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example retrieves the definition of a lead object.

```
String xmlData = binding.getObjectDef(sessionId, "lead");
```

## getObjectDefNames()

### Purpose

Retrieves an array of object definition integration names.

## Syntax

```
getObjectDefNames(string sessionId);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

## Output

Array of object definition integration names

## Example

```
StringArr arr = binding.getObjectDefNames(sessionId);
```

# getRelationshipDef()

## Purpose

Retrieves a description of a relationship definition as an XML document in a sub-node of the `DataObjectDef` node.

See [Relationship XML Definition](#) on page 703 for a description of the `RelationshipDef` XML node and an example of its use.

## Syntax

```
getRelationshipDef(string sessionId, string relName);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*relName*

String containing the integration name of the relationship definition to retrieve

## Output

An XML document with the relationship definition in a `RelationshipDef` node.

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example retrieves a relationship definition:

```
String xmlData = binding.getRelationshipDef(sessionId, "R123456");
```

## metadatasearch()

### Purpose

Searches for a string in metadata and returns results in XML format.

### Syntax

```
metadataSearch(string sessionId, string query);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

A text pattern to search.

### Output

An XML document with nodes for each metadata entity found for the search string.

### Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

```
String xmlResults = binding.metadataSearch(sessionId, "test");
```

## updateApplicationDef()

### Purpose

Updates an existing application definition and its components from an XML document. The XML document must include a valid original ID attribute of the root XML node.

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

## Syntax

```
updateApplicationDef(string sessionId, string xmlString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

XML description of the application in an Application node, including the objects to be updated

## Output

Status text:

```
Application ABC has been updated
```

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example shows how to invoke `updateApplicationDef`.

```
String infoMessage = binding.updateApplicationDef(sessionId, xmlString);
```

## updateFieldDef()

### Purpose

Updates an existing field definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the DataFieldDef XML node and an example of its use.

## Syntax

```
updateFieldDef(string sessionId, string xmlString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the field in an `DataFieldDef` node.

## Output

Status text: Field *ABC* has been updated

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

The following example shows how to invoke `updateFieldDef()`:

```
String infoMessage = binding.updateFieldDef(sessionId, xmlString);
```

# updateObjectDef()

## Purpose

Updates an existing object definition and its components from an XML document.

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

## Syntax

```
updateObjectDef(string sessionId, string xmlString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the object in an `DataObjectDef` node, along with fields and relationships.

## Output

Status text: Object *ABC* has been updated

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

### Example

The following example invokes `updateObjectDef()`:

```
String infoMessage = binding.updateObjectDef(sessionId, xmlString);
```

## updateRelationshipDef()

### Purpose

Updates an existing relationship definition from an XML document.

See [Relationship XML Definition](#) on page 703 for a description of the `RelationshipDef` XML node and an example of its use.

### Syntax

```
updateRelationshipDef(string sessionId, string xmlString);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*xmlString*

An XML string describing the field in an `RelationshipDef` node.

### Output

Status text: Relationship *R123456* has been updated

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

### Example

The following example updates a relationship definition:

```
String infoMessage = binding.updateRelationshipDef(sessionId, xmlString);
```

## REST Metadata Methods

This section describes the REST metadata methods.

### createApplicationDef

#### Purpose

Creates a new application definition and its components from an XML document. This API will create the objects corresponding tabs and menus, and attach both to the newly created application. See the XML schema for this document: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

#### HTTP Method

POST

#### URL

`https://www.rollbase.com/rest/api/createApplicationDef`

#### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the application in an `Application` node, including the objects to be created.

#### Permissions Required

Full administrative privileges

#### Response

Application ABC has been created

### createFieldDef

#### Purpose

Creates a new field definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the `DataFieldDef` XML node and an example of its use.

## HTTP Method

POST

## URL

`https://www.rollbase.com/rest/api/createFieldDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the field in a `DataFieldDef` node.

## Permissions Required

Full administrative permissions

## Response

Status text: Field *ABC* has been created

# createObjectDef

## Purpose

Creates a new object definition and its components from an XML document. See the XML schema for this document: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

## HTTP Method

POST

## URL

`https://www.rollbase.com/rest/api/createObjectDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

XML description of object in the `DataObjectDef` node, including the fields, and relationships to be created.

## Permissions Required

Full administrative permissions

## Response

"test" object definition has been created. (ID=1953742)

# createRelationshipDef

## Purpose

Creates a new relationship definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the `RelationshipDef` XML node and an example of its use.

## HTTP Method

POST

## URL

`https://www.rollbase.com/rest/api/createRelationshipDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the relationship in a `RelationshipDef` node.

## Permissions Required

Full administrative permissions

## Response

Status text: Relationship *R123456* has been created

# deleteApplicationDef

## Purpose

Permanently deletes an existing application and its components. Any objects assigned to this application, that are not assigned to other applications, and the corresponding records will also be deleted.

See [Application XML Elements](#) on page 688 for a description of the `Application` XML node and an example of its use.

## HTTP Method

DELETE or GET

## URL

`https://www.rollbase.com/rest/api/deleteApplicationDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*appId*

The original application ID.

## Permissions Required

Full administrative privileges

## Response

Application *ABC* has been deleted

# deleteFieldDef

## Purpose

Deletes a field definition from the specified object.

See [Field XML Definition](#) on page 695 for a description of the `DataFieldDef` XML node and an example of its use.

## HTTP Method

DELETE or GET

## URL

`https://www.rollbase.com/rest/api/deleteFieldDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*fieldName*

The field integration name.

### Permissions Required

Full administrative permissions

### Response

Status Text: Field ABC has been deleted

## deleteObjectDef

### Purpose

Permanently deletes an existing object definition, its components and its records.

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

### HTTP Method

DELETE or GET

### URL

`https://www.rollbase.com/rest/api/deleteObjectDef`

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

### Permissions Required

Full administrative permissions

### Response

Object ABC has been deleted

## deleteRelationshipDef

### Purpose

Deletes the relationship definition specified by the integration name.

See [Field XML Definition](#) on page 695 for a description of the `RelationshipDef` XML node and an example of its use.

## HTTP Method

DELETE or GET

## URL

<https://www.rollbase.com/rest/api/deleteRelationshipDef>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*relName*

The relationship integration name.

## Permissions Required

Full administrative permissions

## Response

Status text: Relationship R123456 has been deleted

# getApplicationDef

## Purpose

Retrieves the full description of a Rollbase application definition as an XML document. The XML schema for this document can be found at: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getApplicationDef>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*appId*

The original application ID.

## Required Permissions

One of: VIEW, CREATE, EDIT, DELETE (VIEW by default)

## Response

XML document with the application definition description in an `Application` node.

# getFieldDef

## Purpose

Retrieves a full description of a field definition as an XML document in the sub-node describing objects.

See [Field XML Definition](#) on page 695 for a description of the `DataFieldDef` XML node and an example of its use.

## HTTP Method

GET

## URL

`https://www.rollbase.com/rest/api/getFieldDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*fieldName*

The field integration name.

## Permissions Required

One of: VIEW, CREATE, EDIT, DELETE (VIEW by default)

## Response

An XML document with the field definition in a `DataFieldDef` node.

# getObjectDef

## Purpose

Retrieves the full description of an object definition as an XML document. See the XML schema for this document: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

## HTTP Method

GET

## URL

`https://www.rollbase.com/rest/api/getObjectDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

## Permissions Required

One of: VIEW, CREATE, EDIT, DELETE (VIEW by default)

## Response

An XML document with the object definition description in a `DataObjectDef` node

# getObjectDefNames

## Purpose

Retrieves a list of object definition names. Output is limited to objects for which the current user has permissions to view, create, edit, and delete.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getObjectDefNames>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*permissions*

An optional parameter specifying one of: `view`, `create`, `edit`, or `delete`

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Required Permissions

One of: VIEW, CREATE, EDIT, DELETE (VIEW by default)

## Response

Names of object definitions in XML or JSON.

## Example

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<resp status="ok">
<names>
<name>visitor</name>
<name>process</name>
<name>COMMLOG</name>
</names>
</resp>
```

# getRelationshipDef

## Purpose

Retrieves an XML document containing a full description of a relationship definition in a sub-node of the node describing an object.

See [Field XML Definition](#) on page 695 for a description of the `RelationshipDef` XML node and an example of its use.

## HTTP Method

GET

## URL

`https://www.rollbase.com/rest/api/getRelationshipDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*relName*

The relationship integration name.

## Permissions Required

One of: VIEW, CREATE, EDIT, DELETE (VIEW by default)

## Response

An XML document with a relationship definition in the `RelationshipDef` node.

# metadataSearch

## Purpose

Searches for a string in metadata and returns results in XML format.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/metadataSearch>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*query*

A text pattern to search.

## Response

An XML document with nodes for each metadata entity found for the search string.

## Permissions Required

Full administrative permissions.

## Example

Sample XML response:

```
<resp status="ok">
<SearchResults>
<SearchResult id="376768" name="AJAX API Test" type="Application"
updatedAt="2014-08-22T18:57:41Z" />
<SearchResult id="376879" name="AJAX Test" type="Page"
updatedAt="2014-08-25T21:16:27Z" />
<SearchResult id="376881" name="AJAX Test" type="Menu"
updatedAt="2014-08-22T18:58:20Z" />
<SearchResult id="376825" name="All Tests" type="View" objDefName="test"
updatedAt="2014-08-22T18:57:41Z" />
</SearchResults>
</resp>
```

## updateApplicationDef

### Purpose

Updates an existing application definition and its components from an XML document. The XML document must include a valid Original ID as an attribute to the root XML node for the application to be updated. See the XML description.

See [Application XML Elements](#) on page 688 for a description of the Application XML node and an example of its use.

### HTTP Method

PUT or POST

### URL

`https://www.rollbase.com/rest/api/createApplicationDef`

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the field in a DataFieldDef node.

### Permissions Required

Full administrative permissions

### Response

Application ABC has been updated

## updateFieldDef

### Purpose

Updates an existing field definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the DataFieldDef XML node and an example of its use.

### HTTP Method

PUT or POST

### URL

`https://www.rollbase.com/rest/api/updateFieldDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the field in a `DataFieldDef` node.

## Permissions Required

Full administrative permissions

## Response

Status text: Field *ABC* has been updated

# updateObjectDef

## Purpose

Updates an existing object definition and its components from an XML document. See the XML schema for this document: <https://www.rollbase.com/webapi/wsdl/metadata.xsd>

See [Object XML Definition](#) on page 690 for a description of the `DataObjectDef` XML node and an example of its use.

## HTTP Method

PUT or POST

## URL

`https://www.rollbase.com/rest/api/updateObjectDef`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the object in a `DataObjectDef` node, including fields and relationships.

## Permissions Required

Full administrative privileges

## Response

Object *ABC* has been updated

## updateRelationshipDef

### Purpose

Updates an existing Relationship definition from an XML document.

See [Field XML Definition](#) on page 695 for a description of the `RelationshipDef` XML node and an example of its use.

### HTTP Method

PUT or POST

### URL

`https://www.rollbase.com/rest/api/updateRelationshipDef`

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*xmlString*

An XML description of the relationship in a `RelationshipDef` node.

### Permissions Required

Full administrative permissions

### Response

Status text: Relationship *R123456* has been updated

# Rollbase REST Methods

The Rollbase REST API uses the same workflow mechanism as the standard Rollbase user interface. For instance, triggers designed to run on object record creation will run if a record is created through the REST API.

Each REST method is accessed through URLs of the form:

`https://www.rollbase.com/rest/api/{method_name}`

Parameters to methods can be supplied as URL parameters for GET calls or in the HTTP request for POST calls. An XML document containing the result of the call or an error message will be returned as a response.

Example of a GET call:

`https://www.rollbase.com/rest/api/login?loginName=JohnSmith&password=welcome`

A successful response:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<sessionId>feddec47ce58445381cf64c60992e1c9@614642</sessionId>
</resp>
```

An error response:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="fail">
<err>Login session is not found</err>
</resp>
```

## appXML

### Purpose

Returns a Rollbase application as an XML document.

### HTTP Method

GET

### URL

<https://www.rollbase.com/rest/api/appXML>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*appId*

ID of application used to generate XML.

### Permissions Required

Full Administrative privileges

### Response

Application XML

## bulkCreate

### Purpose

Creates records from a bulk CSV upload. The `output` parameter is optional.

### HTTP Method

POST

**URL**

<https://www.rollbase.com/rest/api/bulkCreate>

**URL Parameters**

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*mapId*

The original ID of a previously created **Data Import Map**.

*sync*

If `true`, process import synchronously and return IDs of created records. If `false` (default), process import asynchronously. The user will receive an email with results of import.

*Request Body*

The CSV data.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

**Permissions Required**

CREATE for object type

**Response**

Report on created record similar to report after CSV import

## bulkCreateOrUpdate



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

**Purpose**

Updates existing records or creates new records from a bulk CSV upload. The `output` parameter is optional.

**HTTP Method**

POST

**URL**

<https://www.rollbase.com/rest/api/bulkCreateOrUpdate>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*mapId*

The original ID of a previously created **Data Import Map**.

*[fieldId] | [triggerId]*

ID of unique field used to identify records to be updated or ID of a <uicontrol>Unique Fields Combination</uicontrol> trigger.

*sync*

If `true`, process import synchronously and return IDs of created records. If `false` (default), process import asynchronously. The user will receive an email with results of import.

*Request Body*

The CSV data.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

UPDATE or CREATE for object type

## Response

Report on created/updated record similar to report after CSV import.

# bulkDelete

## Purpose

Deletes (moves to Recycle Bin) a group of specified records. The `output` parameter is optional.

## HTTP Method

DELETE or GET

## URL

<https://www.rollbase.com/rest/api/bulkDelete>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*ids*

Comma-separated list of record IDs.

*objName*

The object integration name.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

DELETE for the specified records

## Response

Report on deleted record similar to report after CSVdelete.

# bulkUpdate



**Warning:** Support for using this method with external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection) is a **beta** feature. **Do not** use this method in production systems.

---

## Purpose

Updates records from a bulk CSV upload. The `output` parameter is optional.

## HTTP Method

PUT or POST

## URL

<https://www.rollbase.com/rest/api/bulkUpdate>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*mapId*

The original ID of a previously created **Data Import Map**.

*[fieldId] | [triggerId]*

ID of unique field used to identify records to be updated or ID of a <uicontrol>Unique Fields Combination</uicontrol> trigger.

*sync*

If `true`, process import synchronously and return IDs of created records. If `false` (default), process import asynchronously. The user will receive an email with results of import.

*Request Body*

The CSV data.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

UPDATE for object type

## Response

Report on updated record similar to report after CSV update

# clearDataObjectCache

## Purpose

Clears the cache of object data records on production servers. Progress recommends you use this method after heavy use of API operations. The `output` parameter is optional.

