



**Sybase Unwired WorkSpace - Mobile
Business Object Development**

Sybase Unwired Platform 2.1

DOCUMENT ID: DC01283-01-0210-02

LAST REVISED: October 2011

Copyright © 2011 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Develop Mobile Business Objects	1
Product Task Flow	1
Tutorials	2
Samples	3
Documentation Roadmap for Unwired Platform	3
Developer Task Flow	5
Understanding the Unwired Platform Development Environment	5
Understanding Fundamental Mobile Development Concepts	6
Learning Unwired WorkSpace Basics	6
Basic and Advanced Developer Profiles	8
Mobile Business Objects	9
Data Sources	9
Device Application Types	10
Deployment to Unwired Server	11
Data Synchronization and Data Refresh	11
Starting and Stopping Unwired Platform Components ...	31
Starting Sybase Unwired WorkSpace	31
Starting Unwired Platform Services	32
Development Postinstallation Tasks	32
Configure	33
Configure - Eclipse Development Environment	33
Creating a Data Source Connection Profile	33
Creating a Sybase Unwired Server Connection Profile	50
Preferences	52
Importing and Exporting Connection Profiles and Projects	60
Importing the Public Certificate	63

Using Unwired Server in a Development Environment	67
Develop	69
Developing a Mobile Business Object	69
Mobile Business Objects	70
Creating Mobile Business Objects	78
Binding Mobile Business Objects to Data Sources	85
Working with Mobile Business Objects	118
Modifying Mobile Business Object Properties ...	119
Mobile Business Object General Properties	119
Mobile Business Object Data Properties	125
Mobile Business Object Mobility Properties	172
Packaging and Deploying Mobile Business Objects ..	217
Deploying a Mobile Application Project	217
Creating a Mobile Deployment Package	218
Deploying Mobile Deployment Packages while Creating a Deployment Profile	221
Deleting a Mobile Deployment Package	228
Deleting a Deployment Profile	228
Viewing Deployment Errors	228
Managing Deployed Packages and Mobile Business Objects	229
Develop a Device Application	230
Generating Object API Code	230
Eclipse Basics	237
Opening a Perspective	237
Perspectives	237
Perspective Shortcut Bar	238
Rearranging Views in a Perspective	238
Moving the Perspective Shortcut Bar	239
Resetting an Active Perspective to its Default Appearance	240
Resetting an Inactive Perspective to its Default Appearance	240

Opening a View	241
Views	241
Detaching a View	242
Floating a View	242
Creating a Fast View	243
WorkSpace Navigator	244
Enterprise Explorer	247
Editors	247
Resources	248
Help	251
Help Features	251
Searching the Help	253
Navigating the Help	254
Opening the Online Help Bookshelf	255
Searching all Documentation Sets	255
Narrowing a Search	256
Search Keyboard Shortcuts	257
Setting Help Display Preferences	257
Troubleshoot	259
API Documentation	261
Glossary: Sybase Unwired Platform	263
Index	275

Contents

Develop Mobile Business Objects

Use Sybase® Unwired WorkSpace to develop mobile business objects (MBOs), and generate Object API code that can be used to create native device applications and mobile workflows.

Mobile business objects help form the business logic for mobile applications and mobile workflows by defining the data you want to use from your backend system and to expose through your mobile application or workflow, and the methods and operations to perform.

See *Supported Hardware and Software* for the most current version information.

See *Fundamentals* for high-level mobile computing concepts, and a description of how Sybase Unwired Platform implements the concepts in your enterprise.

Product Task Flow

Use Sybase® Unwired Platform to develop mobile applications, and to manage the production environment. Understanding the end-to-end product task flow enables you to use Unwired Platform strategically in your enterprise.

Developers Use Sybase Unwired WorkSpace to develop mobile applications. The developer's license includes everything necessary to develop and test your creations—access to sample or external data sources, access to the Eclipse development environment, API classes, and Unwired Server. The basic steps for creating a mobile application include:

1. Create a connection profile to a structured or unstructured data source.
2. Create a connection profile to Unwired Server.
3. Create mobile business objects.

Use Unwired WorkSpace to create a project container, then create one or more mobile business objects (MBOs). Mobile business objects contain the business logic, operations (create, update, delete, and other), attributes, and relationships for the mobile application, and identify synchronization keys and set up personalization. For example, an MBO may include the business logic for creating, editing, and deleting customer records. You can create an MBO by dragging and dropping an object from the data source, or using the creation wizard and then bind the MBO to a data source. Alternatively, you can create an MBO and defer binding to a data source, or create a local business object. Generate replication-based or message-based code, and deploy to Unwired Server.

4. Create device applications or mobile workflows:
 - a. Generate client object API code in Eclipse, then develop the device application in a native IDE. Implement error handling.
 - b. Use Mobile Workflow Forms Editor to develop a message-based workflow package.

Develop Mobile Business Objects

5. Deploy the mobile application project from Sybase Unwired WorkSpace to Unwired Server.
6. Deploy the device application (which contains MBOs) to an emulator or mobile device, and test.

System Administrators Use the Sybase Control Center administrative console, a Web-based user interface to configure and deploy mobile applications and workflow packages from Unwired Server to the production environment, and to manage the production environment. Multiple users can use the administrative console. Steps for deploying the mobile application in a production environment include:

1. Configure the mobile application for deployment into the production environment.
Make any configuration changes necessary, such as switching from a development or test database to the production database.
2. Deploy the mobile application package.
Once configured, deploy the mobile application package. Once deployed, users can access the mobile application from mobile devices. Mobile applications can be pushed to the device or scheduled for deployment. Unwired Server manages synchronization between the data source and the mobile device.
Optionally use Afaria® with Unwired Platform to provision mobile applications, and manage devices and users. You can purchase Afaria separately to further enhance the management of your mobile enterprise.

Mobile Device Users Use mobile devices (including smartphones, laptops, handheld devices, and notebooks) to access mobile applications.

From the mobile device:

- Log in to a mobile application; navigate the user interface; synchronize data and applications through Unwired Server to the data source; and create, update, and delete data records and transactions.
- Use mobile workflow forms to carry out steps in a business workflow process from the mobile device.

See also

- *API Documentation* on page 261

Tutorials

Tutorials demonstrate how to use Sybase Unwired Platform tools to mobilize your enterprise, using step-by-step instructions. They are available on Product Documentation, and SAP™ Development Network (SDN).

Check the Sybase Product Documentation Web site regularly for updates: access <http://sybooks.sybase.com/nav/summary.do?prod=1289>, then navigate to the most current version.

Check SDN regularly for tutorials, articles, and projects: <http://www.sdn.sap.com/irj/sdn/mobile?rid=/webcontent/uuid/40ea4956-b95c-2e10-11b3-e68c73b2280e>.

See also

- *Developer Task Flow* on page 5

Samples

Sample applications are fully developed, working applications that demonstrate the features and capabilities of Sybase Unwired Platform.

Check the SAP Development Network (SDN) Web site regularly for new and updated samples: <https://cw.sdn.sap.com/cw/groups/sup-apps>.

See also

- *Developer Task Flow* on page 5

Documentation Roadmap for Unwired Platform

Learn more about Sybase® Unwired Platform documentation.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role. *Fundamentals* is available on Production Documentation.

Check the Sybase Product Documentation Web site regularly for updates: access <http://sybooks.sybase.com/nav/summary.do?prod=1289>, then navigate to the most current version.

Develop Mobile Business Objects

Developer Task Flow

Get started using the Unwired WorkSpace development environment to mobilize your enterprise.

This describes a typical task flow creating a mobile application using a data source and mobile business objects to create the server-side code. You can follow similar task flows to create mobile applications using a variety of data sources (like a Web service, database, or SAP®), and using other client interface tools (like the client object API).

Understanding the Unwired Platform Development Environment

This provides basic information for understanding the Unwired Platform development environment.

Developing with Multiple Tools

Tools within Sybase Unwired Platform help you to move through the MBO development, MBO deployment, monitoring and management, and device application code generation and customization processes.

Supported Tooling Environments

Unwired Platform provides an Eclipse development environment. The Unwired WorkSpace environment works like a plugin for Eclipse, and provides development tools.

Use backend integration models to connect to your enterprise data and create the business logic, then generate device application code for BlackBerry, Windows Mobile, and Windows. Develop the user interface in its native integrated development environment (IDE).

Use Unwired WorkSpace to develop message-based Workflow clients.

Emulators and Simulators

Use installed versions of emulators and simulators specific to the device IDE to which you are deploying to test your device applications.

Command Line Utilities for Development

Command line utilities support development outside of the user-interface-based tools.

Understanding Fundamental Mobile Development Concepts

This provides basic information for understanding mobile development using Unwired Platform.

Learning Unwired WorkSpace Basics

If you are already familiar with Eclipse, you will find Sybase Unwired Platform features are well integrated. If you are not familiar, you can quickly learn the basic layout of Unwired WorkSpace and the location of online help.

- From the Welcome page, select the **Development** icon to learn about the tasks you must perform. To close this page, click the **X**.
- You can reopen the Welcome page by selecting **Help > Welcome**.
- From Sybase Unwired WorkSpace, look at the area (window or view) that you will be working in to access, create, define, and update mobile business objects(MBOs).

Window	Description
WorkSpace Navigator view	This view displays mobile application project folders, each of which contains all project-related resources in subfolders, including MBOs, data source references to which the MBOs are bound, personalization keys, and so on. Use this view to review and modify MBO-related properties.
Enterprise Explorer view	A window that provides functionality to connect to various enterprise back-end systems; for example, database servers, SAP servers, and Sybase Unwired Server.

Window	Description
Mobile Application Diagram	<p>The Mobile Application Diagram is a graphical editor where you create and define mobile business objects.</p> <p>Use the Mobile Application Diagram to create MBOs (including attributes and operations), then define relationships with other MBOs. You can:</p> <ul style="list-style-type: none"> • Create MBOs in the Mobile Application Diagram using Palette icons and menu selections – either bind or defer binding to a data source, when creating an MBO. For example, you may want to model your MBOs before creating the data sources to which they bind. This MBO development method is sometimes referred to as the top-down approach. • Drag items from Enterprise Explorer and drop them (drag and drop) onto the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the data source being dropped on the Mobile Application Diagram. This is a convenient mechanism for a bottom-up approach. <p>Each new mobile application project generates an associated mobile application diagram.</p>
Palette	<p>The Palette is accessed from the Mobile Application Diagram and provides controls, such as the ability to create MBOs, add attributes and operations, and define relationships, by dragging-and-dropping the corresponding icon onto the Mobile Application Diagram or existing MBO.</p>
Properties view	<p>Select an object in the Mobile Application Diagram to display and edit its properties in the Properties view. While you cannot create an MBO from the Properties view, most development and configuration is performed here.</p>
Outline view	<p>Displays an outline of the file that is currently open in the editor area, and lists structural elements. The contents are editor-specific.</p>

Window	Description
Problem view	Displays problems, errors, or warnings that you may encounter. This is a valuable source for collection troubleshooting information.
Error Log view	Displays error log information. This is a valuable source for collecting troubleshooting information.

- To access the online help, select **Help > Help Contents** from the main menu bar. Expand any of the documents that appear in the left pane. Some documents are for Sybase Unwired Platform, while others are for the Eclipse development environment.

See also

- *Basic and Advanced Developer Profiles* on page 8
- *Mobile Business Objects* on page 9
- *Data Sources* on page 9
- *Device Application Types* on page 10
- *Deployment to Unwired Server* on page 11
- *Data Synchronization and Data Refresh* on page 11

Basic and Advanced Developer Profiles

Optionally you can set Basic and Advanced developer profile preferences for Unwired WorkSpace. You can select the specific features to enable or disable viewing from each of the profiles. Features that are disabled are grayed out. You can also right-click in Mobile Application Diagram, and select **Switch Developer Profile > Basic/Advanced** to switch.

- **Basic** – is a subset of the features available to the Advanced developer, and allows you to develop and deploy MBOs. Customize the Basic profile so that you see only required properties, wizards, screens, and so on.
- **Advanced** – includes all Unwired WorkSpace features, wizards, and properties, enabling additional MBO customization not provided in the Basic profile.

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Mobile Business Objects* on page 9
- *Data Sources* on page 9
- *Device Application Types* on page 10
- *Deployment to Unwired Server* on page 11
- *Data Synchronization and Data Refresh* on page 11

Mobile Business Objects

A mobile business object (MBO) is derived from a data source, and helps form the business logic for mobile applications. MBOs are grouped in Mobile Application Projects, and then the projects are deployed to an Unwired Server and referenced in mobile devices (clients).

The MBO construct is the representation of the entity model as defined within the enterprise data sources. The MBO abstracts the enterprise information system (EIS) managing the data, and the on-device access to the EIS data. Several MBO specializations include:

- Local business object construct – allows modeling of entities with no binding to a data source in the enterprise, and abstracts the on-device persistence and access.
- Structure construct – allows the modeling of arbitrary complex (nested) types, which are used to model an operation interface with complex arguments.

Multiple MBOs can be generated from a single EIS read operation for Web services that have multiple XSLTs defined, or a SAP BAPI/RFC operation that has multiple output tables.

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Basic and Advanced Developer Profiles* on page 8
- *Data Sources* on page 9
- *Device Application Types* on page 10
- *Deployment to Unwired Server* on page 11
- *Data Synchronization and Data Refresh* on page 11

Data Sources

A data source is the enterprise information system (EIS) where data is retrieved from and transactions are executed. A connection profile is a design-time connection to a data source. Connection profiles are created to specific data sources by providing connection information such as host, port, login, and password among others. The connection profiles are used to define MBOs and operations, and mapped to existing, or used to create new, server connections when the package is deployed to Unwired Server.

Unwired Platform hides the interaction complexity with datasource-specific protocols, such as JDBC™ for database and SOAP for Web services.

Unwired Platform currently supports multiple EIS connection types. See *Supported Hardware and Software* for information.

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Basic and Advanced Developer Profiles* on page 8
- *Mobile Business Objects* on page 9

- *Device Application Types* on page 10
- *Deployment to Unwired Server* on page 11
- *Data Synchronization and Data Refresh* on page 11

Device Application Types

Sybase Unwired Platform supports two mobile business object-based application types: native application and Hybrid Web Container-based mobile workflow.

Native Application

The native application model enables the developer to write custom code (C#, Java, or Objective-C, depending on the platform) to create a device application. In native application development, the application is based on compiled code that is specific to a particular mobile operating system. Native application development provides the most flexibility in terms of leveraging the device services, but each application must be provisioned individually after being compiled, even for minor changes or updates. Native applications support offline capabilities, leveraging synchronization.

Hybrid Web container-based mobile workflow

The Hybrid Web Container offers a fast and simple way to build applications that support business processes, such as approvals and requests. With the Hybrid Web Container-based development, the server-side of the application is metadata-driven and the client-side of the application is a fully generated Web application package. This mobile workflow package of platform-independent HTML, JavaScript and CSS resources can be deployed automatically to the Container, a native application on the device, without writing any code. The Hybrid Web Container hosts an embedded browser and launches the individual mobile workflow applications. The workflows are assigned to users by administrators. Once assigned, those workflows can be initiated by the user (client-initiated) or automatically triggered as a result of a back-end event that is sent to the Unwired Server as a data change notification request (server-initiated).

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Basic and Advanced Developer Profiles* on page 8
- *Mobile Business Objects* on page 9
- *Data Sources* on page 9
- *Deployment to Unwired Server* on page 11
- *Data Synchronization and Data Refresh* on page 11

Deployment to Unwired Server

Deploy mobile business objects (MBOs) in a Mobile Application or deployment package to Unwired Server.

The deployment package includes everything needed for the MBO to work in a production environment, including server-side artifacts that support the enterprise information system (EIS) connection and device application, and any other MBO functionality.

The production system administrator can then deploy the package to the production environment.

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Basic and Advanced Developer Profiles* on page 8
- *Mobile Business Objects* on page 9
- *Data Sources* on page 9
- *Device Application Types* on page 10
- *Data Synchronization and Data Refresh* on page 11

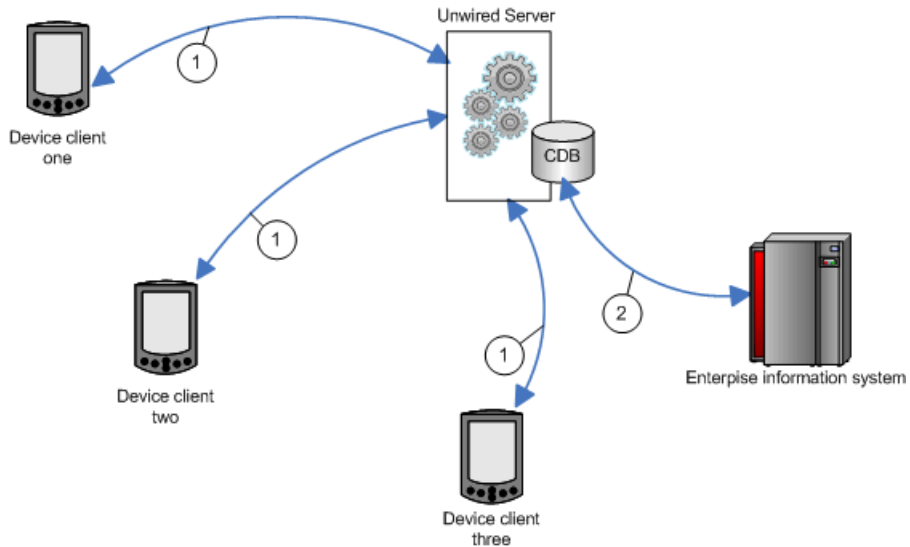
Data Synchronization and Data Refresh

Since dataset variations occur between multiple clients and the enterprise information system (EIS) data to which mobile business object (MBO) data is bound, synchronization is required to reconcile differences and bring each client into coherence with the working copy of the EIS data maintained in the Unwired Server cache database (CDB), before writing updates back to the EIS.

These terms describe maintaining data consistency:

- Synchronization – synchronize between the CDB and mobile-device applications. Synchronization transactions require a connection. If a mobile device does not have a connection to Unwired Server, synchronization cannot occur until a connection is established. However, data updates are aggregated and synchronized when a connection becomes available.
- Data refresh – also called cache refresh, synchronize between the CDB and an EIS. Because information is held in the CDB, even if the EIS server fails, the device still has read access to the data in the CDB.

Developer Task Flow



1. Each client maintains one instance of the data. Similarly, there is only one version of the dataset in the CDB, and only one version in the EIS system.
2. Since variations occur between the different clients and the EIS data, synchronization brings each client into coherence with the working copy of the EIS data that is maintained in the CDB.

See also

- *Learning Unwired WorkSpace Basics* on page 6
- *Basic and Advanced Developer Profiles* on page 8
- *Mobile Business Objects* on page 9
- *Data Sources* on page 9
- *Device Application Types* on page 10
- *Deployment to Unwired Server* on page 11

Unwired Server Cache

The Unwired Server cache is the replicated data store component of the cache database (CDB) and is the integration point for synchronization and data refresh. It manages synchronized data between Unwired Server and device applications, and refreshed data between Unwired Server and the enterprise information system (EIS). Administrators can control cache behaviours in Sybase Control Center.

The cache performs a number of important functions, including:

- Maintaining a local copy of enterprise data.
- Managing updates between the CDB and the EIS servers (data refresh).

- Managing updates between CDB and device application data (synchronization), even in environments where there are thousands of simultaneous synchronizations.
- Partitioning of data – partitions for MBO data, for example, based on client parameters. When a device application passes a client parameter used for both synchronization and data refresh, the CDB:
 - Tracks rows under different partitions based on the synchronization parameter values. A synchronization parameter maps to an attribute that acts as a filter or variable that lets you limit the data that is returned to the device to rows in the table based on a supplied value.
 - Keeps track of which partitions each client is interested in from prior synchronizations. For example, Unwired Server knows that client one is only interested in rows containing the "ABC" parameter value, while client two cares only about rows that contain "def".

Many of the complexities of maintaining synchronization and data refresh are transparent to the administrator, however, you can:

- Configure a dedicated CDB to run within a cluster. A cluster can have any number of Unwired Servers, but only one CDB. If you have a dedicated CDB within a cluster, it should be the first node of the cluster.
- Increase the number of worker threads dedicated to the CDB as the number of Unwired Server instances in a cluster increases.
- Modify CDB port number.
- Monitor synchronization and data refresh performance.

See also

- *Synchronization and Data Refresh Triggers* on page 13
- *Synchronization and Data Refresh Data Flow* on page 16
- *Data Refresh Data Flow* on page 20
- *Mobile Workflow Data Flow* on page 23
- *Synchronization and Data Refresh Strategies* on page 24

Synchronization and Data Refresh Triggers

Initiate synchronization and data refresh using a combination of methods to effectively meet mobile application and system requirements.

See also

- *Unwired Server Cache* on page 12
- *Synchronization and Data Refresh Data Flow* on page 16
- *Data Refresh Data Flow* on page 20
- *Mobile Workflow Data Flow* on page 23
- *Synchronization and Data Refresh Strategies* on page 24

Synchronization Triggers

Define synchronization through mobile business object (MBO) and device application configuration and programming, or after deployment, through Unwired Server settings.

Table 1. Synchronization methods and triggers

Method	Description
Push	<p>For the push method, either the the MBO developer or the administrator configures synchronization timing (on-demand or scheduled). Typically a refresh schedule or data change notification (DCN) is paired with a subscription template for a given MBO with push synchronization enabled. When MBO data in the CDB changes:</p> <ul style="list-style-type: none"> • Notifications are sent at a set interval. The default is one minute and ends when the client acknowledges it has received notification. • Unwired Server determines when individual clients need to be notified of changes and can override device application settings and synchronize the device application with the contents of the CDB. • If Unwired Server does not force a synchronization, device application logic determines how to respond to the push notification. The device application developer can: <ul style="list-style-type: none"> • Register to receive push notifications from Unwired Server. • Implement a push listener. • Implement logic to react to push notifications. For example, if certain data changes the device application synchronizes with the CDB. <hr/> <p>Note: Unwired Server initiates notifications only for replicated-based synchronization, while messaging-based synchronization pushes data to the device without notification.</p>

Method	Description
Pull	<p>For the pull method, the MBO developer configures the amount of data that is to be synchronized (through a combination of settings such as Synchronization group, synchronization parameters, data filter, and so on), and the device application developer adds the screens and logic that allows a user to pass synchronization parameters and attributes to trigger and control synchronization, including:</p> <ul style="list-style-type: none"> Starting the device application – this can automatically trigger synchronization. Adding a synchronization event button to the device application – allows the device application user to synchronize based on how the synchronization event is configured. For example, the MBO developer could include a synchronization parameter that filters data displayed by the device application, or supply client parameters (username and password) by which the device synchronizes. Device application logic – the device application developer adds logic that triggers synchronization based on an event.

Data Refresh Triggers

Define data refresh through mobile business object (MBO) settings, programmatically through the data change notification (DCN) interface, or through Unwired Server settings.