## HTTP Method

GET

## URL

`https://www.rollbase.com/rest/api/clearDataObjectCache`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

Full administrative permissions

## Response

Standard success or failure output.

# create

## Purpose

Creates a new record using data from an XML document. This method can be used to create a single record or a single record and related records.

---

**Note:** Do not use this method with PHP CURL. Instead use [create2](#)

---

## HTTP Method

POST

## URL

<https://www.rollbase.com/rest/api/create>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*requestBody*

XML document containing object name, "useIds" attribute (true or false - same meaning as in Web API), and Fields for the new record (see example below).

## Permissions Required

CREATE for object

## Response

The ID and integration name of the new record and a status of `ok` (see examples below).

## Example

Create input example:

```
<?xml version="1.0" encoding="utf-8" ?>
  <data objName="person" useIds="false">
    <Field name="club_member">false</Field>
    <Field name="lastName">Smith</Field>
    <Field name="status">Created</Field>
  </data>
```

Create output example (XML):

```
<resp status="ok">
  <data id="314452" objName="person" />
  <Msg>12 fields have been processed.</Msg>
</resp>
```

Create output example (JSON):

```
{
  id: 314452
  objName: "person"
  status: "ok"
}
```

To create related objects in the same call, include a "composite" XML node which wraps a set of "data" XML nodes as described above. A new record is created for each "data" XML node. These new nodes have relationships with the core record. Consider this example with dependent nodes:

```
<?xml version="1.0" encoding="utf-8" ?>
<data objName="person" useIds="false">
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
<composite>
<data objName="payment" useIds="false">
<Field name="amount">500.00</Field>
<Field name="paym_date">09/12/2011</Field>
</data>
<data objName="payment" useIds="false">
<Field name="amount">450.00</Field>
<Field name="paym_date">09/22/2011</Field>
</data>
</composite>
</data>
```

This XML creates one "person" record and two related "payment" records in one call. Output example:

```
<resp status="ok">
<data id="314452" objName="person" />
<Msg>12 fields have been processed. 2 related records have been created.</Msg>
</resp>
```

## createArr

### Purpose

Creates a group of new records using data from an XML document.

### HTTP Method

POST

### URL

<https://www.rollbase.com/rest/api/createArr>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*Request Body*

XML document containing object name, "useIds" attribute (true or false - same meaning as in the Web API), and Fields for new records wrapped in `<data>` XML nodes (see example below).

## Required Permissions

CREATE for object type(s)

## Response

The ID and integration name of the new record and a status of `ok` (see examples below).

## Example

Input example:

```
<?xml version="1.0" encoding="utf-8" ?>
<request>
<data objName="person" useIds="false">
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
</data>
<data objName="person" useIds="false">
<Field name="club_member">true</Field>
<Field name="lastName">Green</Field>
<Field name="status">Created</Field>
</data>
</request>
```

Output example (XML):

```
<resp status="ok">
<data id="314452" objName="person"/>
<data id="314453" objName="person"/>
</resp>
```

Output example (JSON):

```
{
  "ids": 314452,
  "objName": "person",
  "status": "ok"
},
{
  "ids": 314453,
  "objName": "person",
  "status": "ok"
}
```

```
}
```

## createCustomer

### Purpose

Creates a new customer record and starts the creation process using data from URL parameters. For private cloud users, the user credentials for the user sending this request must belong to the Master server. The URL parameter password can be used to set specific password for first administrative user, otherwise, a temporary password will be created and sent through email. The output parameter is optional.

### HTTP Method

POST

### URL

<https://www.rollbase.com/rest/api/createCustomer>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*useIds*

Boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise use integration codes and record names.

*field1; [field2; ...]*

Parameter's name - integration name of field to set; parameter's value - value of field to set. Same for other fields.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

### Permissions Required

CREATE for Customer records

### Response

ID of newly created record as a long wrapped in an XML document or a JSON string (see example below).

## Example

Single create input example:

```
companyName=API+Test&email=myemail@mycompany.com&lastName=Admin
&loginName=myemail@mycompany.com&timeZone=PST
&mailSender=noreply@mycompany.com&password=my_password
```

---

**Note:** The API call must not have any blank spaces in it.

---

Output example:

```
{
  id: 7941
  objName: "CUSTOMER"
  status: "ok"
}
```

# create2

## Purpose



**Warning:** This method is deprecated. Please change any existing code to use [createRecord](#) on page 743, which takes an object ID as a parameter.

---

Creates a new record using data from URL parameters. The names of the URL parameters used by this API are field integration names. If a field is not found, the system ignores the URL parameter. Field values must be formatted the same way as CSV imported values. For more information, see [Importing Data](#).

---

**Note:** Use this method with the PHP CURL package.

---

## HTTP Method

POST

## URL

<https://www.rollbase.com/rest/api/create2>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*useIds*

true or false.

*output*

The output format, one of: `xml` (default), `json` or `csv`.

*field1; [field2; ...]*

Parameter's name - integration name of field to set; parameter's value - value of field to set.

## Permissions Required

CREATE for object

## Response

The ID and integration name of the new record wrapped in an XML document (see example below).

## Example

Input example (URL parameters):

```
&objDefName=person&useIds=false&club_member=false&lastName=Smith&status=Created
```

Output example:

```
<resp status="ok">
<data id="314452" objName="person" />
<Msg>12 fields have been processed</Msg>
</resp>
```

# createRecord

## Purpose

Similar to [create](#) on page 738, creates a new record. Returns the ID of the newly created record as a string. This method works for external objects, including those mapped to OpenEdge service objects.

Provide the integration names of the fields to set along with their values. If a field is not found, the system will ignore that URL parameter. Fields values must be formatted the same way as you would for a CVS import, see [Importing Data](#) on page 359.

## HTTP Method

POST

## URL

`https://www.rollbase.com/rest/api/createRecord`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*useIds*

Boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise use integration codes and record names.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*field1, field2, ...*

Integration name of the field(s) and their values. You must supply a value for required fields. Some fields, such as those that represent relationships, can have multiple values. Use the field name only once, and specify the ID of the records to attach separated with a `|` symbol.

## Permissions Required

CREATE for the object type.

## Response

The ID of the newly created record as a long value and a status of `ok`, wrapped in an XML document or JSON string.

## Example

Input example (URL parameters):

```
&objDefName=person&useIds=false&club_member=false&lastName=Smith&status=Created
```

Output example (XML):

```
<resp status="ok">
<data id="314452" objName="person" />
<Msg>12 fields have been processed</Msg>
</resp>
```

Output example (JSON):

```
{
  id: 314452
  objName: "person"
  status: "ok"
}
```

# delete

## Purpose

Moves the specified data record to the Recycle Bin. The `output` parameter is optional.

## HTTP Method

DELETE or GET

## URL

<https://www.rollbase.com/rest/api/delete>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*id*

Record's ID.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

DELETE for object

## Response

Standard success or failure output

## Example

Output example:

```
<resp status="ok">
<Msg>Record has been deleted</Msg>
</resp>
```

# deleteArr

## Purpose

Moves group of specified data record to the Recycle Bin. The `output` parameter is optional.

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you must specify both the `objName` and `id` fields.

## HTTP Method

DELETE or GET

## URL

<https://www.rollbase.com/rest/api/deleteArr>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*ids*

Comma-separated list of record IDs.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

DELETE for object

## Response

Standard success or failure output

## Example

Output example:

```
<resp status="ok">
<Msg>2 records have been deleted.</Msg>
</resp>
```

# deleteRecord

## Purpose

Moves Rollbase object records to the Recycle bin. Deletes external objects, including those mapped to OpenEdge Service objects.

## HTTP Method

DELETE or GET

## URL

`https://www.rollbase.com/rest/api/deleteRecord`

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*id*

The record ID.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

DELETE permission for the requested record.

## Response

Standard success or failure output.

## Example

Output example in XML format:

```
<resp status="ok">
<Msg>Record has been deleted.</Msg>
</resp>
```

# getApplicationIds

## Purpose

Retrieves IDs of all the applications available for the currently logged user.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getApplicationIds>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Response

IDs of all the applications available for the currently logged user.

## Example

Sample Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<resp status="ok">
<ids>
<id>123456</id>
<id>123490</id>
</ids>
</resp>
```

# getBinaryData

## Purpose

Retrieves the file attachment associated with a particular file field from a specific record.

## HTTP Method

GET method

## URL

<https://www.rollbase.com/rest/api/getBinaryData>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*id*

A long value containing the record ID.

*fieldName*

A string containing the file field integration name.

## Permissions Required

EDIT permission for the requested record.

## Response

The file attachment from the specified file field. The content type and length of output will be set to correspond to the attachment.

# getBuildStatus

## Purpose

Retrieves status of current Rollbase build and license info in XML format.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getBuildStatus>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Response

Build and license information in XML format.

## Example

Sample Response:

```
<?xml version="1.0" encoding="UTF-8" ?>
<resp status="ok">
<status>
<releaseNo>3.7.4</releaseNo>
<releaseDate>2012-08-16T</releaseDate>
<edition>Multi-Tenant</edition>
<expiration>2013-01-30T</expiration>
<host>localhost:8080</host>
</status>
</resp>
```

# getCodeById

## Purpose

Retrieves the integration code of a picklist item or a status by providing its ID.

---

**Note:** This API only works when the ID is numeric. Therefore, Picklist(char) or Radio Button(char) are not supported.

---

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getCodeById>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*objName*

The object integration name.

*field*

The field integration name.

*id*

Numeric ID of picklist item or status.

## Permissions Required

VIEW for requested object

## Response

Integration code of item with matching ID or null

## Example

Output example in JSON:

```
{ "code": "CREATED" }
```

# getCount

## Purpose

Returns the total number of records in a View (see [Views and Search](#) for more information about Views).

## HTTP Method

GET

**URL**

<https://www.rollbase.com/rest/api/getCount>

**URL Parameters**

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*viewId*

The original view ID.

*filterName*

Name of the field to filter output using the "equals" option (replaces the filter set in the View, if any).

*filterValue*

Value of the field to filter output using the "equals" option.

**Permissions Required**

VIEW for requested object

**Response**

Total number of records in View

**Example**

Output example in XML:

```
<resp status="ok">
<count>100</count>
</resp>
```

Output example in JSON:

```
{ "count": 100 }
```

## getDataField

**Purpose**

Retrieves the value of a single field from a specific record.

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you must specify both the `objName` and `id` fields.

## HTTP Method

GET

### URL

<https://www.rollbase.com/rest/api/getDataField>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*objName*

The object integration name.

*id*

The record ID.

*fieldName*

The field integration name.

*useLegacyDateFormat*

This optional URL parameter only applies to JSON output and, specifically, to DATE fields. If `useLegacyDateFormat=False`, or is not specified, dates will be returned in the normal JSON format, such as: `"Wed Mar 05 17:52:38 IST 2014"`. If `useLegacyDateFormat=True`, dates will be returned in a format similar to the following example: `new Date("Wed Mar 05 2014 17:53:33 (IST)")`. The following example shows results in normal and legacy formats:

- Normal:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": "Wed Mar 05 17:52:38 IST 2014"  
}
```

- Legacy:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": new Date("Wed Mar 05 2014 17:53:33 (IST)")  
}
```

## Permissions Required

VIEW for requested object

## Response

Field's value.

## Example

Output example in XML:

```
<?xml version="1.0" encoding="utf-8" ?>
  <resp status="ok">
    <Field name="status">Created</Field>
  </resp>
```

Output example in JSON:

```
{ "status": "Created" }
```

# getDataObj

## Purpose



**Warning:** This method is deprecated. Please change any existing code to use [getRecord](#) on page 757, which takes an object ID as a parameter.

---

Retrieves all field data (including formulas) for a given record in XML format.

The composite output is generated recursively in a tree-type structure. The following rules apply:

- The maximum level of recursion is provided as a URL parameter.
- Each data record is included in the output only once.
- No more than 1000 total data records are included in the output.
- The `objNames` input parameter can be used to explicitly set the list of output data objects.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getDataObj>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*id*

The record ID.

*composite*

The number of levels for related records to be included recursively in the output, defaults to 0.

*objNames*

A comma-separated list of object names. If present, only objects from this list are included in the output.

*fieldList*

A comma-separated list of field names. If present, only fields from this list are included in the output.

## Permissions Required

VIEW for requested object

## Response

All of the field data for the specified object record.

## Example

Output example for core record:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<data id="131227" objName="person">
<Field name="id">131227</Field>
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
<Field name="updatedAt">20110111T143331Z</Field>
</data>
</resp>
```

Output example for core and related record (composite parameter set to 1):

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<data id="131227" objName="person">
<Field name="id">131227</Field>
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
<Field name="updatedAt">20110111T143331Z</Field>
<Field name="R986">131228,131229</Field>
<composite>
<data id="131228" objName="payment">
<Field name="amount">500.00</Field>
<Field name="paym_date">20110115T143331Z</Field>
<Field name="R986">131227</Field>
</data>
<data id="131229" objName="payment">
<Field name="amount">450.00</Field>
<Field name="paym_date">20110116T143331Z</Field>
<Field name="R986">131227</Field>
</data>
</composite>
</data>
</resp>
```

# getIdByCode

## Purpose

Retrieves the ID of a picklist item or status by providing its integration code.

---

**Note:** This API only works when the ID is numeric. Therefore, Picklist(char) or Radio Button(char) are not supported.

---

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getIdByCode>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*objName*

The object integration name.

*field*

The field integration name.

*code*

Integration name of picklist item or status.

## Permissions Required

VIEW for requested object

## Response

ID of item with matching integration code or -1

## Example

Output example in JSON:

```
{ "id": 4567 }
```

# getPage

## Purpose

Retrieves the specified range of data records in a view. The selected view defines the sorting and filtering of output. The amount of processing and time required to get complete results can vary widely, depending on whether it fetches related records and the number of rows you specify per page.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getPage>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*viewId*

The original view ID.

*startRow*

The number of the row from which to start output. Defaults to 0

*rowsPerPage*

The number of data records in output. Defaults to the user-specified number of records per page.

*composite*

The number of levels for related records to be included recursively in the output, defaults to 0.

*objNames*

A comma-separated list of object names. If present, only objects from this list are included in the output.

*fieldList*

A comma-separated list of field names. If present, only fields from this list are included in the output.

*filterName*

Name of the field to filter output using the “equals” option (replaces the filter set in the View, if any).

*filterValue*

Value of the field to filter output using the "equals" option.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

VIEW for requested object

## Response

Set of data records in View in requested range

## Example

Output example in XML format (no recursive output for related records):

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<data id="131227" objName="person">
<Field name="firstName">Bill</Field>
<Field name="lastName">Smith</Field>
</data>
<data id="131228" objName="person">
<Field name="firstName">John</Field>
<Field name="lastName">Roth</Field>
</data>
</resp>
```

Output example in JSON format (no recursive output for related records):

```
[
{"objName": "lead", "id": 131227, "firstName": "Bill", "lastName": "Smith" },
 {"objName": "lead", "id": 131228, "firstName": "John", "lastName": "Roth" }
]
```

# getRecord

## Purpose

Retrieves all field data (including formulas) for a given record in XML format.