Table 2. Data refresh methods and triggers

Method	Description
DCN (development) Push Listener (administration)	<p>The enterprise information system (EIS) developer implements DCN using HTTP or HTTPS GET or POST methods. Depending on which is implemented, the administrator needs to configure the push listener synchronization gateway for the correct encrypted or unencrypted protocol chosen. The DCN can be initiated by a database trigger, stored procedure, or some other event to:</p> <ul style="list-style-type: none"> Notify Unwired Server that a particular MBO in the CDB needs to be refreshed. Allow the EIS to invoke a particular MBO operation with a set of specified parameters.
Cache group	<p>The MBO developer defines any number of cache groups to which one or more MBOs are added based on data refresh requirements. An update policy applies to all MBOs within a cache group.</p>
Cache update policies	<p>The MBO developer can add a cache update policy to create, update, or delete operations to control how any EIS affecting operation is applied to the CDB.</p>

Method	Description
Load parameters	The MBO developer can create an MBO that uses load parameters to: <ul style="list-style-type: none"> • Control data refresh for individual MBOs • Create CDB partitions for individual users based on parameter values. • Control synchronization if paired with a synchronization parameter.

Synchronization and Data Refresh Data Flow

The way in which data flows through Unwired Server, the enterprise information system (EIS), and device applications depends on the choices you make during design and development.

Sybase Unwired Platform supports both replicated and nonreplicated data within device applications. Except as noted, all references in the synchronization and data refresh overview refer to replicated data flow.

See also

- *Unwired Server Cache* on page 12
- *Synchronization and Data Refresh Triggers* on page 13
- *Data Refresh Data Flow* on page 20
- *Mobile Workflow Data Flow* on page 23
- *Synchronization and Data Refresh Strategies* on page 24

Synchronization Data Flow

Synchronization is when a device application's data is updated with the contents of the Unwired Server cache database (CDB).

Based on various cache settings, the enterprise information system (EIS) can update or refresh the cache during synchronization. Device application initiated synchronization occurs at the request of the user, through a menu button or triggered programmatically as the result of some application action or timer.

Filtering and Synchronizing Data

Use filters and paramters to synchronize selected subsets of data.

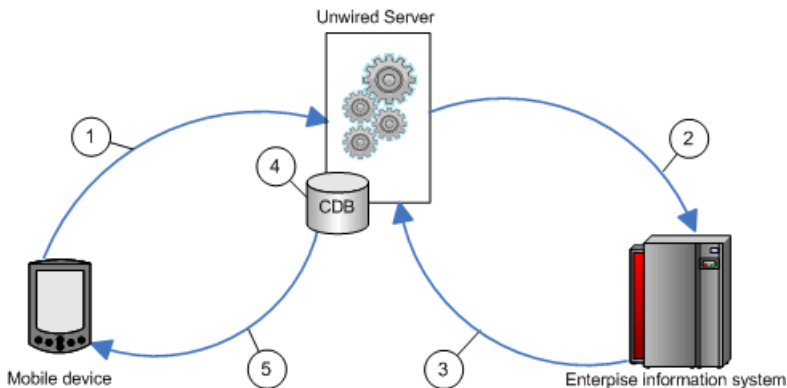
Specifying a synchronization parameter during mobile business object (MBO) development allows you to control the amount and type of data that is returned to the device during synchronization. Without synchronization parameters, large amounts of unnecessary data may be downloaded to devices, making viewing difficult and needlessly expending resources, such as device battery life, memory, and network bandwidth.

Load Parameter Data Flow

Load parameters allow you to limit data stored in the Unwired Server cache and returned to the device based on the values the device user supplies via the parameter over time. They can be paired with synchronization parameters to also control synchronization.

Pairing a load parameter with a synchronization parameter during mobile business object (MBO) development, indicates that the user will supply values for this parameter over time and the aggregate set of data based on the values provided over time are synchronized with that device. If not paired (or mapped) to a synchronization parameter, no such synchronization filtering occurs for the device and the parameter is simply used to update the Unwired Server cache database (CDB) by retrieving a subset of data from the enterprise information system (EIS).

An initial read operation populates a CDB table with all rows of MBO data, which can be included in the data returned in synchronization requests made from one or more clients. In some cases, a load parameter is desired to refine the data requested from the EIS. Mapping the load parameter to a synchronization parameter partitions data in the CDB according to values sent from each device client.

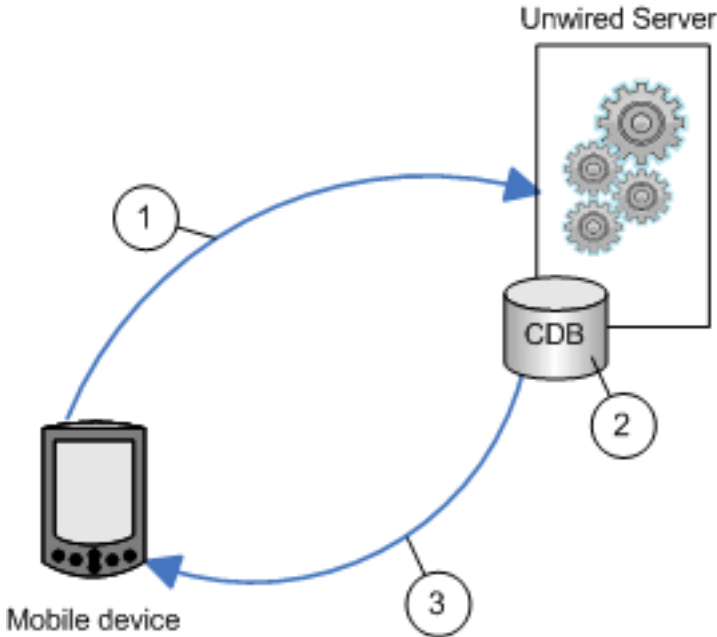


1. The user initiates a synchronization request and includes a parameter value, for example, a user name. Be aware that passwords should not be used as parameters.
2. If personalization keys are used as the load parameter, Unwired Server passes the query to the EIS. If the parameters are user credentials, they are validated by the EIS.
3. The EIS refreshes Unwired Server based on the parameter, for example, it refreshes data only for the validated user. If the parameter is region, and the parameter value is "western," only results for the western region refresh.
4. Unwired Server creates a partition (branch) with the results in the CDB for the validated user, or updates the partition if the user has previously synchronized.
5. Unwired Server synchronizes the device with the data in the CDB partition for that user.

Synchronization Parameter Data Flow

An attribute corresponds to a column in a table. A synchronization parameter maps to an attribute that acts as a filter or variable that lets you limit the data that is returned to the device to rows in the table based on a supplied value.

For example, if a table has a "country" column, a user can supply "USA" as the value in his or her synchronization request. Unwired Server filters and returns only the rows that meets the specified criteria.



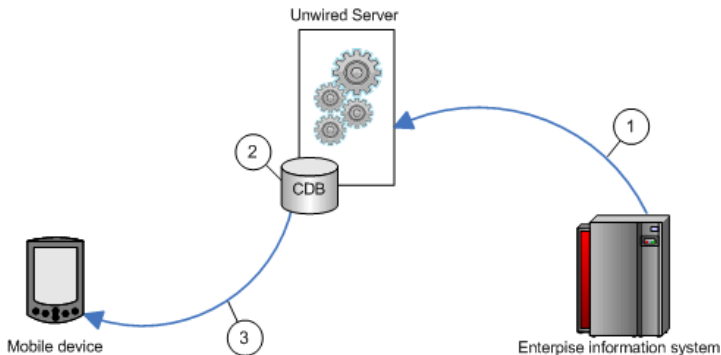
1. The user initiates a synchronization request that includes an attribute value (synchronization parameter).
2. Unwired Server filters the data in the CDB. For example, if the attribute is "country" and the user supplied the value "USA," only rows that contain "USA" are returned to the device.
If the user later supplies the value "Europe", rows for both "USA" and "Europe" are returned to the device, and so on.
3. Unwired Server synchronizes the device with the results.

Result Set Filter Data Flow

A `ResultSetFilter` is a custom Java class deployed to Unwired Server that manipulates rows and columns of data before synchronization.

Result set filters are more versatile (and more complicated to implement) than an attribute filter implemented through a synchronization parameter, since you must write code that

implements the filter, instead of simply mapping a parameter to a column to use as the filter. See *Developers Reference: Server API*.

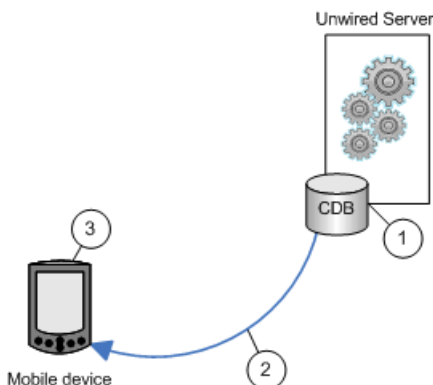


1. Enterprise information system (EIS) data is sent to Unwired Server.
2. The result set filter filters the results, and applies those results to the CDB for a given MBO. For example, the result set filter combines two columns into one.
3. The device application synchronizes with the results contained in the CDB. The client cannot distinguish between MBOs that have had their attributes transformed through a `ResultSetFilter` from those that have not.

Synchronization Initiated by Unwired Server

For replication-based synchronization, you can configure Unwired Server to initiate a push notification to inform users when cached mobile business object (MBO) data changes. Messaging-based applications are inherently capable of sending notifications when the data changes are noted in the CDB.

The Unwired Server administrator schedules notifications to inform registered mobile devices when data changes in the CDB. You can configure Unwired Server to either let device application logic determine if it should synchronize with the changed data, or if configured to do so, override device application logic and force a synchronization.



Developer Task Flow

1. Unwired Server detects a change in the data cache; for example, through a data change notification (DCN) or a data refresh.
2. Unwired Server notifies registered devices of changes to cached MBO data. If it is configured to do so, Unwired Server may force a synchronization with the device; for example, if the data is critical.
3. Implement logic in device applications to appropriately react to push notifications, if the Unwired Server does not force a synchronization.

Data Refresh Data Flow

Data refresh occurs when enterprise information system (EIS) data updates are propagated to the Unwired Server cache database (CDB).

There are two general categories by which data refresh occurs:

- Unwired Server pulls – pulls updates from the EIS either through Unwired Server configuration or policies defined by the MBO developer.
- EIS pushes – a DCN option that includes all information required for an update are pushed to Unwired Server.

See also

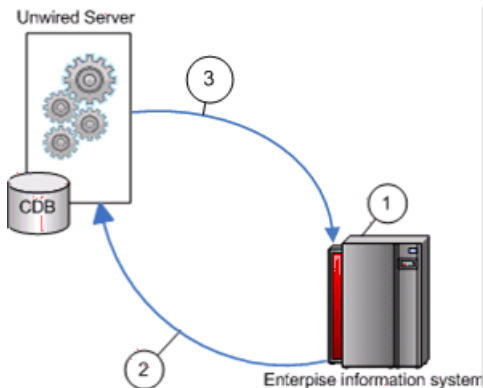
- *Unwired Server Cache* on page 12
- *Synchronization and Data Refresh Triggers* on page 13
- *Synchronization and Data Refresh Data Flow* on page 16
- *Mobile Workflow Data Flow* on page 23
- *Synchronization and Data Refresh Strategies* on page 24

Data Change Notification Data Flow

Data change notifications (DCNs) refresh data when a change to the enterprise information system (EIS) occurs.

DCN requests are sent to Unwired Server as HTTP GET or POST operations. Each DCN can instruct Unwired Server to modify cached MBO data.

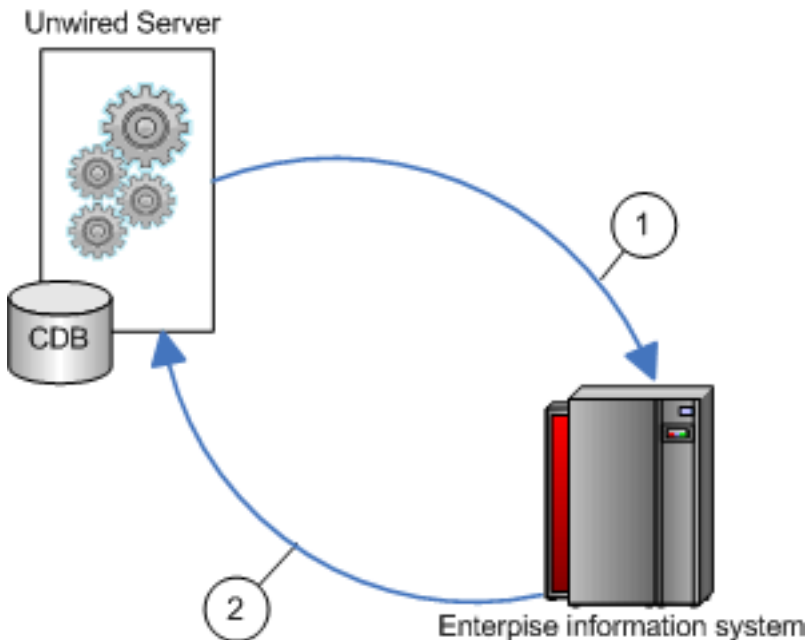
A DCN can be invoked by a database trigger, an EIS event, or an external process. DCNs are more complex to implement than other data refresh methods, but ensure that changes are immediately reflected in the cache.



1. An event initiates the DCN.
2. The DCN (HTTP POST or GET) is issued to Unwired Server.
3. A result response is returned to the EIS.

Cache Group Data Flow

A cache group policy determines the frequency and the level to which mobile business objects (MBOs) belonging to that group are refreshed.



1. The deployed MBO triggers a cache update depending on the cache group to which it belongs.

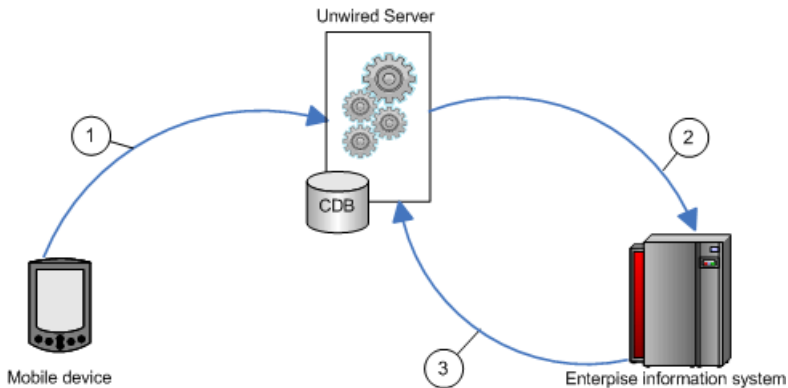
Developer Task Flow

2. The CDB is updated based on the cache group settings.

Cache Update Policy Data Flow

A cache update policy provides a variety of options by which data refresh is controlled for a given create, update, or delete operation, ranging from applying only the results to the cache to invalidating the MBO and refreshing all data (invalidate the cache).

Typically, any EIS data-changing operation invalidates the MBO data to which it is bound, requiring it to be refreshed. This can be inefficient and unnecessary, depending on the nature of the data that changes.

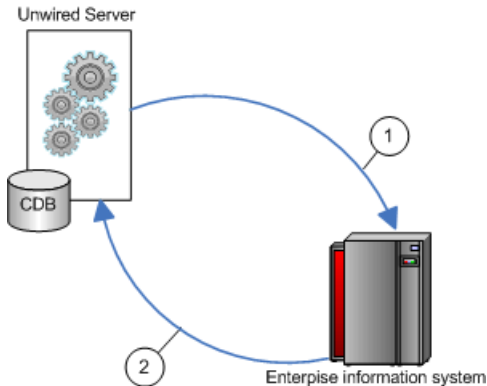


1. The device application initiates a create, update, or delete operation.
2. Unwired Server passes the operation to the EIS, where the operation is executed.
3. The CDB is updated based on the cache update policy. A subsequent synchronization reflects the changed CDB.

Data Refresh Initiated by Unwired Server

Configure Unwired Server to "poll" the enterprise information system (EIS) at scheduled intervals to determine if data has changed. If it has, the EIS refreshes cached MBO data.

The Unwired Server administrator uses the Administration Console to schedule data refresh intervals for a given MBO. This simple and flexible data refresh strategy uses more system resources than data change notification (DCN).



1. Unwired Server polls the EIS at an interval determined by the Unwired Server administrator.
2. If data changes, the CDB is refreshed.

Mobile Workflow Data Flow

In the business workflow model, when you invoke an MBO operation located on Unwired Server using a Submit action, you can specify synchronous behavior (the messaging application waits for a successful or failed response from Unwired Server before proceeding). From an MBO/EIS perspective, Unwired Server updates are asynchronous (the server does not wait for a response).

Some of the differences between messaging mobile applications and replicated mobile applications include:

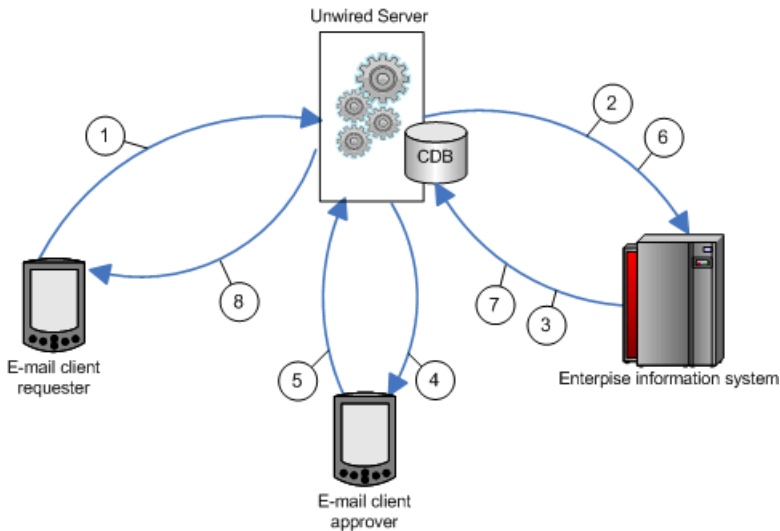
- The MBOs included in the application are no different than any other MBO: same types of parameters, attributes, CDB caching, synchronization methods, and so on.
- There is no permanent storage of the message. For example, a workflow application consists of the MBO portion, which is managed by Unwired Server, and the message portion. The message portion of the application is the transient store and forward system to deliver the messages reliably between server and device client, and takes advantage of the capability to build messages on the fly and send to the interested devices with them having to explicitly know or request it.

This workflow example is a travel approval application that includes:

- A TravelRequest MBO that includes:
 - dates, location, estimated costs, purpose, and a unique ID.
 - status and comment – included in the MBO definition but implemented by the business process widget.
 - An object query that returns a row based on the submitted ID.
- Two triggers:
 1. Sends an message to the approver when a new row is inserted into the MBO table.

Developer Task Flow

2. Notifies the requester when the status of the request has been updated by the approver.
- A business process widget that implements the status and comment portion of the application.



1. An e-mail requesting travel is submitted.
2. Depending on the data refresh schedule or the operation's cache policy, the cache in CDB will be updated.
3. Triggers a message to the approver.
4. The travel request is approved through e-mail.
5. The EIS is updated with the approved information.
6. Depending on the data refresh schedule or the operation's cache policy, the cache in the CDB is updated.
7. The requester receives approval.

See also

- *Unwired Server Cache* on page 12
- *Synchronization and Data Refresh Triggers* on page 13
- *Synchronization and Data Refresh Data Flow* on page 16
- *Data Refresh Data Flow* on page 20
- *Synchronization and Data Refresh Strategies* on page 24

Synchronization and Data Refresh Strategies

Combine synchronization and data refresh techniques and strategies to successfully meet the business needs of the mobile application while effectively utilizing resources.

- Design and planning – carefully consider:

- Limiting data in the mobile business object (MBO) to what is required to meet business needs.
- The size and scope of the mobile application.
- Timing – coordinating device application synchronization with data refresh to achieve optimum results.
- Methods – use a combination of methods to control how much data in the Unwired Server cache is updated when EIS data changes. It is relatively simple to design your system to invalidate MBO data in the CDB and refresh it from the EIS whenever EIS data changes, but this can be inefficient.

See also

- *Unwired Server Cache* on page 12
- *Synchronization and Data Refresh Triggers* on page 13
- *Synchronization and Data Refresh Data Flow* on page 16
- *Data Refresh Data Flow* on page 20
- *Mobile Workflow Data Flow* on page 23

Synchronization Scenarios and Strategies

Mobile application development and administration settings allow you to decide when and how to synchronize and refresh data.

The following table describes strategies to consider when designing and developing your mobile applications. Since most mobile applications include several (if not many) MBOs, you will combine various strategies.

Table 3. Synchronization and data refresh strategies and examples

Scenario	Strategy
Data changes irregularly in the enterprise information system (EIS), and data is noncritical.	Either: <ul style="list-style-type: none"> • Unwired Server initiates a data refresh once a day at off-peak hours that invalidates and refreshes all MBO data, or • The MBO belongs to a synchronization group that includes a daily interval. For example, an organization bulk-loads data by performing a data refresh at the end of the work day. Field service personnel filter an attribute and synchronize (synchronization parameter=region) at the beginning of their day to synchronize only data of interest.

Scenario	Strategy
<p>Clients must immediately synchronize critical EIS data changes.</p>	<p>You could either implement a cache update policy that immediately applies the results of the MBO operation to the CDB. Or,</p> <p>Implement a data change notification (DCN) that performs a data refresh of targeted MBO data. Unwired Server then sends notifications to registered device clients, and optionally force a synchronization to targeted devices.</p> <p>For example, administrators, professors, and others on a college campus have registered their mobile devices with campus police. When an emergency call from campus is received and entered into the system, a DCN updates the cache. Unwired Server forces synchronization with registered devices.</p>
<p>EIS data changes frequently.</p>	<p>Either:</p> <ul style="list-style-type: none"> • Implement an Unwired Server scheduled refresh that polls the EIS at specified intervals and updates the CDB when changes occur, or • Define various cache groups for MBOs that refresh data as necessary. <p>For example, set a short data refresh interval to receive up-to-date quotes for an equity tracking mobile application (stocks, bonds, and so on), which also allows users to buy and sell equities.</p>
<p>Device application users change EIS data frequently.</p>	<p>When defining MBO create, update, and delete operations, include a cache update policy for EIS modifying operations. Choose a policy that updates the CDB only with necessary changes.</p> <p>For example, a mobile application contains regional sales information. When you make a sale to a new customer, the MBO inserts a new row in the corresponding EIS table. To see changes related to your customers only (rows that contain your territory ID), use the apply operations parameter policy when defining the MBO.</p>
<p>A mobile application supports thousands of individual users, each of which has a set of user specific data.</p>	<p>The MBO developer creates personalization keys that are used as client parameters (user_name and password), which are validated by the EIS.</p> <p>For example, a mobile application used for retail sales maintains login information for validated customers that provides access to account information, shopping cart, wish list, and so on.</p>

Scenario	Strategy
A mobile application has both static and changeable data.	<p>The MBO developer configures two cache groups, designed to refresh Unwired Server cache (CDB) for each MBO based on the frequency of EIS data changes to which each MBO is bound.</p> <p>For example, a mobile application contains a sales_order MBO with a many-to-one relationship to the product MBO. While sales_order related data changes often, as sales are made, product data does not. The MBO developer establishes two cache groups:</p> <ol style="list-style-type: none"> 1. The sales_order MBO data cache updates hourly. 2. The product MBO data cache updates daily.
A Human Resources department wants to implement a mobile application used to request and approve travel and expenses.	<p>The message-based mobile application uses both replicated data (managed within the MBO) and nonreplicated messaging data. Unwired Server pushes changes/updates to device application users.</p> <p>For example, the application includes MBO bound data (calendar with requested dates, total cost, and so on). The messaging portion includes additional information including the message. Once requested and e-mailed to the manager, the recipient (manager) approves dates and expenses. The MBOs are updated in the EIS (replicated and synchronized), while the message is not.</p>

The Impact of Synchronization and Data Refresh

When designing and developing your mobile application solutions, consider the impact of varying synchronization and data refresh methods.