The composite output is generated recursively in a tree-type structure. The following rules apply:

- The maximum level of recursion is provided as a URL parameter.
- Each data record is included in the output only once.
- No more than 1000 total data records are included in the output.
- The `objNames` input parameter can be used to explicitly set the list of output data objects.

## HTTP Method

GET

**URL**

<https://www.rollbase.com/rest/api/getRecord>

**URL Parameters**

*objName*

The object integration name.

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*id*

The record ID.

*composite*

The number of levels for related records to be included recursively in the output, defaults to 0.

*objNames*

A comma-separated list of object names. If present, only objects from this list are included in the output.

*fieldList*

A comma-separated list of field names. If present, only fields from this list are included in the output.

**Permissions Required**

VIEW for the requested object

**Response**

All of the field data for the specified object record and related records.

**Example**

Output example for core record:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<data id="131227" objName="person">
<Field name="id">131227</Field>
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
<Field name="updatedAt">20110111T143331Z</Field>
</data>
</resp>
```

Output example for core and related record (composite parameter set to 1):

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<data id="131227" objName="person">
<Field name="id">131227</Field>
<Field name="club_member">false</Field>
<Field name="lastName">Smith</Field>
<Field name="status">Created</Field>
<Field name="updatedAt">20110111T143331Z</Field>
<Field name="R986">131228,131229</Field>
<composite>
<data id="131228" objName="payment">
<Field name="amount">500.00</Field>
<Field name="paym_date">20110115T143331Z</Field>
<Field name="R986">131227</Field>
</data>
<data id="131229" objName="payment">
<Field name="amount">450.00</Field>
<Field name="paym_date">20110116T143331Z</Field>
<Field name="R986">131227</Field>
</data>
</composite>
</data>
</resp>
```

## getRelationships

### Purpose

Retrieves an array of related record IDs. If you supply the optional *fieldList* parameter is supplied, output only includes values for those fields.

### HTTP Method

GET

### URL

<https://www.rollbase.com/rest/api/getRelationships>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*objName*

The object integration name.

*id*

The record ID.

*relName*

The relationship integration name.

*fieldList*

A comma-separated list of field names. If present, only fields from this list are included in the output.

## Permissions Required

VIEW for requested object

## Response

IDs of related records

## Example

Output example (XML):

```
<resp status="ok">
<ids>
<id>314452</id>
<id>128003</id>
</ids>
</resp>
```

Output example (JSON):

```
[ 314452, 128003 ]
```

# getUpdated

## Purpose

Retrieves an array of record IDs of the specified object type that were either created or updated within the given date/time interval. This method is not available for external objects, including those mapped to OpenEdge service objects.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/getUpdated>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*From*

Beginning of date/time interval in ISO 8601 format.

*Till*

End of date/time interval in ISO 8601 format.

*objName*

The object integration name.

## Permissions Required

VIEW for requested object

## Response

IDs of all records found in the search, returned as an array of longs

## Example

Output example in XML:

```
<resp status="ok">
<ids>
<id>314452</id>
<id>128003</id>
</ids>
</resp>
```

Output example in JSON:

```
[ 314452, 128003 ]
```

# header

## Purpose



**Warning:** This method is deprecated.

---

# install

## Purpose

Installs or updates Rollbase application from an XML document. Invoking this API will cause a customer reload. All currently logged users will be forced to log out.

## HTTP Method

GET

**URL**

<https://www.rollbase.com/rest/api/install>

**URL Parameters**

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*overrideChanges*

A value of `true` or `false`: if `true` (default value), override changes made by administrative users.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

Request Body

Application XML.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

**Permissions Required**

Full Administrative permissions

**Response**

Short report on application installation/update

**Example**

Output example:

```
<resp status="ok">
<Report>Application "Test" has been installed.</Report>
</resp>
```

# installByAppId

**Purpose**

Installs or updates a Rollbase application published to the **Application Directory**. Invoking this API will cause customer reload. All currently logged in users will be forced to log out.

**HTTP Method**

POST

**URL**

<https://www.rollbase.com/rest/api/installByAppId>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*id*

ID of package record created for published application in Application Directory,

*overrideChanges*

A value of `true` or `false`: if `true` (default value), override changes made by administrative users.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

Full Administrative privileges

## Response

Short report on application installation/update

## Example

Output example:

```
<resp status="ok">
<Report>Application "Test" has been installed.</Report>
</resp>
```

# login

## Purpose

Performs a user log in and initiates an API session. This method must be called prior to any other API call. The REST `login` creates a server-side session that may expire according to the security level selected for a Customer (see [Security and Access Control](#) for more info). The API client should log in again if the session expires. Progress recommends that an API client logs out after performing group of operations rather than keeping the session open for a long time. When an API client logs in, any previous sessions existing for the same user credentials are terminated.

Master Server users with login permissions can use the `login` API with their credentials to access the REST API for a specified Customer.

## HTTP Method

POST or GET

## URL

<https://www.rollbase.com/rest/api/login>

## URL Parameters

*loginName*

Login name for an active Rollbase user account

*password*

User password

*custId*

ID of Customer to log in. Optional parameter: if not present, the Customer is determined by login name.

*output*

"xml" (default) or "json"

## Response

Session ID that must be used in all subsequent API calls during this session

## Example

Sample XML response:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<sessionId> ecc701d2442e4f6b9fe2a7cc7b52078a@5857</sessionId>
</resp>
```

Sample JSON response:

```
{"status": "ok", "sessionId": "ecc701d2442e4f6b9fe2a7cc7b52078a@5857" }
```

# logout

## Purpose

Terminates the specified API session. This method should be called to explicitly end each Web API session. The *output* parameter is optional.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/logout>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Response

Standard success or failure output

## Example

Sample XML response:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
</resp>
```

# runTrigger

## Purpose

Runs a trigger on an object.

## HTTP Method

PUT or POST

## URL

<https://www.rollbase.com/rest/api/runTrigger>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

Optional integration name to narrow the search to a particular object.

*id*

The record ID.

*triggerid*

Integration name or the original ID of the trigger.

*checkValidation*

If set to `true`, checks validation formula before running the trigger. This is set to `false` by default.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Response

Standard success or failure output.

## Permissions

UPDATE permissions for the requested record.

## Example

Input example (URL parameters):

```
&objName=person&id=314452&triggerId=TR1
```

Output example (in XML format):

```
<?xml version="1.0" encoding="UTF-8"?>
<resp status="ok"> </resp>
```

# search

## Purpose

Performs a full-text search in the database assigned to the account used to obtain the session ID. The search query has the same syntax as that used in the standard Rollbase interface. See [Views and Search](#) for more information on search query syntax.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/search>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*query*

A URL-encoded SQL query that adheres to the same syntax and restrictions as the server-side Query API.

*objName*

Optional integration name to narrow the search to a particular object.

## Response

IDs of all records found in the search, returned as an array of longs

## Example

Output example (XML):

```
<resp status="ok">
<ids>
<id>314452</id>
<id>128003</id>
</ids>
</resp>
```

Output example (JSON):

```
[ 314452, 128003 ]
```

# selectNumber

## Purpose

Runs an SQL SELECT query on the server and returns the first element of the results as a decimal number.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/selectNumber>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*query*

A text pattern to search.

## Permissions Required

VIEW for requested object

## Response

Query result as decimal number

## Example

Output: first element returned by query converted into number and wrapped in XML. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<resp status="ok">
<value>53</value>
</resp>
```

Output example in JSON:

```
{ "value": 53.0 }
```

# selectQuery

## Purpose

Runs an SQL SELECT query on the server and returns the results as a 2D-array.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/selectQuery>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*startRow*

The number of the row from which to start output. Defaults to 0

*maxRows*

The maximum number of rows in output, which can be configured per Rollbase instance.

*query*

A text pattern to search.

*output*

The output format, one of: `xml` (default), `json` or `csv`.

*useLegacyDateFormat*

This optional URL parameter only applies to JSON output and, specifically, to DATE fields. If `useLegacyDateFormat=False`, or is not specified, dates will be returned in the normal JSON format, such as: "Wed Mar 05 17:52:38 IST 2014". If `useLegacyDateFormat=True`, dates will be returned in a format similar to the following

**example:** new Date("Wed Mar 05 2014 17:53:33 (IST)").The following example shows results in normal and legacy formats:

- Normal:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": "Wed Mar 05 17:52:38 IST 2014"  
}
```

- Legacy:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": new Date("Wed Mar 05 2014 17:53:33 (IST)")  
}
```

## Permissions Required

VIEW for requested object

## Response

Query results as 2D array

## Example

Query to run:

```
select id, firstName, lastName from lead order by updatedAt desc  
startRow=0  
maxRows=2
```

Output example in XML:

```
<?xml version="1.0" encoding="utf-8" ?>  
<resp status="ok">  
<row>  
<col>131227</col>  
<col>Bill</col>  
<col>Smith</col>  
</row>  
<row>  
<col>131228</col>  
<col>John</col>  
<col>Roth</col>  
</row></resp>
```

Output example in JSON:

```
[  
  [ 131227, "Bill", "Smith" ],  
  [ 131228, "John", "Roth" ]  
]
```

# selectValue

## Purpose

Runs an SQL SELECT query on the server and returns the first element from the results as single value.

## HTTP Method

GET

## URL

<https://www.rollbase.com/rest/api/selectValue>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*query*

A text pattern to search.

*useLegacyDateFormat*

This optional URL parameter only applies to JSON output and, specifically, to DATE fields. If `useLegacyDateFormat=False`, or is not specified, dates will be returned in the normal JSON format, such as: "Wed Mar 05 17:52:38 IST 2014". If `useLegacyDateFormat=True`, dates will be returned in a format similar to the following example: `new Date("Wed Mar 05 2014 17:53:33 (IST)")`. The following example shows results in normal and legacy formats:

- Normal:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": "Wed Mar 05 17:52:38 IST 2014"  
}
```

- Legacy:

```
{  
  "objName": "CUSTOMER",  
  "createdAt": new Date("Wed Mar 05 2014 17:53:33 (IST)")  
}
```

## Permissions Required

VIEW for requested object

## Response

Query result as single value

### Example

Example of first element returned by query wrapped in XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<resp status="ok">
<value>53</value>
</resp>
```

Output example in JSON:

```
{ "value": 53.0 }
```

# setBinaryData

## Purpose

Sets the value of a single field for a specific record.

## HTTP Method

GET or POST

## URL

<https://www.rollbase.com/rest/api/setBinaryData>

## URL Parameters

*id*

Record's id.

*fieldName*

The field integration name.

*value*

Value to be set. Must be formatted the same way as values in spreadsheet imports. For File Upload fields: Base-64 encoded binary value of file.

*contentType*

For File Upload fields only: MIME content type of uploaded data .

*fileName*

For File Upload fields only: original name of file being uploaded.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

## Permissions Required

EDIT for requested object

## Response

Status message wrapped in XML

# setDataField

## Purpose

Sets the value of a single field for a specific record.

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you must specify both the `objName` and `id` fields.

## HTTP Method

GET or POST

## URL

<https://www.rollbase.com/rest/api/setDataField>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*id*

The record ID.

*fieldName*

The field integration name.

*value*

Value to be set. Must be formatted the same way as values in spreadsheet imports. For File Upload fields: Base-64 encoded binary value of file.

*contentType*

For File Upload fields only: MIME content type of uploaded data.

*fileName*

For File Upload fields only: original name of file being uploaded.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

EDIT for requested object

## Response

Status message wrapped in XML

## Example

Output example:

```
<?xml version="1.0" encoding="utf-8" ?>
<resp status="ok">
<Msg>Field "amount" has been updated</Msg>
</resp>
```

# update

## Purpose

Updates an existing data record and, optionally, related records from values passed in as an XML document.

---

**Note:** Do not use this method with PHP CURL. Instead use `update2`

---

## HTTP Method

PUT or POST

## URL

<https://www.rollbase.com/rest/api/update>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

Request Body

XML document valid record ID, "useIds" attribute (true or false - same meaning as in the Web API) and data Fields to be updated (see example below). You only need to include the Fields that should be changed; Fields not included will remain unchanged.

## Permissions Required

UPDATE for object

## Response

Standard success or failure output.

## Example

Input example:

```
<?xml version="1.0" encoding="utf-8" ?>
  <data objName="person" id="314452" useIds="false">
    <Field name="club_member">false</Field>
    <Field name="lastName">Smith</Field>
    <Field name="status">Created</Field>
  </data>
```

Output example:

```
<resp status="ok">
  <Msg>12 fields have been processed.</Msg>
</resp>
```

Like create, update can be used to update several related records in one call. To do that, include a composite XML node that wraps a set of data XML nodes with valid id attributes as described above. Records corresponding to each "data" XML node are updated. These updated nodes will have a relationship with the core record.

Consider this example with dependent nodes:

```
<?xml version="1.0" encoding="utf-8" ?>
<data objName="person" id="314452" useIds="false">
  <Field name="club_member">false</Field>
  <Field name="lastName">Smith</Field>
  <Field name="status">Created</Field>
  <composite>
    <data objName="payment" id="314453" useIds="false">
      <Field name="amount">500.00</Field>
      <Field name="paym_date">09/12/2011</Field>
    </data>
    <data objName="payment" id="314454" useIds="false">
      <Field name="amount">450.00</Field>
      <Field name="paym_date">09/22/2011</Field>
    </data>
  </composite>
</data>
```

This XML updates one existing record and two related records in one call. Output example:

```
<resp status="ok">
<data id="314452"/>
```

```
<Msg>12 fields have been processed. 2 related records have been updated.</Msg>
</resp>
```

## updateArr

### Purpose

Updates a group of records using data from an XML document.

### HTTP Method

POST

### URL

<https://www.rollbase.com/rest/api/updateArr>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

*Request Body*

An XML document with valid `id` and `useIds` attributes (`true` or `false`) and data Fields to be updated wrapped in `<data>` XML nodes (see example below). You only need to include the Fields that should be changed; all Fields not included will remain unchanged.

### Required Permissions

UPDATE for object type(s)

### Response

Standard success or failure output.

### Example

Input example:

```
<?xml version="1.0" encoding="utf-8" ?>
<request>
<data id="314452" useIds="false">
<Field name="club_member">true</Field>
</data>
<data id="314453" useIds="false">
<Field name="lastName">Gray</Field>
```

```
</data>
<request>
```

Output example:

```
<resp status="ok">
<Msg>2 records have been updated.</Msg>
</resp>
```

## updateCustomer

### Purpose

Updates a customer record using data from the URL parameters. For private cloud users, the user credentials for the user making this request must be stored on the Master server. The `output` parameter is optional.

### HTTP Method

POST

### URL

<https://www.rollbase.com/rest/api/updateCustomer>

### URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*id*

The ID of the Customer record.

*useIds*

Boolean value: if true, use numeric IDs for lookup and picklist values, otherwise use integration codes and record names.

*field1; [field2; ...]*

Parameter's name - integration name of field to set; parameter's value - value of field to set. Same for other fields.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

### Permissions Required

UPDATE for Customer records

### Response

None

## Example

Single create input example:

```
id=8330&companyName=API+Test+2&timeZone=EST
```

# updateRecord

## Purpose

Updates an existing data record from URL parameters.

Provide the integration names of the fields to set along with their values. If a field is not found, the system will ignore that URL parameter. Fields values must be formatted the same way as you would for a CVS import, see [Importing Data](#) on page 359.

## HTTP Method

PUT or POST

## URL

```
https://www.rollbase.com/rest/api/updateRecord
```

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*objName*

The object integration name.

*useIds*

Boolean value: if true, use numeric IDs for lookup and picklist values; otherwise use integration codes and record names.

*field1, field2, ...*

Integration name of the field(s) and their values. You must supply a value for required fields. Some fields, such as those that represent relationships, can have multiple values. Use the field name only once, and specify the ID of the records to attach separated with a | symbol.

*output*

Optional parameter specifying the output format, one of: `xml` (default) or `json`.

## Permissions Required

UPDATE for the requested record.

## Response

Standard success or failure output.

## Example

Input example (URL parameters):

```
&objName=person&id=314452&useIds=false&club_member=false&lastName=Smith&status=Created
```

Output example in JSON format:

```
{"status": "ok"}
```

# update2



**Warning:** This method is deprecated. Please change any existing code to use [updateRecord](#) on page 777, which takes an object ID as a parameter.

---

## Purpose

Updates an existing data record from URL parameters.

## HTTP Method

POST or PUT

## URL

<https://www.rollbase.com/rest/api/update2>

## URL Parameters

*sessionId*

The session ID obtained at log in as an HTTP header, a URL parameter, or with HTTP basic authentication.

*id*

The record ID.

*useIds*

Boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise use integration codes and record names.

*field1*

Parameter's name – URL-encoded integration name of field to set; parameter's value – URL-encoded value of field to set.

*field2*

Same as *field1* for other fields.

Names of URL parameters used by this API are integration names of your fields. If a field is not found, the system ignores the URL parameter and keeps the current field's value. Fields' values must be formatted the same way as CSV imported values; for more information, see [Importing Data](#).

## Permissions Required

UPDATE for the requested record

## Response

Standard success or failure output.

## Example

Input example (URL parameters):

```
&id=314452&useIds=false&club_member=false&lastName=Smith&status=Created
```

Output example:

```
<resp status="ok">
  <Msg>12 fields have been processed</Msg>
</resp>
```

# view

## Purpose

---



**Warning:** This method is deprecated.

---

# Rollbase SOAP Methods

Many of the topics in this section include code samples written in Java using the Axis package.

## Rollbase SOAP API WSDL

The Rollbase SOAP API WSDL is located at: <https://www.rollbase.com/webapi/wsdl/api.wsdl>. This link is available in **Setup**, under **Applications Setup > SOAP APIs**. Rollbase uses Literal WSDL encoding. If your SOAP infrastructure does not support Literal WSDL encoding, consider using the REST API.

## Working with Field Data

When working with records using the Rollbase SOAP API, you use a special container called `DataField`. This serves as a container for data stored in a single record field and includes the following attributes:

- **Integration name (mandatory):** Integration name of the field the data belongs to.
- **Data value:** the actual data that belongs to the field in this record, represented as string. The string format depends on the type of the given field:
  - For check boxes: `true` or `false`, `yes` or `no`
  - For numeric fields: a numeric value as decimal string
  - For single picklists: the integration code of the selected item or item name
  - For multi-select picklists: the integration codes of the selected items or item names separated by the `"|"` symbol
  - For lookup fields: the numeric IDs or names (must be unique) of related records separated by the `"|"` symbol
  - For date fields: a date string in the format corresponding to the current user's date format preference. Example: `05/01/2009`
  - For date and time fields: a date-time string in the format corresponding to the current user's date format preference. The time zone for this string is the same as time zone selected in the current user's preferences. For example: `05/01/2009 1:00 PM` is interpreted as `05/01/2009 1:00 PM PST`, if the current user has the PST time zone setting.

## DataObj Container Class

### Purpose

Container class for a complete record. You can use this class to pass a complete record with all of its fields. APIs that return a complete record will return an instance of this class. Attribute names are case-sensitive.

For Rollbase objects, you must include the `id` and `objDefName`. For external objects, you must include the `UID` and `objDefName`.

### Syntax

#### Class Attributes

`id`

A string containing the record ID.

`UID`

A string containing the object ID for external objects, including objects mapped to OpenEdge Services.

*objDefName*

A string containing the object definition integration name.

*DataFieldArr*

An array of `DataField` instances containing the values for the record's fields.

## Example

See [getRecord\(\)](#) on page 805 for an example.

# bulkCreate()

## Purpose

Creates new records using existing Import Map and supplied CSV string.

## Syntax

```
bulkCreate(string sessionId, long mapId, boolean sync, string csvString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*mapId*

A long value containing the ID of an existing import map.

*csvString*

A CSV string containing import data.

*mapId*

A long value containing the ID of an existing import map.

## Output

Comma-separated IDs of newly created records for synchronous import

## Permissions Required

CREATE permission

# bulkCreateUpdate()

## Purpose

Updates existing records or creates new ones using existing Import Map and supplied CSV string.

## Syntax

```
bulkCreateUpdate(string sessionId, string mapId, [string uniqueFieldId|uniqueCombinationId], boolean sync, string csvString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*mapId*

A long value containing the ID of an existing import map.

*uniqueFieldId*

ID of a unique field used to identify the records to be updated

*uniqueCombinationId*

ID of a **Unique Fields Combination** trigger

*sync*

A boolean value: if `true`, do synchronous import and return IDs of created records; if `false`, do asynchronous import.

*csvString*

A CSV string containing import data.

## Output

Comma-separated IDs of updated records for synchronous update

## Permissions Required

UPDATE permission for each of requested records. CREATE permission to create new records.

# bulkUpdate()

## Purpose

Updates existing records using existing Import Map and supplied CSV string.

## Syntax

```
bulkUpdate(string sessionId, long mapId, [string uniqueFieldId|uniqueCombinationId], boolean sync, string csvString);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*mapId*

A long value containing the ID of an existing import map.

*uniqueFieldId*

ID of unique field used to identify records to be updated

*uniqueCombinationId*

ID of the **Unique Fields Combination** trigger

*sync*

A boolean value: if `true`, do synchronous import and return IDs of created records; if `false`, do asynchronous import.

*csvString*

A CSV string containing import data.

## Output

Comma-separated IDs of updated records for synchronous update

## Permissions Required

UPDATE permission for each record

# clearDataObjectCache()

## Purpose

Clears the cache of object data records from production servers. Progress recommends that you call this method after heavy use of the SOAP API to perform updates.

## Syntax

```
clearDataObjectCache(sessionId);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

## Output

None

## Permissions Required

All metadata methods require administrative privileges, regardless of the view, edit, create and delete permissions set on the application or object. Establish the session by logging in with credentials for an administrative user.

## Example

Output example:

```
<resp status="ok">
  <Msg>Cleared cache of object records</Msg>
</resp>
```

# create()

## Purpose

Creates a new record.

This method only works with native Rollbase objects. To create an external object record, see [createRecord\(\)](#) on page 789

## Syntax

```
create(string sessionId, string objDefName, DataFieldArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing Integration name for object definition

*arr*

A DataFieldArr array of values for fields of a newly created record. Record ID and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

ID of newly created record as a long

## Permissions Required

CREATE permission for requested object type

## Example

```
// Populate data Fields to be set for new record
DataField[] Fields = new DataField[5];
DataField Field = new DataField();
Field.setName("firstName");
Field.setValue("John");
```

```
Fields[0]=Field;
Field.setName("lastName");
Field.setValue("Smith");
Fields[1]=Field;

// Call create API
binding.create(sessionId, "lead", new DataFieldArr(Fields), true);
```

## createArr()

### Purpose

For native Rollbase objects, this method creates an array of new object records. This API increments the hits counter for each array element. This method and [createArrNoAudit\(\)](#) each take an array of transport Objects of type `DataObj` wrapped in a `DataObjArr` instance. This transport Object carries:

- Object name
- Record ID (for update only)
- Fields to be set

This is a convenient form for transporting data. Note that, this API is optimized for faster performance in cases when related records must be resolved by name.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [createArr2\(\)](#) on page 786.

---

### Syntax

```
createArr(string sessionId, DataObjArr arr, boolean useIds);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

A `DataObjArr` array of `DataObjs` with data for new records. Record IDs and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

### Output

Ids of newly created records as a `LongArr`

### Required Permissions

CREATE permission for requested object type

## Example

```
// Array of transport objects to be sent to createArr
DataObj[] arr = new DataObj[10];

// Populate data Fields to be set for record 0
DataField[] Fields = new DataField[5];
DataField Field = new DataField();
Field.setName("amount");
Field.setValue("1000");
Fields[0]=Field;

// Create record 0
DataObj obj = new DataObj();
obj.setObjDefName("lead");
obj.setFields(Fields);
arr[0] = obj;

// Repeat for record 1

// Call createArr API
binding.createArr(sessionId, new DataObjArr(arr), false);
```

# createArr2()

## Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), creates an array of new object records. This API increments the hits counter for each array element. This transport Object carries:

- Object name
- Record ID (for update only)
- Fields to be set

This is a convenient form for transporting data.

Note: createArr API is optimized for faster performance in cases when related records must be resolved by name.

---

**Note:** For native Rollbase objects, you can use the method, [createArr\(\)](#) on page 785.

---

## Syntax

```
createArr2(string sessionId, DataObjArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

A DataObjArr array of DataObjs with data for new records. Record IDs and auto-numbers are assigned automatically.

*useIds*

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

Ids of newly created records as a `StringArr`

## Required Permissions

CREATE permission for requested object type

## Example

```
// Array of transport objects to be sent to createArr
DataObj[] arr = new DataObj[10];

// Populate data Fields to be set for record 0
DataField[] Fields = new DataField[5];
DataField Field = new DataField();
Field.setName("amount");
Field.setValue("1000");
Fields[0]=Field;

// Create record 0
DataObj obj = new DataObj();
obj.setObjDefName("lead");
obj.setFields(Fields);
arr[0] = obj;

// Repeat for record 1

// Call createArr2 API
binding.createArr2(sessionId, new DataObjArr(arr), false);
```

# createArrNoAudit()

## Purpose

Creates an array of new object records. This API increments the hits counter for each array element. This is the same as [createArr\(\)](#), except that it blocks Audit records.

## Syntax

```
createArrNoAudit(string sessionId, DataObjArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

A `DataObjArr` array of `DataObjs` with data for new records. Record IDs and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

Ids of newly created records as a `LongArr`

## Required Permissions

CREATE permission for requested object type

# createCustomer()

## Purpose

Starts the process of creating a new customer tenant. A welcome email containing a temporary password is sent to the first administrative user when the process is complete. If the password field is included in the list of fields, as in the example below, its value is used as the password for the first administrative user. In this case, the system does not send the welcome email to first administrative user and the password will not expire after the first log in.

## Syntax

```
createCustomer(string sessionId, DataFieldArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

A `DataFieldArr` array of values for fields of a newly created record. Record ID and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

ID of newly created Customer

## Permissions Required

CREATE permission for Customer object type

## Example

```
DataField[] arr = new DataField[7];
arr[0] = getField("companyName", "API Test");
arr[1] = getField("email", "myemail@mycompany.com");
```

```
arr[2] = getField("lastName", "Admin");
arr[3] = getField("loginName", "myemail@mycompany.com");
arr[4] = getField("timeZone", "Pacific/Guam");
arr[5] = getField("mailSender", "noreply@mycompany.com");
arr[6] = getField("password", "my_password");
binding.createCustomer(sessionId, new DataFieldArr(arr), true);
```

## createRecord()

### Purpose

Similar to [create\(\)](#) on page 784, creates a new record. Returns the ID of the newly created record.

This method works with external objects (including those mapped to OpenEdge services) and native Rollbase objects.

### Syntax

```
createRecord(string sessionId, string objDefName, DataFieldArr arr, string
useIds);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*arr*

A DataFieldArr array of values for fields of a newly created record. Record ID and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

### Output

The ID of the newly created record as a string.

### Permissions Required

CREATE permission for the object type.

## delete()

### Purpose

Moves the specified record to the Recycle Bin. For users and Customers this API deletes records permanently.

This method only works with native Rollbase objects. To delete an external object record, see [deleteRecord\(\)](#) on page 792.

### Syntax

```
delete(string sessionId, long id);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

### Output

None

### Permissions Required

DELETE permission for requested record

### Example

```
long id = 123456;  
binding.delete(sessionId, id);
```

## deleteArr()

### Purpose

For native Rollbase objects, this method moves the specified array of records to the Recycle Bin. This method increments the hits counter for each array element.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [deleteRecords\(\)](#) on page 792.

---

### Syntax

```
deleteArr(string sessionId, LongArr ids);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*ids*

`LongArr` array of record ids to be deleted.

## Output

None

## Permissions Required

DELETE permission for requested record

## Example

```
long[] ids = new long[] { 123456, 123457 };  
binding.deleteArr(sessionId, new LongArr(ids));
```

# deleteArrNoAudit()

## Purpose

For native Rollbase objects, this method moves the specified array of records to the Recycle Bin. Similar to [deleteArr\(\)](#), except `deleteArrNoAudit()` blocks Audit records. This method will increment the hits counter for each array element.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [deleteRecords\(\)](#) on page 792.

---

## Syntax

```
deleteArrNoAudit(string sessionId, LongArr ids);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*ids*

`LongArr` array of record ids to be deleted.

## Output

None

## Permissions Required

DELETE permission for requested record

## deleteRecord()

### Purpose

Moves Rollbase objects to the Recycle bin.

This method works with external objects (including those mapped to OpenEdge services) and native Rollbase objects.

### Syntax

```
deleteRecord(string sessionId, string objDefName, string uid);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the record ID.

### Output

None.

### Permissions Required

DELETE for requested record.

## deleteRecords()

### Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this method moves the specified array of records to the Recycle Bin. This method will increment the hits counter for each array element.

---

**Note:** For native Rollbase objects, you can use the method, [deleteArr\(\)](#) on page 790 and [deleteArrNoAudit\(\)](#) on page 791.

---

### Syntax

```
deleteRecords(string sessionId, String objDefName, StringArr ids, Boolean blockAudit);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*ids*

StringArr array of record ids to be deleted.

*blockAudit*

If set to true, no audit records are created.

## Output

None

## Permissions Required

DELETE permission for requested record

## Example

Introduce example here.

```
String[] arr = new String[] { "100890", "100891" };
binding.deleteRecords(sessionId, "client", new StringArr(arr));
```

# detailedSearch()

## Purpose

Performs a detailed search throughout the customer's Rollbase database. This is equivalent to the search capabilities provided by the Rollbase UI.

## Syntax

```
detailedSearch(string sessionId, string query, string objDefName,
searchFilterArr filterArr, string joinType, string expression)
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

String query for full-text search (see above)

*objDefName*

String integration name for selected object definition. This parameter is optional and allows you to narrow the search to a specified object.

*filterArr*

SearchFilterArr instance which wraps an array of SearchFilter instances

*joinType*

Type of join between filters. Valid values are: AND (default), OR, or null (if expression is present)

*expression*

String SQL Expression that includes tokens for filters. Example: ((1 OR 2) AND 3)

## Output

IDs of all records found in the search, returned as LongArr

# getBinaryData()

## Purpose

Retrieves the file attachment associated with a particular file field from a specific record.

## Syntax

```
getBinaryData(string sessionId, long id, string fieldName);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*fieldName*

A string containing the field integration name.

## Output

Binary data (file attachment) from specified file Field wrapped in a ByteArr

## Permissions Required

VIEW permission for requested record

## Example

```
long id = 123456;
ByteArr arr = binding.getBinaryData(sessionId, id, "picture");
if (arr != null {
    byte[] data = arr.getArr();
}
```

# getCodebyId()

## Purpose

Retrieves the integration code of a pick item or status by ID.

---

**Note:** This API only works when the ID is numeric. Therefore, Picklist(char) or Radio Button(char) are not supported.

---

## Syntax

```
getCodeById(string sessionId, string objDefName, string fieldName, long id);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing integration name of object definition

*fieldName*

String containing integration name of Field (picklist or status)

*id*

Long numeric ID of picklist item or status

## Output

Integration code of item with matching ID or null

## Example

```
String code = binding.getCodeById(sessionId, "lead", "type", 98765);
```

## getCount()

### Purpose

Retrieves the total number of records in a view (see [Views and Search](#) for more information about views).

### Syntax

```
getCount(string sessionId, long viewId, string filterName, string filterValue);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*viewId*

The original ID of the view (as a long value).

*filterName*

The name of the field to filter output using the equals option (will replace the filter set in the view, if any).

*filterValue*

The value of the field to filter output using the equals option.

### Output

Number of Records

### Permissions Required

VIEW permission for requested record

### Example

```
int count = binding.getCount(sessionId, 123, "lastName", "Grey");
```

## getIdByCode()

### Purpose

Retrieves the ID of the pick item or status by integration code.

---

**Note:** This API only works when the ID is numeric. Therefore, Picklist(char) or Radio Button(char) are not supported.

---

## Syntax

```
getIdByCode(string sessionId, string objDefName, string fieldName, string code);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

String containing integration name of object definition

*fieldName*

String containing integration name of Field (picklist or status)

*code*

String containing integration name of picklist item or status

## Output

ID of item with matching integration code or -1

## Example

```
long id = binding.getIdByCode(sessionId, "lead", "type", "HOT");
```

# getDataField()

## Purpose

For native Rollbase objects, this method retrieves the value of a single field from a specific record.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [getDataField2\(\)](#) on page 798.

---

## Syntax

```
getDataField(string sessionId, long id, string fieldName, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*fieldName*

A string containing the field integration name.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

Field's value wrapped in a `DataField` container.

## Permissions Required

VIEW permission for requested record

## Example

```
long id = 123456;
DataField field = binding.getDataField(sessionId, id, "lastName", false);
if (field != null) {
    String lastName = field.getValue();
}
```

# getDataField2()

## Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this method retrieves the value of a single field from a specific record.

---

**Note:** For native Rollbase objects, you can use the method, [getDataField\(\)](#) on page 797.

---

## Syntax

```
getDataField2(string sessionId, String objDefName, String uid, String
fieldName, Boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the record ID.

*fieldName*

A string containing the field integration name.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

Field's value wrapped in a `DataField` container.

## Permissions Required

VIEW permission for requested record

## Example

```
DataField field = binding.getDataField2(sessionId, "client", "123456",
"lastName", false);
if (field != null) {
    String lastName = field.getValue();
}
```

# getDataObj()

## Purpose



**Warning:** This method is deprecated. Please change any existing code to use [getRecord\(\)](#) on page 805, which takes an object ID as a parameter.

---

Retrieves all field data, including formulas, for a given record.

## Syntax

```
getDataObj(string sessionId, long id, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

All of the field data for the specified object record. Each field value is wrapped in a `DataField` container.

## Permissions Required

VIEW permission for requested record

## Example

```
long id = 123456;
DataObj obj = binding.getDataObj(sessionId, id, true);
if (obj != null) {
    DataField[] fields = obj.getFields();
}
```

# getExchangeRate()

## Purpose

Returns the exchange rate between two currencies on the given date.

## Syntax

```
getExchangeRate(string sessionId, string srcCode, string destCode,
Calendar date);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*srcCode*

The currency code to convert from.

*destCode*

The currency code to convert to.

*date*

The `Calendar` object containing the date.

## Output

Exchange rate as decimal

## Example

```
Calendar today = Calendar.getInstance();
today.setTime(new Date());

double rate = binding.getExchangeRate(sessionId, "EUR", "USD", today);
```

## getPage()

### Purpose

Retrieves a specified range of data records in a View. The selected View defines sorting and filtering of the output. The amount of processing and time required to get complete results can vary widely, depending on whether it fetches related records and the number of rows you specify per page.

### Syntax

```
getPage(string sessionId, long viewId, string filterName, string
filterValue, int startRow, int rowsPerPage, boolean useIds);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*viewId*

The original ID of the view (as a long value).

*filterName*

The name of the field to filter output using the equals option (will replace the filter set in the view, if any).

*filterValue*

The value of the field to filter output using the equals option.

*startRow*

An integer representing the number of the data record to start from.

*rowsPerPage*

An integer representing the number of data records in output.

*useIds*

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

### Output

Selected records as DataObjArr

## Permissions Required

VIEW permission for requested record

### Example

```
 DataObj[] arr = binding.getPage(sessionId, 25513, " lastName", "Grey ", 0, 20,
    false).getObjects();
    for (DataObj obj: arr) {
        System.out.println("\nID="+obj.getId()+
            " objName="+obj.getObjDefName());
        DataField[] fields = obj.getFields();
        for (DataField field: fields) {
            System.out.println("Name="+field.getName()+
                " Value="+field.getValue());
        }
    }
```

## getRelatedIDs()

### Purpose

Retrieves an array of related record IDs. Unlike `getRelationships()`, this API works with external objects, including those based on OpenEdge services.

### Syntax

```
getRelatedIDs(sessionId, objDefName, id, relName);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*id*

A string containing the record ID.

*relName*

A string containing the relationship integration name.

### Output

A `StringArr` containing a list of related record IDs.

## Permissions Required

VIEW permission for the requested record.

## Example

The following example returns the IDs of the records related to the oeTest object through the R291855 relationship.

```
LongArr arr = binding.getRelatedIDs(sessionId, "oeTest", "789456", "R291855");
if (arr != null) {
    String[] ids = arr.getArr();
}
```

# getRelationships()

## Purpose

Retrieves an array of related record IDs.

---

**Note:** This method does not work for external objects. Use [getRelatedIDs\(\)](#) on page 802 for external objects, including those mapped to OpenEdge services.

---

## Syntax

```
getRelationships(string sessionId, long id, string relName);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*relName*

Relationship's integration name

## Output

IDs of related records wrapped in a `LongArr`

## Permissions Required

VIEW permission for requested record

## Example

```
long id = 123456;
LongArr arr = binding.getRelationships(sessionId, id, "R76543");
if (arr != null) {
    long[] ids = arr.getArr();
}
```

## getRuntimeStatus()

### Purpose

Retrieves the runtime status of the Customer on the Web API server.

### Syntax

```
getRuntimeStatus(string sessionId, long id);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

### Output

Runtime status:

- RUNTIME\_ERROR = -1;
- RUNTIME\_UNLOADED = 1;
- RUNTIME\_LOADED = 2;
- RUNTIME\_LOADING = 3;
- RUNTIME\_CREATING = 4;
- RUNTIME\_BUSY = 5;

### Permissions Required

EDIT permission for requested record

### Example

```
int status = binding.getRuntimeStatus(sessionId, 123456);
```

## getUpdated()

### Purpose

Retrieves an array of record IDs of a specified object type that were either created or updated within the given date/time interval. This method is not available for external objects, including those mapped to OpenEdge service objects.

### Syntax

```
getUpdated(string sessionId, Calendar from, Calendar till, string
objDefName);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*Calendar from*

Beginning of date/time interval (not null)

*Calendar till*

End of date/time interval or null

*objDefName*

String integration name for object definition to search

## Output

IDs of all records found in the search, returned as `LongArr`

## Example

```
Calendar from = Calendar.getInstance();
from.setTime(...);
Calendar till = null;

LongArr arr = binding.getUpdated(sessionId, from, till, "lead");
long[] ids = arr.getArr();
```

# getRecord()

## Purpose

Retrieves all field data (including formulas) for a given record in XML format. This method works with external objects, including those mapped to OpenEdge services.

## Syntax

```
getRecord(string sessionId, string objDefName, string uid, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the object ID.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

All of the field data for the specified object record. Each field value is wrapped in a `DataField` container.

## Permissions Required

`VIEW` permission for the requested record.

## Example

Introduce example here.

```
DataProvider obj = binding.getRecord(sessionId, "client", "123456", true);
if (obj != null) {
    DataField[] fields = obj.getFields();
}
```

# login()

## Purpose

Performs a user log in and initiates a SOAP API session. Either this method or [login2\(\)](#) (see below) must be called prior to any other API call. The customer tenant to login is determined by the login name. The Web API login creates a server-side session which might expire according to the security level set for the customer tenant (see [Security and Access Control](#) for more information).

The API client should re-login if the session expires. Progress recommends that an API client logs out after performing a group of operations instead of keeping the API session open for a long time. When an API client logs in, all sessions with the same user credentials are terminated.

Subsequent API calls that use a session ID are considered to be made on behalf of the logged in user. That user's permissions are checked for each subsequent call. For example, to update a record a logged in user must have sufficient permissions or the API call will fail.

## Syntax

```
login(string loginName, string password);
```

## Parameters

*loginName*

String containing login name for an active Rollbase user account

*password*

String containing user password

## Output

Session ID that must be used in all subsequent API calls during this session

# login2()

## Purpose

Performs regular user or super-admin login to the specified customer tenant and initiates a SOAP API session. Unlike the `login()` method, the customer to login to is specified explicitly by ID. Master Server users with super-admin login permissions can use `login2()` to call Web Service APIs on the specified customer tenant.

## Syntax

```
login2(string loginName, string password, string custID);
```

## Parameters

*loginName*

String containing login name for an active Rollbase user account

*password*

String containing user password

*custID*

ID of customer tenant to log into

## Output

Session ID that must be used in all subsequent API calls during this session

# logout()

## Purpose

Terminates the specified API session. This method should be called to explicitly end each SOAP session.

## Syntax

```
logout(string sessionID);
```

## Parameters

*sessionID*

A string containing the session ID obtained at log in.

## Output

None

## Example

```
URL url = new URL("https://www.rollbase.com/webapi/services/rpcrouter");

RpcrouterSoapBindingStub binding = (RpcrouterSoapBindingStub)
    new IWebServicesServiceLocator().getrpcrouter(url);
binding.setTimeout(60000);

String sessionId = binding.login("username", "password");
System.out.println("loginsuccessful: sessionId="+sessionId);

// Perform some API calls

binding.logout(sessionId);
```

# metadataSearch()

## Purpose

Searches for a string in metadata and returns results in XML format.

## Syntax

```
metadataSearch(string sessionId, String query);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

The string to search.

## Output

An XML document with nodes for each metadata entity found for the search string.

## Permissions Required

Full administrative permissions.

## Example

```
String xmlResults = binding.metadataSearch(sessionId, "test");
```

## SearchFilter()

### Purpose

Defines a single search filter (there can be up to five per search).

### Syntax

```
SearchFilter(string fieldName, string opCode, string opValue);
```

### Parameters

*fieldName*

String name of data field used for filtering

*opCode*

Code of operation. The following codes are available (not all codes are compatible with all fields; the same rules apply as in the UI):

- EQ equals
- NEQ not equal
- ST starts with
- CT contains
- NCT does not contain
- LT less than
- GT greater than
- LE less or equal
- GE greater or equal
- IN in array
- NIN not in array
- CH checked
- NCH unchecked
- INC including
- NINC not including
- NUL is null
- NNUL not null
- IS is
- TREE in sub-tree
- NTREE not in sub-tree
- FLAG flagged
- UFLAG unflagged

- VIEW viewed
- UVIEW unviewed

*opValue*

Filter's value as a string. Use comma-separated enumerations for arrays.

## Output

### Example

```
SearchFilter[] filters = new SearchFilter[1];
SearchFilter filter = new SearchFilter();
filter.setFieldName("firstName");
filter.setOpCode("EQ");
filter.setOpValue("Smith");
filters[0] = filter;

SearchFilterArr filterArr = new SearchFilterArr(filters);

LongArr arr = binding.detailedSearch(sessionId, null, "lead", filterArr, "AND",
null);
long[] ids = arr.getArr();
```

# selectNumber()

### Purpose

Runs an SQL SELECT query on the server and returns a single decimal value as the result. This is a simplified version of [selectQuery\(\)](#) and is suitable for calculations such as finding a sum of values.