Table 4. Impact of various data refresh methods

Data refresh method	Implication
Data change notification (DCN)	Requires a developer familiar with the EIS from which the DCN is sent. Provides more flexibility than a scheduled refresh, but is more complicated to implement. HTTP GET methods are less secure than HTTP POST.
Unwired Server scheduled data refresh	Easily implemented by the Unwired Server administrator. Less targeted than DCN or a cache group. Uses more system resources since it must periodically query the EIS for changes.
Cache group	Easily implemented by the MBO developer. A cache group is a collection of cache tables to which a common refresh policy is applied.

Data refresh method	Implication
Cache update policy	Easily implemented by the MBO developer. Updates the CDB for an EIS data effecting operation (create, update, or delete) based on the policy associated with the operation. For example, you could update a modified row or invalidate and refresh the entire MBO.
Load parameter	Easily implemented by the MBO developer. Load parameters map to data source arguments and refresh data based on the supplied value. They can be used alone or mapped to synchronization parameter to control both data refresh and synchronization.

Table 5. Impact of various synchronization methods

Synchronization method	Implication
Initiated by Unwired Server	<p>Easily implemented by the Unwired Server administrator. The primary consideration is balancing performance with resource usage when determining how frequently synchronization is initiated, and to how many registered devices:</p> <ul style="list-style-type: none"> • Download from Server – control when data is downloaded to remote devices. For example, if set to 10 minutes, the server notifies the client of data updates at most every 10 minutes, even if data changes more often. • Device notifications – notifications continue at a predetermined interval until acknowledged by the registered device, even when devices are disconnected or out of range.
Initiated by the device	<p>Options include:</p> <ul style="list-style-type: none"> • Filtering results – implemented by the MBO and device application developer. The MBO can be configured through load and synchronization parameters to have the CDB maintain partitions for each user, resulting in a growing list of clients who synchronize a filtered data set. • Custom listener – implemented to listen for messages sent by the notifier. When the listener receives a message, it can be programmed to initiate synchronization.

Synchronization method	Implication
Defined within the MBO	Options include: <ul style="list-style-type: none"> • Synchronization group – groups MBOs by synchronization needs. • Synchronization tab – defines synchronization requirements for individual MBOs.

Bulk Loading Cache Strategy

Bulk loads transfer large quantities of data into the Unwired Server cache database (CDB). Because of the demands this places on resources, special consideration needs to be given when using this strategy.

When Used

Bulk loads are typically used when:

- Creating an initial data set of reference data for the device application. Bulk loads are ideal for data that does not change often.
- The structure of EIS data changes and the application data structure needs to be normalized on the device.
- Repairing a complete data set when corrupted on the device or on the CDB.

How Used

For each MBO, the developer selects the largest set of data that is applicable to most mobile users and designs the MBO to load data into the Unwired Server cache all at once. The developer then needs to define synchronization criteria for the MBO, so that synchronization parameters index into the cache to retrieve device-relevant data subsets. Good for data where the device is primarily driving change (work orders) or reference data that does not change often

Considerations

- Preloads the cache up front and isolates specific device invocations from the enterprise application.
- Large up-front cost of data that may never be used. Bulk loads are best performed when minimal amount of users are connected to the EIS server.

Starting and Stopping Unwired Platform Components

Once you have completed the postinstallation tasks for your installation, you may need to start and stop Unwired Platform components during the normal course of operations.

A set of Windows services support Unwired Server. If you did not set these services to start automatically on system start-up, you can change them to start automatically at any time after installation. See *System Administration Guide > System Reference > Unwired Platform Windows Services*.

Starting Sybase Unwired WorkSpace

Start Unwired WorkSpace from the Windows Start menu.

Prerequisites

To ensure that Eclipse starts properly, be sure the PATH environment variable does not include any embedded double quote characters.

Task

If Unwired Server is not running, you can still create and edit MBOs and generation code, but you cannot deploy MBOs.

1. From Windows, select **Start > Programs > Sybase > Unwired Platform<version> > Unwired WorkSpace**.

Create a new workspace for Sybase Unwired WorkSpace Eclipse Edition the first time you launch it.

2. If you cannot start or stop Unwired Platform Server services through the Windows Start menu, see *Troubleshooting Sybase Unwired Platform > Troubleshoot Sybase Control Center for Sybase Unwired Platform > Unwired Server Fails to Start*.

See also

- *Starting Unwired Platform Services* on page 32
- *Development Postinstallation Tasks* on page 32

Starting Unwired Platform Services

Start Unwired Server, the sample database, the cache database (CDB), and other essential services from the Windows Start menu.

In Windows, select **Start > Programs > Sybase > Unwired Platform<version> > Start Unwired Platform Services**.

See also

- *Starting Sybase Unwired WorkSpace* on page 31
- *Development Postinstallation Tasks* on page 32

Development Postinstallation Tasks

Before starting and working with Unwired WorkSpace, configure your environment.

How you configure components will vary depending on your needs.

Table 6. Sybase Unwired WorkSpace postinstallation configuration tasks

Configuration tasks	Where to find more information
<ul style="list-style-type: none">• Creating a connection profile• Connecting to Unwired Server• Setting preferences• Migrating projects and applications• (Optional) Importing the Public Certificate	<p>Unwired WorkSpace online help</p> <ol style="list-style-type: none">1. Start the Sybase Unwired WorkSpace.2. Select Help > Help Contents.3. On the Sybase Unwired Platform bookshelf, select <i>Sybase Unwired WorkSpace - Mobile Business Object Development > Configure > Configure - Eclipse Development Environment</i>.

See also

- *Starting Sybase Unwired WorkSpace* on page 31
- *Starting Unwired Platform Services* on page 32
- *Creating a Data Source Connection Profile* on page 33

Configure

Perform the postinstallation configuration tasks for the Unwired Platform components that compose your system.

Configure - Eclipse Development Environment

Configure your Eclipse Edition development environment by connecting to Unwired Server and your data source, setting preferences, and setting up the tools you need.

Creating a Data Source Connection Profile

A connection profile contains the connection property information needed to connect to a server in your enterprise.

1. In the Enterprise Explorer, right-click one of the following connection categories and select **New**.
 - Database Connections
 - SAP Servers
 - Web Services
 - Rest Web Services
2. Follow the instructions on the wizard pages to create the selected connection.

See also

- *Creating a Sybase Unwired Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing and Exporting Connection Profiles and Projects* on page 60
- *Importing the Public Certificate* on page 63
- *Development Postinstallation Tasks* on page 32

Connection Profiles

A connection profile enables a runtime connection to enable mobile application development.

A connection profile contains the connection property information needed to connect to a runtime instance. It manages the configured connections running on an application server. A connection profile is added through an extension point. When you create a connection profile, you specify standard configuration parameters, such as a connection URL.

See also

- *Creating a Database Connection Profile* on page 34

Configure

- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49
- *Creating a Sybase Unwired Server Connection Profile* on page 50
- *Testing a Connection Profile* on page 48

Creating a Database Connection Profile

Use the Enterprise Explorer to create and manage database connection profiles from a central location.

1. In the Enterprise Explorer, right-click **Database Connections** and select **New**.
2. On the Wizard Selection page, select an option and click **Next**:
 - DB2 for Linux, UNIX, and Windows
 - DB2 for i5/OS
 - DB2 for z/OS
 - Oracle
 - SQL Server
 - Sybase ASA
 - Sybase ASE
3. Follow the instructions on the wizard pages to create the selected database connection profile.

Before creating connection profiles for SQL Server, DB2, and Oracle databases, you must install the appropriate drivers and JAR files.

Configuring Your Environment to Use a JDBC Driver

Download the appropriate JDBC driver and configure your environment to connect to Oracle, DB2, and Microsoft SQL Server 2005 and 2008 databases.

1. Download the driver.

JDBC driver for:	URL
Oracle	http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html
DB2	http://www-306.ibm.com/software/data/db2/express/download.html
SQL Server JDBC driver 3.0	Go to http://msdn.microsoft.com/en-us/data/aa937724.aspx and select <i>Download SQL Server JDBC Driver 3.0</i> .

2. Shut down Unwired WorkSpace and Unwired Platform Services.

3. For Unwired WorkSpace, put the driver in the correct location.

JDBC driver for:	Action
Oracle	Place the JDBC driver, for example <code>ojdbc14.jar</code> , in: <code><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.oracle_<version>.<plugin version number>\lib</code>
DB2	Unzip the <code>db2JdbcJars.zip</code> file and copy the JAR files to: <code><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.db2_<version>.<plugin version number>\lib</code>
SQL Server JDBC driver 3.0	Copy <code>sqljdbc4.jar</code> to: <code><SUP installation root>\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.com.sqlserver_<version>.<plugin version number>\lib</code>

4. Copy the appropriate JAR file to the specified Unwired Server location.

JAR file for:	Action
Oracle	Copy <code>ojdbc14.jar</code> to the server location: <code><SUP Installation root>\Servers\UnwiredServer\lib\3rdparty</code>
DB2	Copy the JAR files, for example <code>db2jcc.jar</code> and <code>db2jcc_license_cu.jar</code> , to the server location: <code><SUP Installation root>\Servers\UnwiredServer\lib\3rdparty</code>
SQL Server JDBC driver 3.0	Copy <code>sqljdbc4.jar</code> to the server location: <code><SUP Installation root>\Servers\UnwiredServer\lib\3rdparty</code>

Note: If you do not copy the JAR files to the server location, you will encounter runtime errors due to the missing JDBC driver.

5. Restart Unwired WorkSpace and Unwired Platform Services.

Configure

See also

- *Creating a DB2 Connection Profile* on page 36
- *Creating an Oracle Connection Profile* on page 37

Creating a DB2 Connection Profile

A connection profile contains the connection property information needed to connect to a DB2 database in your enterprise.

Prerequisites

Before creating the connection profile, copy your DB2 JAR files (for example, `db2jcc.jar` and `db2jcc_license_cu.jar`) to:

```
<SUP-Install-Path>\Unwired_Work-Space\Eclipse  
\sybase_workspace\mobile\eclipse\plugins  
\com.sybase.uep.com.db2_1.5.2.<plugin version number>\lib
```

Task

1. In **Enterprise Explorer**, right-click **Database connections** and select **New**.
2. Select the DB2 database type for which you are creating the connection profile. For example, **DB2 for Linux, UNIX, and Windows**. Enter a name and optional description and click **Next**.
3. Select the **New driver definition** icon adjacent to the **Drivers** field.
4. From the driver templates choose **IBM Data Server Driver for JDBC and SQLJ**.
5. From the Jar list tab click **Edit Jar/zip** and verify that `db2jcc.jar` and `db2jcc_license_cu.jar` files point to `<SUP-Install-Path>\Unwired_Work-Space\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.com.db2_1.5.2.<plugin version number>\lib`.
6. Click **Yes** (from the Update all Jars to use same path prompt), then **OK** on the previous screen (Edit driver definition).
7. Complete the following information.

Table 7. Specify DB2 database connection details page

Field	Description
Database	The name of the DB2 database
Host	The host name on which the database resides.
Port number	The port to which you are connecting. For example, 50000.

Field	Description
Use client authentication	Select this option and enter the user name and password in the corresponding fields to provide basic authentication to the database.
User Name	Enter the name for the database login.
Password	Enter the password for the database login, if required.
Save Password	Select the checkbox to save the password.
Connection URL	Read-only details specifying the connection URL.
Connect when the wizard completes	Select this option to connect to the database upon exiting the wizard.
Connect every time the workbench is started	Select this option if you want Unwired WorkSpace to connect automatically to the database (when it is started) and display its contents in Enterprise Explorer.
Test connection	Click Test connection to ping the database and to verify that the connection profile is working.