### Syntax

```
selectNumber(string sessionId, string query);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

An SQL SELECT query. See the examples described for [selectQuery\(\)](#).

### Output

Single result of the SELECT query as a decimal number

### Permissions Required

VIEW permission for all requested records

## Example

```
Double total = binding.selectNumber(sessionId,
    "SELECT SUM(amount) FROM customer WHERE name LIKE 'M%'");
```

# selectQuery()

## Purpose

Runs an SQL SELECT query on the server and returns the results as a 2D-array. The SELECT query used in this method consists of the following parts:

- The SELECT statement expects columns or expressions to be selected (mandatory). Use the field integration names as SQL column names. You can use expressions such as COUNT(1). You cannot use \* to retrieve all columns.
- The FROM clause must consist of exactly one object name (mandatory).
- The WHERE clause can include a valid SQL expression to narrow the selection (optional). Use field integration names as SQL column names.
- The ORDER BY clause can include a valid SQL expression to order the selection (optional). Use field integration names as SQL column names.

You can use special tokens in your queries such as:

- TODAY for current time
- WEEK for 12PM of last Sunday
- MONTH for 12PM of 1st day of current month
- QUARTER for 12PM of 1st day of current quarter
- YEAR for 12PM of 1st day of current year
- CURR\_USER for id of currently logged in user

Object and Field names are case-sensitive, while other components of the SQL query are not.

Use #code suffix to fetch integration codes for picklist fields rather than IDs. See [Adding Logic to an Application](#) for more information.

Examples of valid queries on the USER object (you can find more examples in [Adding Logic to an Application](#)):

```
SELECT id, name, updatedAt, updatedBy FROM USER WHERE name LIKE 'M%' ORDER BY
name

SELECT count(1) FROM USER WHERE name LIKE 'M%'

SELECT count(1) FROM USER WHERE updatedBy>=QUARTER
```

The `selectQuery()` method, as well as `selectValue()` and `selectNumber()`, use the same permissions model as the server-side query API (see for more details). Either the current user or the API User must have VIEW access for all records of the given type.

## Syntax

```
selectQuery(string sessionId, string query, int maxRows);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

SQL SELECT query

*maxRows*

An integer value representing the maximum number of rows to retrieve (can be configured per Rollbase instance)

## Output

SELECT query wrapped in a `StringArr2D`

## Permissions Required

VIEW permission for all requested records

## Example

```
StringArr2D values = binding.selectQuery(sessionId, "SELECT loginName, firstName, lastName FROM USER ORDER BY lastLogin DESC", 10);
StringArr[] recs = values.getArr();
for (StringArr rec: recs) {
    String[] data = rec.getArr();
    String loginName = data[0];
    String firstName = data[1];
    String lastName = data[2];
}
```

# selectValue()

## Purpose

Runs an SQL SELECT query on the server and returns a single `String` value as the result.

## Syntax

```
selectValue(string sessionId, string query);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

An SQL SELECT query. See the examples described for [selectQuery\(\)](#).

## Output

Single result of SELECT query as string

## Permissions Required

VIEW permission for all requested records

## Example

```
// Login name of last login user (one value)
String loginName = binding.selectValue(sessionId,
"SELECT loginName FROM USER ORDER BY lastLogin DESC");
```

# setBinaryData()

## Purpose

Sets the binary data (file attachment) associated with a particular file Field from a specific record.

## Syntax

```
setBinaryData(string sessionId, long id, string fieldName, string
contentType, string suggestedFileName, ByteArr binData);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*fieldName*

File Field's integration name

*contentType*

MIME content type of data. For example:

```
text/html
text/plain
application/pdf
application/msword
application/excel
application/vnd.ms-powerpoint
application/wordperfect5.1
application/rtf
```

*suggestedFileName*

Suggested file name (with valid file extension)

*binData*

File's binary data wrapped in a `ByteArr`

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
// Read binary data
byte[] data;
long id = 123456;

ByteArr binData = new ByteArr(data);
binding.setBinaryData(sessionId, id, "pdfData", "application/pdf",
"invoice.pdf", binData);
```

# setDataField()

## Purpose

For native Rollbase objects, this method sets the value of a single Field for a specified record.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [setDataField2\(\)](#) on page 815.

---

## Syntax

```
setDataField(string sessionId, string id, DataField df, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*df*

A new value for the field wrapped in a `DataField` container.

*useIds*

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
// Populate data Field to be set for existing record
DataField Field = new DataField();
Field.setName("R34567");
Field.setValue("100088,100089");

// Call setDataField API
long id = 123456; // Record id - required
binding.setDataField(sessionId, id, Field, true);
```

# setDataField2()

## Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this method sets the value of a single Field for a specified record.

---

**Note:** For native Rollbase objects, you can use the method, [setDataField\(\)](#) on page 814.

---

## Syntax

```
setDataField2(string sessionId, String objDefName, String uid, DataField df, Boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the record ID.

*df*

A new value for the field wrapped in a `DataField` container.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
// Populate data Field to be set for existing record
DataField Field = new DataField();
Field.setName("R34567");
Field.setValue("100088,100089");

// Call setDataField2 API
binding.setDataField2(sessionId, "client", "123456", Field, true);
```

# setExchangeRate()

## Purpose

Sets the exchange rate between two currencies on a given date.

## Syntax

```
setExchangeRate(string sessionId, string srcCode, string destCode,
Calendar date, double rate);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*srcCode*

The currency code to convert from.

*destCode*

The currency code to convert to.

*date*

The `Calandar` object containing the date.

*rate*

Double containing rate to be set

## Output

None

## Example

```
Calendar today = Calendar.getInstance();
today.setTime(new Date());

binding.setExchangeRate(sessionId, "EUR", "USD", today, 1.2823);
```

# setRelationship()

## Purpose

For native Rollbase objects, this method assigns an array of related records to the specified record.

---

**Note:** For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), you can use the method, [setRelatedIDs\(\)](#) on page 818.

---

## Syntax

```
setRelationship(string sessionId, long id, string relName, LongArr arr);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*relName*

Relationship's integration name

*arr*

IDs of related records

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
long id = 123456;
long[] arr = new long[] { 100890, 100891 };

binding.setRelationships(sessionId, id, "R1223456", new LongArr(arr));
```

# setRelatedIDs()

## Purpose

For external objects (such as those mapped to external tables, to OpenEdge Service objects, or through a D2C connection), this method assigns an array of related records to the specified record.

**Note:** For native Rollbase objects, you can use the method, [setRelationship\(\)](#) on page 817.

## Syntax

```
setRelatedIDs(string sessionId, String objDefName, String uid, String relName, StringArr arr);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the record ID.

*relName*

Relationship's integration name

*arr*

IDs of related records

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
String[] arr = new String[] { "100890", "100891" };
```

```
binding.setRelatedIDs(sessionId, "client", "123456", "R1223456", new
StringArr(arr));
```

## textSearch()

### Purpose

Performs a full-text search in the Rollbase database. The search query has the same syntax used in the standard Rollbase interface. For more information about search syntax, see [Views and Search](#).

### Syntax

```
textSearch(string sessionId, string query, [string objDefName]);
```

### Parameters

*sessionId*

A string containing the session ID obtained at log in.

*query*

Query for full-text search like "John Smith" (see [Views and Search](#) for more details)

*objDefName*

Integration name for selected object definition. This parameter is optional and allows you to narrow the search to a specified object.

### Output

IDs of all records found in the search, returned as `LongArr`

### Example

```
LongArr arr = binding.textSearch(sessionId, "John Smith", "lead");
long[] ids = arr.getArr();
```

## update()

### Purpose

Updates an existing record.

This method only works with native Rollbase objects. To update an external object record, see [updateRecord](#) on page 823.

### Syntax

```
update(string sessionId, long id, DataFieldArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

A long value containing the record ID.

*arr*

A `DataFieldArr` array of values for fields of a newly created record. Record ID and auto-numbers are assigned automatically.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
// Populate data Fields to be updated for existing record
DataField[] Fields = new DataField[5];
DataField Field = new DataField();
Field.setName("R34567");
Field.setValue("100088,100089");
Fields[0]=Field;

// Call update API
long id = 123456; // Record id - required
binding.update(sessionId, id, new DataFieldArr(Fields), true);
```

# updateArr()

## Purpose

Updates array of existing Object records. Note that this method increments the API hits counter for each array element.

## Syntax

```
updateArr(string sessionId, DataObjArr arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

An Array of DataObj instances with data for existing records. DataObj instances must include valid record IDs.

*useIds*

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested records

## Example

```
// Array of transport objects to be sent to updateArr
DataObj[] arr = new DataObj[10];

// Populate data Fields to be updated for record 0
DataField[] Fields = new DataField[5];
DataField Field = new DataField();
Field.setName("amount");
Field.setValue(1000);
Fields[0]=Field;

// Update record 0
DataObj obj = new DataObj();
obj.setId(123456); // Required for update
obj.setObjDefName("lead");
obj.setFields(Fields);
arr[0] = obj;

// Repeat for record 1

// Call updateArr API
binding.updateArr(sessionId, new DataObjArr(arr), false);
```

# updateArrNoAudit()

## Purpose

Updates array of existing object records. This method is similar to [updateArr](#), except that it blocks audit records. This method will increment the API hits counter for each array element.

## Syntax

```
updateArrNoAudit(string sessionId, DataObj arr);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*arr*

An Array of DataObj instances with data for existing records. DataObj instances must include valid record IDs.

*useIds*

A boolean value: if `true`, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested records

# updateCustomer()

## Purpose

Updates a Customer record with the supplied values.

## Syntax

```
updateCustomer(string sessionId, long id, arr, useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*id*

The customer ID.

*arr*

Values for fields to be updated.

*useIds*

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

## Output

None

## Permissions Required

UPDATE permission for requested record

## Example

```
DataField[] arr = new DataField[1];
arr[0] = getField("companyName", "API Updated Test");
binding.updateCustomer(sessionId, 123456, new DataFieldArr(arr), true);
```

# updateRecord

## Purpose

Updates an existing record.

This method works with external objects (including those mapped to OpenEdge services) and native Rollbase objects.

## Syntax