8. Click **Next** to see the summary page, or **Finish** to create the connection profile.

The connection profile displays under **Database connections** in Enterprise Explorer.

Creating an Oracle Connection Profile

A connection profile contains the connection property information needed to connect to a Oracle database in your enterprise.

Prerequisites

Before creating the connection profile, copy your Oracle JAR file (for example, `ojdbc14.jar`) to:

```
<SUP-Install-Path>\Unwired_Work-Space\Eclipse
\sybase_workspace\mobile\eclipse\plugins
\com.sybase.uep.com.oracle_1.5.2.<plugin version number>\lib
```

Task

1. In **Enterprise Explorer**, right-click **Database connections** and select **New**.
2. Select **Oracle** as the connection profile type, and click **Next**.
3. Select the **New driver definition** icon adjacent to the **Drivers** field.

Configure

- From the driver templates choose Oracle Thin Driver.
- From the Jar list tab, click **Edit Jar/zip** and verify that `ojdbc14.jar` points to `<SUP-Install-Path>\Unwired_Work-Space\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_1.5.2.<plugin version number>\lib`
- Click **Yes** (from the Update all Jars to use same path prompt), then **OK** on the previous screen (Edit driver definition).
- Complete the following information.

Table 8. Specify Oracle database connection details page

Field	Description
SID	The Oracle System ID (SID) used to identify the Oracle database.
Host	The host name on which the database resides.
Port number	The port to which you are connecting. For example, 1521.
User Name	Enter the name for the database login.
Password	Enter the password for the database login, if required.
Save Password	Select the checkbox to save the password.
Connection URL	Read-only details specifying the connection URL.
Catalog	Select the catalog which allows the user to obtain information about the database.
Connect when the wizard completes	Select this option to connect to the database upon exiting the wizard.
Connect every time the workbench is started	Select this option if you want Unwired Work-Space to connect automatically to the database (when it is started) and display its contents in Enterprise Explorer.
Test connection	Click Test connection to ping the database and to verify that the connection profile is working.

- Click **Next** to see the summary page, or **Finish** to create the connection profile.

The connection profile displays under **Database connections** in Enterprise Explorer.

Creating an SAP Connection Profile

A connection profile contains the connection property information needed to connect to a server in your enterprise.

1. In Enterprise Explorer, right-click **SAP Servers** and select **New**.
2. Enter a name and optional description for the connection profile and click **Next**.
3. Complete the following information:

Table 9. SAP Connection Properties page Connection tab

Field	Description
<p>Create from properties file</p> <p>You can create the connection in two methods: Selecting this option allows you to retrieve connection information from a properties file that contains SAP connection information. Or, you can follow the wizard instructions and input the values manually.</p>	<p>Select Create from properties file and then select Browse to select a *.properties file that contains the SAP server connection information. All fields on the Connection page are populated automatically based on the selected properties file.</p> <p>Example contents are:</p> <ul style="list-style-type: none"> • jco.client.client=800 • jco.client.user=uepusr01 • jco.client.passwd=Sybase123 • jco.client.ashost=i18nsap1 • jco.client.sysnr=01

Field	Description
Application Server	<p>The host name on which the server resides. The default is "localhost".</p> <p>For SAP R3 systems that use a router, application server must have the format:</p> <p>/H/proxyhost/H/application_server</p> <p>where "H" is literal, and proxyhost and application_server are variables that represent the proxy host and application server respectively. For example:</p> <p>/H/sapm20.anr.ms.test.com/S/3299/H/sapm20.anr.ms.test.com</p> <p>http://sapm20.anr.ms.test.com/S/3299/H/sapm20.anr.ms.test.com</p> <p>Note: When connecting to an SAP message server, application server information is not needed, instead only jco.client.mshost, jco.client.gwhost, jco.client.group, and jco.client.r3name properties are required.</p>
System ID	The unique identity of the SAP server.
System Number	The SAP system number.
Client ID	The SAP Client ID. The default is based on the SAP server configuration.
User Name	Enter the name for the server login.
Password	Enter the password for the server login.
Save Password	Select the checkbox to save your password.

Table 10. SAP Connection Properties page Advanced tab

Field	Description
Default Code Page	The default language that the SAP server is using.
Code Page Number	The default is auto-filled and varies depending on the language you select. For example, if you choose English, then the default is 1100.

Field	Description
Language	ISO two-character language code (for example, EN, DE, FR), or SAP-specific single-character language code. As a result, only the first two characters are ever used, even if a longer string is entered. The default is EN. If you select a language from the drop down list, code page number will be auto-filled according to the language.
Other Properties	Select this to set advanced properties.

- Click **Finish** to create the connection profile.

The connection profile displays under SAP Servers in the Enterprise Explorer.

See also

- *Binding an SAP Data Source to a Mobile Business Object* on page 96
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 106
- *Implementing SSO for SAP* on page 87

SAP External Libraries Overview

Understand the purpose of the various external files you can optionally download from SAP and install into Unwired Platform to enable communication with an SAP EIS.

- **SAP Cryptographic Libraries** – required by Unwired Platform to enable Secure Network Communications (SNC) between Unwired Server or Unwired WorkSpace and the SAP EIS.
- **SAP SSO2 Token Libraries** – required only if you use single sign-on (SSO) with SSO2 tokens **and** want to enable persisted token caching. The default, and recommended approach, is to use the Unwired Server authentication timeout interval setting, which does not require these libraries.
- **SAPCAR utility** – required to extract files from the SAP cryptographic library.

Installing the SAPCAR Utility

Unzip and install the latest SAPCAR utility on your Unwired Server or Unwired WorkSpace host, which you can use to extract the contents of compressed SAP files, for example RFC and cryptographic library files.

The installation package is available to authorized customers on the SAP Service Marketplace. There are different distribution packages for various hardware processors. Select the package appropriate for your platform.

- Go to the SAP Web site at <http://service.sap.com/swdc>.

Configure

2. From the SAP Download Center, navigate to **Support Packages and Patches > Browse our Download Catalog > Additional Components**
3. Select **SAPCAR**.
4. Select the current version. For example, **SAPCAR 7.10**, then select and download the SAPCAR appropriate for your platform.

Installing the SAP Cryptographic Libraries on Unwired Platform

Installation and configuration is required if you want to configure Secure Network Communications (SNC) for Unwired Platform SAP JCo connections. SNC may be required by the SAP EIS in question, if SSO2 tokens or X.509 certificates are used for connection authentication.

Prerequisites

Download and install the SAPCAR utility, which is required to extract the contents of the cryptographic library.

Task

Unzip and install the contents of the latest SAP Cryptographic archive on your Unwired Server host. There are different distribution packages for various hardware processors.

Make sure you are installing the correct libraries for your environment, and into folders based on the particular architecture of your machine.

1. Go to the SAP Web site at <http://service.sap.com/swdc> and download the latest SAP cryptographic library suitable for your platform.
 - a) Navigate to **Installations and Upgrades > Browse our Download Catalog > SAP Cryptographic Software > SAP Cryptographic Software**.
 - b) Select and download the platform specific file.
2. Create a directory in which you unzip the Cryptographic zip file. For example: C:\sapcryptolib.
3. Copy the appropriate Windows cryptographic library for your machine (for example, 90000101.SAR) to the C:\sapcryptolib directory.
4. Open a command prompt and navigate to C:\sapcryptolib.
5. Extract the SAR file. For example:
SAPCAR_4-20002092.EXE -xvf C:\90000101.SAR -R C:\sapcryptolib
6. Depending on your platform:
 - 64-bit – delete the nt-x86_64 directory from the C:\sapcryptolib directory. Copy the contents of the nt-x86_64 subdirectory to C:\sapcryptolib.
 - 32-bit – delete the ntia64 and nt-x86_64 directories from the C:\sapcryptolib directory. Copy the contents of the ntintel subdirectory to C:\sapcryptolib. Delete the ntintel subdirectory.

7. Add the SECUDIR environment variable based on machine type and Unwired Platform component:
 - 64-bit Unwired Server – requires access to the 64-bit crypto libraries. Set SECUDIR in <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\bin\userasetenv.bat to point to the location of the 64-bit crypto libraries. For example:

```
set SECUDIR=C:\sapcryptolib
```
 - 32-bit Unwired Server – requires access to the 32-bit crypto libraries. Set SECUDIR in <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\bin\userasetenv.bat to point to the location of the 32-bit crypto libraries. For example:

```
set SECUDIR=C:\sapcryptolib
```
 - Unwired WorkSpace – runs in 32-bit mode regardless of machine type and requires access to the 32-bit crypto libraries. Set SECUDIR in C:\Sybase\UnwiredPlatform\Eclipse\UnwiredWorkSpace.bat above the line SUP_ROOT=<SUP_HOME>. For example:

```
set SECUDIR=C:\sapcryptolib
```

Creating a Web Service Connection Profile

A connection profile contains the connection property information needed to connect to a Web service in your enterprise.

1. In **Enterprise Explorer**, right-click **Web Services** and select **New**.
2. Complete the following information.

Table 11. Web service connection details page

Field	Description
Select the Web service from either a: <ul style="list-style-type: none"> • Local file (default) – browse to a file located on the file system that contains connection information. • Workspace – browse to the WSDL contained in an existing WorkSpace project. • URL – provide a WSDL URL, for example: http://www.ripedevelopment.com/webservices/LocalTime.asmx?WSDL 	Select the location from which you want to find the Web service, then click Browse to locate the Web service. You must check From URL to enable the Enable HTTP Authentication field.
Enable HTTP Authentication	Select the checkbox to enable HTTP authentication during WSDL document retrieval. If you enable HTTP authentication, you must enter a User name and password.

3. Click **Finish** to create the connection profile.

Configure

The connection profile displays under Web Service in the Enterprise Explorer.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

Creating a REST Web Service Connection Profile

A connection profile contains the connection property information needed to connect to a REST Web service in your enterprise.

1. In **Enterprise Explorer**, right-click **REST Web Services** and select **New**.
2. Enter the name and optional description of the connection profile and click **Next**.
3. Complete the following information.

Table 12. REST Web service connection details page

Field	Description
Resource base URL	The common base URL referenced by the connection. typically, REST Web service connections begin with the same base URL. For example, <code>http://www.yourcompany.com/</code> could serve as the common prefix for the resource base URL. A base URL must begin with <code>http:</code> .
Resource URI template	A URI template used by mobile business object (MBO) create, read, update, and delete (CRUD) operations on the REST Web service resources through HTTP operations. The URI template should follow this structure: <code>customers/{id(int)}</code>

4. Click **Next** to view a summary of the connection profile, or **Finish** to create the connection profile.

The connection profile displays under REST Web Service in the Enterprise Explorer.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43

- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

REST Web Services

A REST (Representational State Transfer) Web service is a set of architectural principles by which you design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages.

REST Web services support these key principals:

- Provides an ID (structure-like URIs) for every resource
- Links resources together
- Uses standard HTTP methods
- Supports resources with multiple representations (Transfer XML, JavaScript Object Notation (JSON), or both)
- Communicates statelessly

Representational State Transfer – when called, a representation of the resource is returned, which places the client application in a state. Traversing the returned URL accesses another resource, placing the client application into yet another state. Thus, the client application changes (transfers) states with each resource representation.

REST establishes a one-to-one mapping between mobile business object (MBO) create, read, update, and delete (CRUD) operations and HTTP methods:

- Use POST to create a resource on Unwired Server
- Use GET to retrieve a resource
- Use PUT to change or update the state of a resource
- Use DELETE to remove or delete a resource

Example REST URIs include:

```
http://example.com/customers/1234
http://example.com/orders/2007/10/776654
http://example.com/products/4554
http://example.com/orders/2007/11
http://example.com/products?color=green
```

and this XML code fragment which links the resources together:

```
<order self='http://example.com/customers/1234' >
  <amount>23</amount>
  <product ref='http://example.com/products/4554' />
  <customer ref='http://example.com/customers/1234' />
</order>
```

With this representation, the standard HTTP GET method executes on the first resource:

```
http://example.com/customers/1234
```

Table 13. Generic interface relationship

<interface> Resource: GET PUT POST DELETE	/orders: GET - list all orders PUT - unused POST - add a new order DELETE - unused
	/orders/{id}: GET - get order details PUT - update order POST - add item DELETE - cancel order
	/customers: GET - list all customers PUT - unused POST - add a new customers DELETE - unused
	/customers/{id}: GET - get customer details PUT - update customer POST - unused DELETE - delete customer
	/customers/{id}/orders: GET - get all orders for customer PUT - unused POST - add order DELETE - cancel all customer orders

Editing Connection Profile Properties

Edit data source/enterprise resource configuration parameters for a specific connection profile.

Note: You can also view and edit connection profile properties in the Console Editor Application, which is an application you can launch outside the Eclipse workbench.

1. Open Data Source Enterprise Explorer.
2. Expand the category for the connection profile you want to edit properties.
3. Right-click the data source/enterprise resource and select **Properties**.
4. Select a property option to display its associated Properties page.
5. Edit the connection profile properties as necessary.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Connecting to a Connection Profile* on page 49

Renaming a Connection Profile

You can rename a connection profile.

Prerequisites

We recommend disconnecting from the server (connection profile) before renaming it.

Task

1. Open Data Source Enterprise Explorer.
2. Expand the category for the connection profile you want to rename.
3. Right-click the connection profile and select **Rename**.
4. Edit the name of the connection profile and click **OK**.
5. Click **OK** to save any project changes.

You cannot rename the "My Unwired server" or "My Sample Database" connection profiles. If you do, they are automatically recreated when you restart Unwired WorkSpace.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

Duplicating a Connection Profile

You can duplicate a connection profile.

1. Open the Enterprise Explorer and locate the connection profile you want to duplicate.
2. Right-click the connection profile and select **Copy**.
3. Right-click the connection profile folder under which you want the copied connection profile to appear and select **Paste**.

Configure

4. Enter a new name for the connection profile in **Profile Name Input**.

A copy of the connection profile appears under the server category in the Enterprise Explorer.

5. You can also use the following options when right-clicking a connection profile to facilitate creating new connection profiles from existing ones:
 - Undo Copy – removes the previously copied connection profile.
 - Redo Copy – enabled when Undo Copy executes, the removed copied connection profile is added back.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

Testing a Connection Profile

Ping the data source/enterprise resource to test a connection profile.

1. Open Data Source Enterprise Explorer.
2. Expand the category for the connection profile you want to test
3. Right-click the connection profile and select **Ping**.
4. Choose from the following:

Table 14. Ping options

Option	Action
If the ping fails	Verify that the data source/enterprise resource is running and check the connection profile properties.
To view error information	Click Details in the Error dialog.
If the ping succeeds	Click OK .

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43

- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

Connecting to a Connection Profile

Connect to the data source/enterprise resource defined by the connection profile.

1. In Enterprise Explorer, expand the folder for the type of connection profile to which you want to connect: Databases, Unwired Server, Web Services, and so on.
2. Right-click the connection profile and select **Connect**.

Once connected, you can expand the connection profile to access the data source.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39
- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46

Deleting a Connection Profile

You can delete a connection profile.

Prerequisites

Disconnect from the database server.

Task

1. Open Data Source Enterprise Explorer.
2. Expand the category for the connection profile you want to delete.
3. Right-click the connection profile and select **Delete**.
4. Click **Yes** to confirm deletion.

Note: You cannot delete a connection profile if the profile name begins with leading whitespace, for example " myConnectionProfile". You must rename the profile and remove the leading whitespace before deleting it.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 39

Configure

- *Creating a Web Service Connection Profile* on page 43
- *Creating a REST Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Connecting to a Connection Profile* on page 49

Creating a Sybase Unwired Server Connection Profile

Create a new Sybase Unwired Server connection profile to supply runtime connection information.

1. In Enterprise Explorer, right-click **Unwired Servers** and select **New**.
2. Enter a name and optional description of the connection profile and click **Next**.
3. Complete the following information on the wizard pages:

Table 15. Specify Connection Details page

Field	Description
Host	The host name on which the server resides. The default is the machine name.
IP domain	(Optional) The domain name. The default is the Windows domain of the local machine.
Port	The port to which you are connecting. The default port is 2000, use 2001 for HTTPS.
User Name	Enter the name for the server login. The default user name is supAdmin.
Password	Enter the password for the server login, if required. For example, s3pAdmin.
Save Password	Select the checkbox to save the password.

Field	Description
Secure	<p>Select the checkbox to enable a secure connection with the server, which is different than a secure connection session.</p> <p>To establish a secure session with Unwired Server:</p> <ol style="list-style-type: none"> 1. Enable Secure management port on Unwired Server's administration listener. See the Sybase Control Center help topic for information. 2. Enter 2001 as the Port. 3. Add the certificate file path to the <Unwired_Platform_Install_Dir>\Eclipse\UnwiredWorkspace.bat file. <hr/> <p>Note: Unless the Unwired Server administration listener certificate is signed by a trusted certificate authority, or the certificate is imported into the %JAVA_HOME%\jre\lib\cacerts trust store, you see an error message when testing the Unwired Server connection: "unable to find valid certification path to requested target".</p>
Cluster information	<p>The read-only properties for this Server's cluster (or non-cluster) Unwired Server connection information.</p> <p>Connection information includes server host, protocol, and ports used for replication-based and messaging-based synchronization. Depending on whether the cluster uses a load balancer, you may alternatively set the name of the load balancer host.</p>

4. Click **Next** to review a summary of this connection profile, or **Finish** to create the connection profile. The connection profile displays under Unwired Servers in the Enterprise Explorer.

See also

- *Creating a Data Source Connection Profile* on page 33
- *Preferences* on page 52
- *Importing and Exporting Connection Profiles and Projects* on page 60
- *Importing the Public Certificate* on page 63

Preferences

Use preferences to configure the appearance of perspectives and views and to customize its tools.

Some preference settings apply across all development components, while others apply only to a specific component. To enable the sharing of editor preferences across components, see the Eclipse *WorkBench User Guide* on the online bookshelf.

See also

- *Creating a Data Source Connection Profile* on page 33
- *Creating a Sybase Unwired Server Connection Profile* on page 50
- *Importing and Exporting Connection Profiles and Projects* on page 60
- *Importing the Public Certificate* on page 63

Setting Help Display Preferences

Define how you want to display help topics from an Eclipse-based product to appear.

1. Select **Window | Preferences** from the main menu bar.
2. In the left pane, select **Help**.

The Help options appear in the right pane.

3. Specify how to display help topics.
4. Click **OK**.

Setting Mobile Development Preferences

Use the Preferences dialog to set Mobile Application Diagram and mobile business object preferences.

1. Either:
 - Open the Preferences dialog from the menu, by selecting **Window > Preferences**. This is the recommended method, since it shows all preference options. Or
 - Open the Preferences dialog from the Mobile Application Diagram, by right-clicking in the Mobile Application Diagram and selecting **Preferences**.
2. In the left pane of the Preferences dialog, expand **Sybase, Inc > Mobile Development**.
3. In the Preferences dialog, configure preferences for:
 - Developer Profile
 - Logging
 - Miscellaneous
 - Mobile Application Diagram
 - Mobile Business Object
 - Mobile Workflow Forms Editor

4. Change preferences for the desired category, then click **Apply** to apply changes, or click **Restore Default** to use the default preference settings.
5. Click **OK**.

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Development Logging Preferences* on page 54
- *Mobile Business Object Preferences* on page 60
- *Mobile Application Diagram Preferences* on page 58
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Developer Profile Preferences

Set Basic and Advanced developer profile preferences for Unwired WorkSpace.

Select features so they are available from one or both profiles from the Developer Profile Preferences or Details page. Features that are grayed out cannot be modified. Select **Apply** for changes to take effect, or **Restore Defaults** to restore default profile settings.

Selecting features to include from the Developer Profile preferences page modifies all wizards, properties, and Workspace Navigator folders related to that feature.

Table 16. Developer profile properties

Property	Description
Current profile	Select which profile to use during the current session.
Switching dialog	When you switch profiles from the Mobile application Diagram, by default a confirmation prompt displays. Select this option to disable confirmation when switching profiles.
Features	Features and default profile settings include: <ul style="list-style-type: none"> • Cache – Advanced • Code generation – Both (cannot be modified) • Deployment – Both (cannot be modified) • Mobile business object – Both (cannot be modified) • Local business object – Advanced • Object query – Advanced • Synchronization group – Advanced

From the Details preferences page you can control which wizard pages, properties, and Workspace Navigator folders display for a given profile.

Table 17. Developer profile preferences details

Property	Description
Feature	Select a feature from the drop-down list to modify the various wizard pages, properties, and any WorkSpace Navigator folders associated with that feature, or select All and narrow your feature search from each of the Wizard, Property view, and WorkSpace Navigator tabs.
Wizard	Select the feature from the Wizard drop-down list to display all wizard pages associated with the feature. Choose which wizard pages to display for a given profile.
Property view	Select the feature from the Context drop-down list to display all property tabs associated with the feature. Choose which tabs display in the Properties view for a given profile.
WorkSpace Navigator	Select which WorkSpace Navigator folders display within a Mobile Application project for a given profile.

See also

- *Mobile Development Logging Preferences* on page 54
- *Mobile Business Object Preferences* on page 60
- *Mobile Application Diagram Preferences* on page 58
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Logging Preferences

Set mobile development logging preferences for logging events in the Unwired WorkSpace environment.

Note: To see the logging preferences you must access Preferences by selecting **Window > Preferences**, not from the Mobile Application Diagram.

Clear all root log receivers – clears all information for each log described in this section. Some plugins append messages to the root logger, which sends mobile development messages to their appender. The result can be the sending of duplicate messages to the console.

Table 18. Mobile development logging: Console

Property	Description
Enable Console logging	Enable logging to the Eclipse Console view (default). To see the Console view, select Window > Show View > Other > General > Console .
Log Level	Select the level of verbosity for console logging: <ul style="list-style-type: none"> • Debug — designates fine-grained informational events that are useful for debugging an application. • Info (default) — designates informational messages that highlight the progress of the application at a coarse-grained level. • Warn — designates potentially harmful situations. • Error — designates error events that might still allow the application to continue running. • Fatal — designates severe error events that will presumably lead the application to abort.

Table 19. Mobile development logging: Eclipse

Property	Description
Enable Eclipse Logging	Enable logging to the Eclipse log file and the Error Log view (default).
Log Level	Select the level of verbosity for console logging: <ul style="list-style-type: none"> • Debug — designates fine-grained informational events that are useful for debugging an application. • Info — designates informational messages that highlight the progress of the application at a coarse-grained level. • Warn — designates potentially harmful situations. • Error — designates error events that might still allow the application to continue running. • Fatal — designates severe error events that will presumably lead the application to abort.

Table 20. Mobile development logging: File

Property	Description
Enable file logging	Write logs to the specified file in the file system (default).

Configure

Property	Description
Log level	Select the level of verbosity for console logging: <ul style="list-style-type: none">• Debug — designates fine-grained informational events that are useful for debugging an application.• Info — designates informational messages that highlight the progress of the application at a coarse-grained level.• Warn — designates potentially harmful situations.• Error — designates error events that might still allow the application to continue running.• Fatal — designates severe error events that will presumably lead the application to abort.
Log file	Enter or browse to the location and name of the log file. Or use the default log file, uepdev.log.
Rollover frequency	Select how often the log file should be reset: Daily, Weekly (default), or Monthly.
Clear log file on startup	Erases contents of log file each time Unwired WorkSpace is started (default).