```
updateRecord(string sessionId, string objDefName, string uid, DataFieldArr
arr, boolean useIds);
```

## Parameters

*sessionId*

A string containing the session ID obtained at log in.

*objDefName*

A string containing the object definition integration name.

*uid*

A string containing the record ID.

*arr*

A DataFieldArr array of values for fields of a newly created record. Record ID and auto-numbers are assigned automatically.

**useIds**

A boolean value: if true, use numeric IDs for lookup and picklist values; otherwise, use integration codes and record names.

**Output**

None.

**Permissions Required**

UPDATE for the record.

# Rollbase CSS Styles

The topics in this section describe the HTML styles used by Rollbase Cascading Style Sheets (CSSs).

## Table Styles

The following classes define Rollbase table styles.

CSS Class Name	Description
rbs_breadcrumbs	Renders the table that has the setup breadcrumbs link (print   Edit Page Layout)
rbs_componentContentTable	Renders the table that contains all the components in the component pages.
rbs_detailTable	Renders a wide table that displays details of components.(the tds inherit detailHTMLCol style)
rbs_editorFormTable	Renders the form table in page editor for portal pages
rbs_errormsgTable	Renders the error message table
rbs_fieldBottomColumn	Renders a column that fills the entire row and has bottom border (AssignPages.jsp)
rbs_HELPTIPTable	Renders the help tip table with yellow background
rbs_highlightTable	Renders a table with the background color set to highlight color
rbs_lightRedTable	Renders a table with light red background used in displaying error messages
rbs_lightsilverTable	Renders a table that has light silver background

CSS Class Name	Description
rbs_listTable	Renders the table that contains list views
rbs_lookupValue	Renders the table that holds the lookup values
rbs_mainContentTable	Renders the table that has all the contents in a page, sidebar and other contents
rbs_mediumPurpleTable	Renders a table with medium purple background used in report pages.
rbs_miniCalTable	Renders the mini-calendar table
rbs_notificationTable	Renders the notification table in page Editor pages.
rbs_outerMainComponentTable	Renders the outer table that contains the component content in component setup pages
rbs_primaryTableNoBorder	Renders the table with background color set to primary color with no borders
rbs_primaryWideTable	Renders a table with background color set to primary color that is wide and has a left border
rbs_roundedTable	Renders the blank rows in a section in primary color
rbs_secondaryWide	Renders a table with the background color set to secondary color
rbs_setupPageInnerTable	Renders the inner table in component setup pages.
rbs_setupPageOuterTable	Renders the outer table in component setup pages.
rbs_silverSideBorderTable	Renders a table with silver side borders.
rbs_silverTableWide	Renders a table with background color set to silver color with 100% width.
rbs_whiteBlankTable	Renders a blank table with white background
rbs_widePrimary	Renders a table with background color set to primary color that is wide and has a left border and right primary color borders
rbs_wideSilverEmptyTable	Renders a table with background color set to silver color with border bottom
rbs_wideTable	Renders a table that is wide and has no borders.

## Table Cell Styles

The following Rollbase CSS classes define the appearance of table cells.

CSS Class	Description
rbs_centerwide	Renders a column that is center aligned
rbs_InfoSmallWide	Renders detailed information that describes various features e.g., filters expression info. This style Renders info in small size and is used as column style
rbs_PageTopicWide	Renders the topic of the page in component setup pages (Object: Portal: etc)
rbs_recordActionCol	Render record action column (edit, delete button etc)
rbs_reportExpand	Render expanding column in report
rbs_rightWide	Render column wide and right
rbs_shortColFieldbottom	Renders short column with bottom border
rbs_silverbottom	Renders column with silver bottom border
rbs_thinSectionTitle	Renders the column that displays the thin section's title
rbs_warningIconCol	Renders the column that display warning icon in dialog box pages
rbs_wideLeftTop	Render column wide and left and top (Components.jsp)
td.rbs_boldDataCol	Render columns that have information that is bold (relationsEdit, relationshipConvert pages)
td.rbs_detailHTMLCol	Renders a column with detail information about the component property in Setup Page
td.rbs_disabledFieldProp	Renders html that displays disabled field properties in field create/Edit pages.
td.rbs_emailLeftDataCol	Renders column in 2 column layout pages that has the field input controls in email pages. This style is more prominently used in Email sending pages

CSS Class	Description
td.rbs_errorDataCol	Renders an error column in a 2 column layout pages that has the field input controls that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_errorDataColWide	Renders an error column in single column layout pages that has the field input controls that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_errorLabelRequired	Renders an error column in a 2 column layout pages that has the field labels that are required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_errorLabelRequiredWide	Wide Renders an error column in a single column layout pages that has the field labels that are required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_errorLabelRight	Renders error column in a 2 column layout pages that has the field labels that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_errorLabelRightWide	Renders error column in a single column layout pages that has the field labels that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_filtersTable	Renders the filters table in search, view, report and chart pages.
td.rbs_grayDetailHTMLcol	Renders a column with detail information about the component property in Setup Page
td.rbs_grayDetailInfoCol	Renders the help information about the component in detail in gray in setup pages
td.rbs_leftDataCol	Renders column in a 2 column layout pages that has the field input controls that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_leftDataColWide	Renders column in single column layout pages that has the field input controls that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_listItemValueRight	Renders the right column that displays the total summary of lists.

CSS Class	Description
td.rbs_listSummaryNumber	Renders the column that shows the totals of grouped list items in list views. Displays the number.
td.rbs_listSummaryText	Renders the left column that displays the total summary of lists.
td.rbs_propertiesEditorRightCol	Renders the right column that displays component properties in page editor page.
td.rbs_rightLabel	Renders column in a 2 column layout pages that has the field labels that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_rightLabelRequired	Renders column in a 2 column layout pages that has the field labels that are required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_rightLabelWide	Renders column in a single column layout pages that has the field labels that are not required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_rightWideLabelRequired	Renders column in a single column layout pages that has the field labels that are required. This style is more prominently used in Component setup pages and in record pages.
td.rbs_SetupColumnRequired	Renders an empty column in a two column layout pages. This style is more prominently used in component Setup pages.
td.rbs_SetupDataColWide	Renders a column in single column layout pages that has the field input controls in component setup pages. This style is more prominently used in Component setup pages and in record pages.
td.rbs_SetupLabelNoFieldBottom	Renders column in a two column layout pages where the column needs no bottom border. This style is more prominently used in component Setup pages.
td.rbs_silverCol	Renders a column that has white borders. Used in component setup pages.
td.rbs_summaryNumberCol	Renders the right column that displays the group summary of lists.
td.rbs_summaryTextCol	Renders the left column that displays the group summary of lists.

## Table Row Styles

The following CSS classes define Rollbase table row styles.

CSS Class	Description
rbs_parentRow	Renders a list Item that is a parent row for other items. This row has the group name for the list Items. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.
rbs_rowHighlightUnviewed	Renders a row to render a list Item that is currently not being viewed. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.
rbs_rowHighlightViewed	Renders a row to render a list Item that is currently being viewed. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.
rbs_rowUnviewed	Renders a row to render a list Item that has not been viewed yet. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.

CSS Class	Description
rbs_rowViewed	Renders a row to render a list item that has been viewed previously. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.
rbs_whiteListItem	Renders a white row to render a list item that has not been viewed yet. Other styles that must inherit this style are td.listItemValue, td.smallIcon, td.actionCol, td.groupSpacer, td.checkbox, td.listItemNumberValue, td.summary, td.totalSummary, td.rbs_listItemValueRight. these define the individual styles of column within this row.

## Text Styles

the following CSS classes define Rollbase text styles.

CSS Class	Description
rbs_ComponentHeaderInfo	Renders the components header info "Red= required information"
rbs_alertMsg	Renders text that display error messages or alerts
rbs_centerBold	Renders text center aligned and bold
rbs_graybold	Renders bold gray text (numbers in filters)
rbs_InfoGrayBoldWide	Renders detailed information that describes various features e.g., filters expression info. This style Renders info in that is gray and bold.
rbs_infoMessage	Renders the messages in #FFF1A8 color
rbs_InfoSmallWide	Renders detailed information that describes various features e.g., filters expression info. This style Renders info in small size and is used as column style
rbs_largeCourier	Renders text large and courier font
rbs_miniCalHeader	Renders Month and year text in calendar
rbs_rightLargeText	Renders text that is large and right aligned

CSS Class	Description
rbs_selectedButton	Renders a div with #FFF8CE color
rbs_silverBold	Renders the text in silver and bold and center. used in rendering the relation type names
rbs_smallGray	Renders text that is small in size and is gray
rbs_smallInfoGrayBold	Renders detailed information that describes various features e.g., filters expression info. This style Renders info in gray and small size and is bold
rbs_smallSpacer	Renders spacer column (lists)
rbs_smallStatus	Renders silver text very small
rbs_unselectedButton	Renders a div with gray color
rbs_xxlargeAlert	Renders a very large text as alert (used in confirmcheckDialog.jsp)

## Page Editor Styles

The following CSS classes define the style of Rollbase Page Editor.

rbs_AvailCompName	Renders the text that displays the available component names in page editor.
rbs_componentName	Renders component name in page editor
rbs_loginFormFields	Render the login form mockup in page editor
rbs_peDivSpacer	Renders a spacer div in page editor
rbs_peloginTable	Renders the mockup login table in page editor for login pages
rbs_propertiesEditorLeftCol	Renders the left column of properties table in page Editor
rbs_scriptComponent	Renders the container that has script component in page editor
rbs_READONLY	Renders the text read only

## Link Styles

The following CSS classes define Rollbase link styles.

CSS Class	Description
rbs_BacktoCol	Renders the Back to List links
rbs_BoldNavyLink	Render a bold link in navy color (used to display the portal main link)
rbs_calNext	Renders month navigation links in mini calendar
rbs_highlightLinks	Renders columns that are highlighted yellow
rbs_NewComponentLink	Renders the new component link (New Object, New Template etc) in component pages(objects.jsp, tabs.jsp etc) in Setup Pages
rbs_ReturnToAppLink	Renders the link "return to <appname> application" in Object definition page
rbs_viewSectionLinks	Renders links in a section like in Edit View
rbs_wideSectionLinks	Renders links in a section and fills the entire row like Edit Chart Link.

## Sidebar Styles

The following CSS classes define Rollbase sidebar styles.

CSS Class	Description
rbs_sidebarContent	Renders the column that has the entire sidebar content
rbs_sidebarLink	Render links in side bar

## Field Types

The topics in this section describe field types.

### Text Field

With the Text field type, users can enter any combination of characters. You can optionally define the following properties:

- Default size of HTML Input Field shown in forms (you can override this default as a Page-level property using the Page Editor).
- The maximum allowed length (number of characters).

- Check if this Field allows multiple values for supported languages (for Customers with more than one language assigned, see [Language Support](#) on page 432).
- An input mask to enforce certain kinds of input based on a pre-defined format. Input mask can include:
  - # symbol for numeric character
  - \* symbol for any input character
  - Letters and separators
  - Example: PO: ####-\*\*-####
- Check if you want to apply input mask on-the-fly when text field loses focus during editing.
- Text to be used as the default value for new records.

Note: The maximum allowed length cannot exceed 100 characters. However this limitation does not apply to external databases, see [Using External Tables as Rollbase Objects](#) on page 352.

## Text Area Field

With the Text Area Field type, Users can enter multiple lines of text. You can optionally define the following properties:

- A maximum length of input (number of characters) to accept.
- Default height and width of HTML text area (you can override this default as a Page-level property using the Page Editor).
- Enable a Rich Text Editor so users can enter text as formatted HTML.

Note: The length of text stored in Text Area cannot exceed 80,000 characters.

## Checkbox Field

With the Checkbox Field type, users can select a Checked (true) or Unchecked (false) value. When creating a Checkbox Field, you specify:

- Whether you want the Field to be checked or unchecked by default.
- An optional checkbox for hiding the display label for this Field on UI Pages.
- Additional text to be rendered on the right of checkbox.

Note: If Checkbox is marked as "required" it must be checked in input form, otherwise it will cause a validation error. This feature is rarely used except for some legal applications.

## Decimal Field

With the Decimal Field type, users can enter any number with decimals. You can optionally define the following properties:

- Number of decimal places to display to the right of the decimal point.
- Decimal separator: dot or comma.

- Number to be used as the default value for new records.
- Minimum and maximum allowed values.

## Currency Field

With the Currency Field type, users can enter a dollar or other currency amount. You can optionally define the following properties:

- Currency Format (this setting will only affect formatting of the Field).
- Default value for new records.
- Minimum allowed value for this Field.
- Maximum allowed value for this Field.

Note: On Edit pages Rollbase adds HTML DOM handler to automatically format value of Currency field (according to selected format) when input field loses focus.

When you create a new Currency Field for Objects that support the multi-currency attribute, you can optionally create a counterpart Base Currency Field that will transfer the foreign currency amount into the currency of your bookkeeping.

A screenshot of a configuration interface. It shows a checkbox labeled "Also create 'Base Currency' field to convert this field's value into" followed by a dropdown menu. The dropdown menu is currently set to "US Dollar".

Note: Due to database limitations Currency and Decimal fields cannot hold values larger than 100,000,000,000.0

## Base Currency Field

This Field is only available for objects that support the multi-currency attribute. The Base Currency Field is a dependent Field that transfers the foreign currency amount specified in the selected Currency Field (see above) into the currency of your bookkeeping. You can create this Field as a counterpart for new a Currency Field. To create a Base Currency Field from scratch specify the following:

- Currency Field which holds the original amount in foreign currency.
- Base Currency to transfer to (typically currency of your bookkeeping).

A screenshot of a configuration interface. It shows two dropdown menus. The top dropdown is labeled "Base Currency" and is set to "US Dollar". The bottom dropdown is labeled "Currency Field" and is set to "Amount".

See [Multi-Currency Support](#) on page 214 for more info on multi-currency support in Rollbase.

## Date Field

With the Date Field type, users can enter a date or pick a date from a calendar popup. You can optionally define the following properties:

- Date format as full text or standard abbreviation.
- Whether you want to use the current date, or the current date plus or minus several days, as the default creation date for new records.
- Whether you want to use the current date, or the current date plus or minus several days, as the default update date for changed records.

## **Date/Time Field**

With the Date/Time Field, users can enter a date and a time, or pick a date from a calendar popup. When the user selects a date from the calendar popup, Rollbase enters that date and the current time into the Date/Time Field.

When you create a date/time Field, you can specify whether you want to format the value as full text or as an abbreviation and whether you want the Field to be automatically populated in specified situations.

## **Time Field**

With the Time Field type, users can enter the time (hours and minutes). This Field does not provide any type-specific settings.

## **Email Field**

With the Email Field type, users can enter an email address that is server-side validated to ensure it is in a valid email format.

You can optionally define a maximum length for the email address.

## **Phone Number Field**

With the Phone Number Field type, users can enter a phone number which will be automatically formatted according to specified input mask, like **###-###-####** If input text is longer than the number of # symbols in format remaining characters are not formatted.

## **Password Field**

With the Password Field type, users can enter and securely store passwords for authentication of Portal Users or for any other purpose, such as integration with third-party systems. You can optionally define the following properties:

- Minimum length (number of chartects) password may have.
- Check if password must contain both letters and digits.

For security reasons passwords are encrypted when stored internally. Actual password's value is never displayed on UI pages. However password's value is always accessible through formulas and API.

When user is entering password on New Record or Edit page he/she is required to confirm entered value by re-typing it again in text box below. If password is already stored in the system it is not populated in text boxes for security reasons. Instead there is a note "Password on file" below text boxes. Provide a new value to override password on file. Use "Remove" link to clear password's value without providing a new value.

The form consists of two text input fields: 'Password' and 'Confirm Password'. Below these fields is the text 'Password on file'. To the right of this text are two buttons: 'Remove' and 'Save'.

## Integer Field

With the Integer Field type, users can enter any number (without decimals). This Field offers the same settings (min value, max value, default value) as other numeric input Fields. You can also choose separator (comma, dot, or none) for thousands.

## Percent Field

With the Percent Field type, users can enter a percentage (number including decimals) and a % sign is automatically added. You can optionally define a maximum number of decimal places.

## Picklist Field

With the Picklist Field type, users can select a value from a list you define. Several other Field types (multi-select picklists, radio buttons and group of checkboxes) have a similar configuration.

First, define a list of values, one value per line, for the picklist. Each value represents a selectable item. If you want a particular value to be selected by default for new records, prefix it with {D}. Optionally, at the end of each line, add an integration code for that value (no longer than 10 characters). Formulas and APIs can use this integration code to safely access unique picklist values (e.g., for data import and export).

You can also choose whether to display values sorted alphabetically or in the order entered (default).

In some situations you may want picklists belonging to different Objects to have the same set of values, so Rollbase allows you to share picklists among Objects with the "Shared As" property. You can either choose "Share as New" or use one of the existing shared picklists in your account. In the latter case, the content of the "Values" box is replaced with existing values from the selected shared picklist.

**Important:** Changing text value of picklist item will destroy modified item and create a new item. Existing records will lose assigned value. If you just want to change text of picklist item use "Replace" functionality explained below in "Field Actions" section.

**Important:** Length of picklist item name cannot exceed 100 characters. Length of integration code cannot exceed 40 characters.



## Picklist Multiselect

With the Picklist (Multi-Select) Field type, users can select any number of values from a list of values you define. You define Multi-select picklists in the same way as single-select picklists, with two additional optional capabilities:

- You can mark more than one value as a default value with the {D} prefix.
- You can choose a height (in rows) to determine the number of values visible at any given time. If more than one value is visible the user can use the CTRL key plus their mouse to select multiple items.

Field Label

Picklists may be shared between objects.

Shared As

Enter the list of values for the picklist field below. Each value should appear on its own line.

Values

You can use the "!" symbol to associate an integration code with each value. For example: Accounts Receivable|AR

Use {D} symbol to indicate value selected by default: {D}Medium

Sort values alphabetically, not in the order entered

List Height

Choose the default height (# of visible values) of this selection box.

## Radio Button Field

With the Radio Button Field type, users can select a single choice from a group of radio buttons. You define Radio Buttons in the same manner as picklists, with an additional layout setting: Alignment that determines whether to align the Radio Buttons vertically or horizontally.

## Group Checkbox Field

With the Group of Checkboxes Field type, users can select any number of checkboxes in a group. You define in the same manner as Multi-Select Picklist Fields are defined with an additional layout setting: Alignment that determines whether to align the Group of Checkboxes vertically or horizontally.

## URL Field

With the URL Field type, Users can enter any website address and the URL is displayed as an HTML link. The content of that link will be displayed in a new browser window.

Tip: You can specify "Link Text" to be used as the text representation of this URL. If link text is not specified the URL itself (possibly truncated with ellipses) will be used.

## Auto-Number Field

The Auto-Number Field type is an automatically generated sequential number or code that uses a display format you define. This number is incremented for each new record created. An Auto-number format may also include year, month and day. When you create an auto-number Field you can specify:

- A display format (mandatory) defined by a template. Syntax and examples are presented inline while creating the Field.
- A starting number (mandatory) for the sequence number portion.
- An option allowing you to assign auto-numbers to all existing Object records once this Field is created or updated.
- By default Auto-Number Fields do not allow duplicate values.

Tip: It is common to use Auto-Number Fields in Object definition name templates.

Important: Auto-Number fields are read-only when new record is created. However you can edit value of field on Edit record page.

Important: New number for Auto-Number field is assigned when user saves new record.

## File Upload Field

The File Upload Field type allows users to upload a file attachment that is stored in this Field. The file will be opened in a new browser window when clicked by the user.

You can specify a maximum size for files, from 128KB up to 2MB. In addition, the Advanced Field Properties section allows you to "Index this Field as part of the text search engine," which automatically indexes files uploaded into this Field.

Check "Make files publicly accessible" if you want the content of uploaded file be accessible without checking for a valid login session and user's access rights.

Important: When editing a File Upload or Image Upload file that holds a previously uploaded file, you have an option to delete that file. This action cannot be un-done, even if you cancel the entire record edit operation.

## Image Upload Field

The Image Upload Field type allows users to upload an image file attachment that is stored in this Field.

You can specify a maximum size for images, from 128KB up to 2MB. The image will be displayed on any Page the Field is included in as an HTML <IMG> tag.

When you define an Image Field, you select a maximum file size to accept. You can define optional parameters:

- Image's display width in pixels.
- Image's display height in pixels.
- Text to show when the mouse hovers over the image.

## Shared Image Field

With the Field type Shared Image, you can upload an image to be shared as a merge Field in Pages, Templates and Formulas (including portal Pages). Shared images are also stored and distributed with published applications.

You configure this Field the same way as Image Upload Fields, except that you must upload an actual image file to be shared (128K max) in GIF, JPG, or PNG format.

Tip: Use shared images throughout your apps for icons and logos to enhance the presentation and user experience.

## Formula Field

Formula field values are calculated from a formula and are automatically updated when any fields used in the formula expression change value. You write formula field expressions in JavaScript. Rollbase executes them to compute a dynamic output that can be displayed on the screen, and can be used by other components.

Formula fields can be displayed in views, see [Views](#) on page 130, but cannot be used for sorting, totaling, and filtering of records in Views. Use an [expression field](#) instead to sort, total or filter. Formula fields execute immediately when you create them and upon any update to the values to which they refer, while expression fields only execute after an update operation.

In addition to fields that specify the label and appearance in views and reports, the **New Field** screen for a formula field includes:

- The return type, which can be `Decimal`, `Currency`, `Integer`, `String`, `Boolean`, or `Date`.
- The currency format (only for fields with a `Currency` return type).
- The number of decimal places (only for fields with `Decimal` return type).
- An optional checkbox for hiding the display label of the field on UI Pages.
- The **Formula Editor** for defining the Javascript expression. You can use merge fields from the current object record and related records to return a dynamically computed value. This value can be numeric or strings. Strings can consist of plain text, rich HTML, or JavaScript code that will be executed on the client's browser. For more information about formulas, including examples, see [Formulas](#) on page 172.

Keep the following in mind when working with formula fields:

- To define loops, you can use the `LOOP_BEGIN` and `LOOP_END` tokens, as described in [Iterating through Records](#) on page 162. However, this technique should only be used for a small number of related records. Looping through a large set of records using these tokens can degrade performance. It is better to use a pre-defined function or the Rollbase Query API to obtain the required record. See the list of methods available in the [Query API](#) on page 561.

- All merge tokens are replaced with the exact contents of the corresponding field's value before the formula logic is executed. For example, a text field included by a token `{!myTextField}` with a value of `Hello World` will be used as is without the quotes around it. Always remember to use quotes around any value where JavaScript requires them.

- To use dates:

- In a formula: create a JavaScript `Date` instance by providing the merge token with quotes around it as the parameter:

```
var myDate = new Date("{!myDateField}");
```

- As the return type of a formula: return the full value of the JavaScript `getTime()` method. For example:

```
return myDate.getTime();
```

- When using images and shared images in formulas, wrap them in quotes to avoid errors. For example, the following formula returns a different starred image (representing a star rating from 1 to 5) based on a picklist value. In this example, the tokens `{!star0}` through `{!star5}` refer to hosted images and `{!rating}` refers to a picklist:



```
if ({!rating}==106469)
  return "{!star1}";
else if ({!rating}==106470)
  return "{!star2}";
else if ({!rating}==106471)
  return "{!star3}";
else if ({!rating}==106472)
  return "{!star4}";
else if ({!rating}==106473)
  return "{!star5}";
else
  return "{!star0}";
```

## Expression Field

Expression fields are similar to [formula fields](#). Unlike formula fields, expression fields can be used for sorting, totaling, and filtering of records in views. Expression field values are updated when a record is updated, while formula field values update dynamically with any change.

Expression fields have the following limitations:

- The JavaScript code for expression fields cannot include loops and references to related records. However, you can use the server-side API to query an expression field.
- The expression field value will not be updated when related records are updated.
- Expression field values are updated when the containing record is updated — not when the record is viewed.
- The data type of an expression field cannot be changed.

- Unlike formula fields, expression fields are stored in the database, so they are subject to limitations on the number of columns per object.

## Template Field

A Template Field type is a non-editable Field that is automatically populated from a template expression you define. When you define a template Field, you can use merge Fields and HTML markup from the current Object record and any related records.

Tip: Template Fields (with a hidden display label) can be a good way to add client-side JavaScript to your Application and Portal Pages.

## Document Template Field

A Document Template Field type is a picklist that allows users to assign a document template to use for a specific Object record. You can assign a default value to this Field and have it assigned to all existing records. This Field renders a link to generate a document on the fly based on the selected document template using the record's data. You can also use it in combination with a Create Template Document Trigger, see [Trigger Overview](#) on page 179.

When used as token in email templates, the Document Template Field will generate an email attachment with the file generated from the document template for the current record.

Tip: Use Fields of this type when you want to allow users to select different types of document templates for different Object records. For example, you may allow your sales users to select a different Quote Template based on the type of order being created. Used with a Create Template Document trigger, you can automate the creation of the appropriate type of document on an as-needed basis.

In View mode Document Template presents a link. When clicked, this link opens parsed document template (populated with record's data) in pop-up window. You can specify size of that window in pixels (650 x 600 by default). For HTML templates only you can choose to render them as PDF rather than HTML.

## Email Template Field

The Email Template Field type is a picklist that allows users to assign an email template to use with a specific Object record, similar to Document Template Field type. You can assign a default value to this Field and have this default value assigned to all existing records. This Field renders a link to preview the email template using the record's data. You can also use it in combination with a Send Email trigger and Send Email workflow action, see [Triggers and Workflows](#) on page 179.

Tip: Use Fields of this type when you want to allow users to select different types of email templates for different Object records. For example, you may allow your sales users to select a different Quote Template based on the type of order being created. Used with a Send Email trigger, you can automate an email based on the appropriate email template on an as-needed basis.

## Related Field

The Related Field type enables access to the value of a Field from a related Object. Related Fields allow you to display and use Fields from related Objects. Before you can use a Field from another Object, you must establish an N-to-1 or 1-to-1 Relationship from the current Object to the related Object.

When creating a related Field, select the appropriate related Object and then select the desired Field as shown below:



Related Fields can be used to edit data of related Record if the following conditions are met:

- Option "Display this field in Edit mode on Edit pages" is checked
- Related Record is previously assigned.

Otherwise Related Fields are read-only on Pages.

Note: Modifying related record through Related Field will not invoke ant triggers

## Integration Link Field

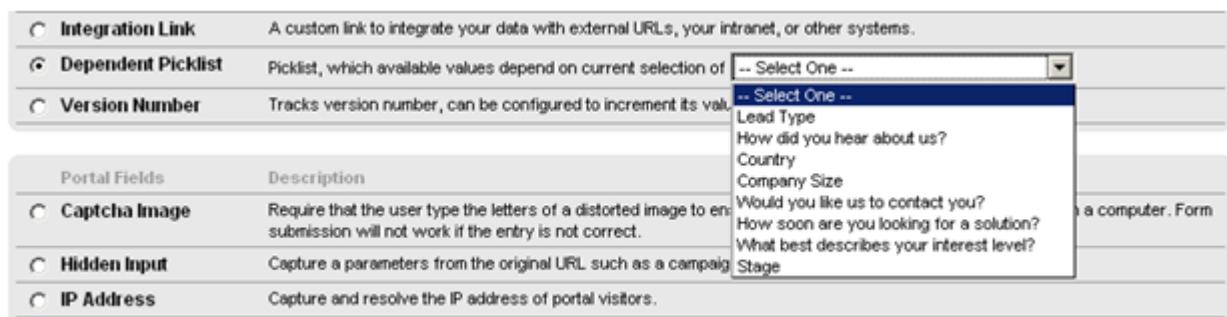
An Integration Link Field type is a custom link, typically used to pass Rollbase data to external systems, your intranet, or other web-based applications by using merge Fields to insert the value of one or more Fields as parameters in the URL. Integration link Fields display as links and they open in a new browser window when clicked.

Integration link Fields work similar to URL Fields, but they allow the use of merge Fields to dynamically construct a URL based on data in the current Object record, versus URL Fields which simply store a hard-coded URL.

Check "Use field's label as link and hide URL", otherwise actual URL will be shown in link.

## Dependent Picklist Field

The Dependent Picklist Field type is a special type of single-select picklist that displays different available values depending on the current selection in another (main) picklist. This main picklist must be selected before creating a dependent picklist.



For each value of the main picklist, you can enter a comma-separated list of available values as shown below.

A typical example of a dependent picklist is a list of States or Provinces that depends on the selection of a Country picklist. Once the selection in the Country picklist changes, Rollbase will automatically update the available list of states.

**Limitation:** Each value in a dependent picklist may belong to only one value from the main picklist. If you need to re-use values in dependent picklists because one or more values needs to be associated with more than one value from the main picklist, you should instead create new Object definitions and use a lookup Field dependency. See the Relationships section below for more information regarding configuring relationship Lookup Fields to use Main and Link lookups to dynamically filter available selection choices.

## Version Number Field

The Version number Field type specifically tracks record-level versions (versus auto-number Fields which are more useful for Object level versioning). Version number Fields assign an integer value to a record when it is created and you can configure them to increment that number whenever the record is updated or cloned.

## Reference Field

With the Reference Field type, Users can reference any object record (typically parent) from current record (typically child). When creating this field select group of object which can be referenced. At run time User can select referenced Object type from drop-down list, then select actual record the same way as in Lookup field (by type-ahead or from pop-up window).



## Advanced Field Properties

In addition to basic Field properties that vary based on the type of the Field, Rollbase provides several advanced Field properties that determine various Field characteristics. Not all of these properties are available for all Field types; for example, Image Upload and Formula Fields cannot be indexed as part of the full text search engine.

- Index this Field as part of the text search engine. This setting is available on text, numeric and date Fields, as well as on file uploads. By enabling this property, this Field will be indexed and added to the global search engine whenever a record is created and updated. Users will then be able to perform global and Object-specific searches on this Field (Field-specific searches in filters, etc, do not require this property to be enabled). If an object definition does not have this option enabled for any field:
  1. Text Index Enabled" box on Object View page is unchecked
  2. Object is not listed in global search drop-down list, see [Search](#) on page 43
  3. "Search" box is not displayed in Selector windows "Keywords" box is not displayed on Search pages
- Track all changes to this Field in each record's Audit Trail for a complete historical log, see [Auditing](#) on page 145. Additional Application Tools for more info on auditing. Enabling this property automatically adds a record-level audit trail entry whenever the Field is updated, showing the value of the Field before and after the update, as well as the user that performed the update and the date and time the update occurred. This option is only available if parent Object has "Audit Trail Enabled" option checked.
- Store values in an encrypted format. This setting is available when new text Field is created (passwords are always encrypted). Values are stored encrypted in the database for added security and to meet compliance requirements. Encrypted fields cannot be used for sorting in Views. Otherwise there is no difference between encrypted fields and regular text fields.
- This Field is required in all forms. This is a global setting specifying whether this Field is always required in form Pages (such as New and Edit). If this property is enabled, it is not possible to make this Field non-required at the Page level using the Page Editor.
- Do not allow duplicate values in this Field. This setting is available only for text-based (including auto-number) Fields. If set, it enforces uniqueness of this Field's value across all records of that Object type.

## System Field Types

The following field types are Rollbase system fields.

### Comments System Field

The Comments Field type is a text area which can be added to Status Change and other pages. Unlike regular text areas Comments field stores value in Comments table: every time new value is entered new comment record is created.

Comments field can be used in templates and formulas. In this case its value is text of the last created comment.

### iCal System Field

The iCal Field type is a read-only Field, automatically generated when you select an Event or Task attribute. It will render a link that you can use to synchronize your event or task with Microsoft Outlook or other programs using export in iCal format.

When used as token in email templates, the iCal Field will generate an email attachment with a file in iCal format.

## **vcard System Field**

The vCard Field type is a read-only Field, automatically generated when you select the Contact attribute. It will render a link that you can use to synchronize your contact with Microsoft Outlook or other programs using export in vCard format.

When used as token in email templates, the vCard Field will generate email attachment with a file in vCard format.

## **Organization Data System Fields**

Tree Fields of this type (Location, Department and Function) are automatically created when you select the "Organization" attribute for an Object. These Fields keep track on location of your record relative to LDF trees. See [Location/Department/Function Permissions](#) on page 402.

If you check "Use user's Function value as default for new records" for this Field type, LDF attribute from user will be copied by default to newly created record.

## **Time Zone System Field**

This field is used to select time zone for Rollbase users. Selected time zone will be used to calculate offset for date/time values before displaying them. Choose from the list of existing time zones. If no time zone is selected - time zone setting from Account info (Customer settings) will be used.

You can check "Use browser's settings to suggest time zone for new records" on field edit page. Please note that suggested time zone will have the same offset value, but it not necessarily will be actual user's time zone.

## **User Role Field**

This field is only available for system "User" object and used to select user's role, see [User Roles and Permissions](#) on page 393. On Field Edit page you can select role which will be used as default for new users ("No Access" if no selection is made).

Important: For security reasons User Role can be only assigned by administrator. For all non-admin users "User Role" field is displayed as read-only.

## **LDF Filter Field**

This Field type is used to visualize LDF permissions for a particular record and user.

## **Parent Object Field**

This Field is a part of Communication Log system object and points to a record to which particular Log refers. This is a required field. The system automatically populates it when new Communication Log record is created.

## **Tag Field**

This field displays search tags assigned for a particular record by different users.

## Portal Field Types

The following Field types are typically only used in external-facing Portals. See [Rollbase Portals](#) on page 257.

### Captcha Image Field

The Captcha Image Field type requires that the user type letters and digits displayed in a distorted image to ensure that the user is in fact a human, rather than a computer. Form submission will not work if the entry is not correct. For example, the figure below shows how a Captcha Image might appear in a Portal Page. See [Rollbase Portals](#) on page 257 for more information.



### Hidden Input Field

The Hidden Input Field type allows you to capture any URL parameter, such as a campaign id or source, from the original URL the visitor used to reach your Portal. You also have an option to capture HTTP cookie instead of URL parameter.

### IP Address Field

The IP Address Field type provides a convenient way to capture, store and resolve the IP address of Portal users. This Field is particularly useful in Portals that require authentication and allow self-registration, see [Rollbase Portals](#) on page 257. This Field will also resolve to display the country code of the IP address, allowing you to receive real information about the source of each of your requests.

---

## Getting Help

---

If you need help with Rollbase, try the following resources:

- For problems logging in, see the FAQs posted on:<http://www.progress.com/products/pacific/resources/faqs>.
- For help with technical issues, see The Progress Technical Users Community Forum
- For customer service, billing and licensing questions, see the contact information posted on:<http://www.progress.com/products/pacific/help>