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Business Object Preferences* on page 60
- *Mobile Application Diagram Preferences* on page 58
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Miscellaneous Preferences

Set miscellaneous mobile development preferences for the Unwired WorkSpace environment.

Table 21. Miscellaneous preferences

Property	Description
Version	Shows version number on mobile application project

Property	Description
Preview	<p>Determines how a mobile business object component is previewed:</p> <ul style="list-style-type: none"> • Show warning when executing – displays warning messages to the console. • Show row number – adds a column in the preview output that lists row numbers. • Maximum rows to display – limits the number of rows to preview. The default is 100 rows. If this number is too high, the preview may take a long time to complete. • Display null value as – enter a value to display for null value. <Null> is the default. • Maximum length displayed in column – truncates the output to the maximum length indicated. The default is 30 characters per column.
Result Set Filters and Result Checker	Configures whether or not to prompt to add Java Nature each time you create a result set filter or result checker.
Actions	Determines whether or not you want to confirm any refresh or remap actions, or delete actions that cascade to subsequent MBOs.
Code generation	<ul style="list-style-type: none"> • Do not show code generation completion dialog again – select if you do not want to see the code generation completion dialog. • JavaDoc generation heap size (MB) – used only when you choose Generate JavaDoc in the configure options in the Generate Code wizard. <p>Note: The default heap size is set to 128 MB, but if you get errors, set the heap size larger than the default 128 MB.</p>
Object query	<ul style="list-style-type: none"> • Do not show prompt dialog for object query auto-generation – do not show a prompt dialog for an automatically generated object query when an attribute is set to the primary key. • Do not show prompt dialog when deleting primary key object query – do not show a prompt dialog when deleting an object query for a primary key. • Do not show prompt dialog when automatically changing return type and index setting for object query – do not show a prompt dialog when the query definition contains a join operation and automatically change the return type and index setting for the object query.

Configure

Property	Description
Data length	Sets the default for various attribute and parameter datatype values: <ul style="list-style-type: none">• Default string length – default preference for string length is 20.• Default binary length – default preference for a binary datatype length is 10.
Migrating	Do not show prompt dialog for migrate dialog again
Cache policy	Specifies a default cache refresh policy for any cache groups you create.
Relationship	<ul style="list-style-type: none">• Set the mapped attributes or parameter's propagate-to attributes as primary key and auto-generate the object query – set to Prompt (default), Yes (perform without prompt), or No (do not perform). Show label for relationship in diagram editor – select this option to label the relationship when displayed in the Mobile Application Diagram.

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Development Logging Preferences* on page 54
- *Mobile Business Object Preferences* on page 60
- *Mobile Application Diagram Preferences* on page 58

Mobile Application Diagram Preferences

You can set various preferences for the Mobile Application Diagram by selecting an option in the left pane and making your selections.

Global settings

Use this section of the Preferences page to enable (or disable) global settings used by the Mobile Application Diagram, including:

- Show connector handles
- Show popup bars
- Enable animated layout
- Enable animated zoom
- Enable anti-aliasing
- Show status line

Appearance

Use this section of the Preferences page to edit colors, fonts, and size of objects in the Mobile Application Diagram.

Section	Description
Colors and fonts	Modify the colors and fonts for various Mobile Application Diagram objects by selecting Change , and setting the desired color and font.
Size	Select Auto size to use the default size setting for the Mobile Application Diagram, or unselect this option to define a different size.

Connections

Set the line style used to define connections within the Mobile Application Diagram: either **oblique** or **rectilinear**.

Pathmaps

Use the **New**, **Edit**, and **Remove** buttons to modify path variables used for modeling artifacts. Pathmaps is a subset of the path variables used in the Linked Resources preferences page.

Printing

Define print settings for the Mobile Application Diagram.

Section	Description
Orientation	Either Landscape or Portrait.
Units	Either inches or millimeters.
Size	Select a size from the drop down list. A default value appears for the width and height, that you can accept or override by entering a different value.
Margins	Enter a value in inches for the margins.

Rulers and grid

Edit rulers and grid settings.

Section	Description
Rulers options	To show ruling lines, select Show rulers for new diagram , then select ruler units from the drop-down list.
Grid options	To modify grid settings, select Show grid for new diagrams or Snap to grid for new diagrams , then enter the grid spacing in the text box.

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Development Logging Preferences* on page 54
- *Mobile Business Object Preferences* on page 60
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Business Object Preferences

You can specify various preferences for creating mobile business objects, including defining prefix values for mobile business objects (including operation, attribute, and parameter prefixes).

Mobile business object

Edit mobile business object preferences.

Section	Description
Drag and drop	Specify the wizards that are launched by dragging and dropping the data source onto the Mobile Application Diagram: <ul style="list-style-type: none"> • Choose the part to create when dragging a data source directly on the diagram: <ul style="list-style-type: none"> • Prompt – create attributes and operations. • Attributes – creates attributes only. • Operation – creates operations only. • Do not show Quick Create dialog again. • Do not show warning dialog for external database again. • Do not show Role Assignments dialog again.
Naming prefix	Assign a default prefix for newly created mobile business objects, attributes, operations, or parameters.

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Development Logging Preferences* on page 54
- *Mobile Application Diagram Preferences* on page 58
- *Mobile Development Miscellaneous Preferences* on page 56

Importing and Exporting Connection Profiles and Projects

You can export mobile application projects and connection profiles from one Unwired WorkSpace and import them into another.

See also

- *Creating a Data Source Connection Profile* on page 33

- *Creating a Sybase Unwired Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing the Public Certificate* on page 63

Exporting Connection Profiles

Export connection profiles to an external file.

Exported connection profiles retain their connection information, allowing you to use them later (provided connection information remains the same) by importing them into other Unwired WorkSpace installations or when migrating to a more current version of Unwired WorkSpace.

1. From Enterprise Explorer, select the **Export** icon (below) to launch the Export Connection Profiles wizard .



2. Select the connection profiles to include in the export, or click **Select all** to export all connection profiles.
3. Specify a file name, or **Browse** to the location of an existing file.
A single file can contain multiple connection profiles. By default, files are encrypted.
4. Click **OK** to export the selected connection profiles to the specified file.

Exporting Mobile Application Projects

Export mobile application projects to an external directory.

Exported mobile application projects retain all of their reference information (data sources, roles, generated code, and so on), allowing you to use them later by importing them into other Unwired WorkSpace installations or when migrating to a more current version of Unwired WorkSpace.

1. From WorkSpace Navigator, right-click the mobile application project you are exporting and select **Export**.
To select multiple projects, either:
 - Ctrl+click to select individual projects, or
 - Shift+click a project to include all projects between the two selections.
2. From the Export wizard, select **General > File System** and click **Next**.
3. Complete the following:
 - a) Select all of the projects you want to export. By default, all project resources are also exported.
 - b) Enter, or **Browse** to the target directory of the exported mobile application projects. A subdirectory with the name of the mobile application project is created for each selected project.

Configure

- c) Click **Finish** to export the projects to the selected directory.

Importing Connection Profiles

Import connection profiles that were exported to an external file.

Prerequisites

Export the connection profile.

Task

Exported connection profiles retain their connection information, allowing you to use them later (provided connection information remains the same) by importing them into other Unwired WorkSpace installations or when migrating to a more current version of Unwired WorkSpace.

1. From Enterprise Explorer, select the **Import** icon (below) to launch the Import Connection Profiles wizard.



2. Specify a file name, or **Browse** to the location of exported file that contains the connection profile you are importing.
3. (Optional) Select **Overwrite existing connection profiles with same names**.
4. Click **OK** to import the selected connection profiles from the specified file.

When the import process completes, the connection profiles are automatically refreshed.

Importing Mobile Application Projects

Import mobile application projects that have been exported to an external directory.

Prerequisites

You must export the mobile application project.

Task

Exported mobile application projects retain all of their reference information, allowing you to use them later by importing them into other Unwired WorkSpace installations or when migrating to a more current version of Unwired WorkSpace.

1. Select **File > Import**.
2. Select **General > Existing Projects into Workspace** and click **Next**.
3. Browse to and select the root directory that contains the mobile application projects and click **OK**.

If there are multiple projects in a folder, select the parent folder; it is scanned for mobile application projects.

4. In the **Projects** section of the Import wizard, select the projects to import.

If the root folder contains projects that already exist in Unwired WorkSpace, they are unavailable for import.

5. Select **Copy projects into workspace** and click **Finish**.
6. If projects are in various root directories, repeat this process until all projects are imported.

Importing the Public Certificate

Use the keytool command to import the public certificate into your testing or Unwired WorkSpace environment, so that you can establish HTTPS connections with Unwired Server.

Prerequisites

You must first configure Unwired Server to accept HTTPS connections. You can then import the public certificate generated during that process and use it to secure HTTPS communications with Unwired Server.

Task

Use the Java **keytool** command to import the public certificate into the JRE on the host from which you want to connect to Unwired Server using HTTPS. The host is your Unwired WorkSpace installation, or the host on which you develop .NET client applications; for example, C:\Sybase\UnwiredPlatform\JDK<version>\jre). This task prepares your system to run J2SE Unwired Platform device applications.

1. From the JRE\bin directory enter the command: **%JAVA_HOME%\bin\keytool -import -keystore "%JAVA_HOME%\jre\lib\security\cacerts" -file (path to the certificate)**

The -file argument is the path to the public certificate.

If the import is successful, replace the default keystore password, which can be whatever you want. For example:

```
Enter keystore password:mykey
Re-enter new password:mykey
```

2. After entering the password you see output, similar to this, that identifies the certificate:

```
Owner: CN=UEP, OU=ITS, O=Sybase, L=Concord, ST=NH, C=US
Issuer: CN=UEP, OU=ITS, O=Sybase, L=Concord, ST=NH, C=US
Serial number: 31
Valid from: Sun May 11 16:04:03 EDT 2008 until: Wed May 12 16:04:03
EDT 2010
Certificate fingerprints:
MD5: 50:E1:8E:53:FE:3C:C9:E6:34:70:71:01:8E:09:C8:CE
SHA1: 20:B5:26:B0:9B:8B:F7:9E:16:BA:2E:13:3D:03:73:32:AA:6A:
52:53
Signature algorithm name: MD5withRSA
Version: 3
```

Configure

```
Extensions:  
#1: ObjectId: 2.5.29.15 Criticality=true  
KeyUsage [  
Key_Encipherment  
Key_Agreement  
Key_CertSign  
Crl_Sign  
]  
#2: ObjectId: 2.5.29.19 Criticality=true  
BasicConstraints:[  
CA:true  
PathLen:10  
]
```

The owner and Issuer information should match the information you entered when you generating the certificate and key.

3. Answer **Y** when asked whether to trust this certificate or not:

```
Trust this certificate? [no]: y  
Certificate was added to keystore
```

If you accept the default [no], the certificate is not added to the keystore.

The certificate should now be available from the keystore. If you enter an incorrect or invalid keystore path, or if the certificate import fails for other reasons, you receive a connection error when a J2SE client running on Windows attempts to connect (assuming this client points to the same %JAVA_HOME% as your import command). For example:

```
java.lang.RuntimeException:  
Synchronization of MetaData failed:  
ianywhere.ultralitej.implementation.JrException:  
UltraLiteJ Error[-44]: Sync upload failure:  
'sun.security.validator.ValidatorException:  
PKIX path building failed:  
sun.security.provider.certpath.SunCertPathBuilderException:  
unable to find valid certification path to requested target'
```

See also

- *Creating a Data Source Connection Profile* on page 33
- *Creating a Sybase Unwired Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing and Exporting Connection Profiles and Projects* on page 60
- *Certificate Generation Command Line Utility Reference* on page 65

Certificate Generation Command Line Utility Reference

Use the Certificate generation utility (gencert) to create a certificate or to sign pre-generated certificate requests.

Syntax

Option	Description
-c	Specifies a certificate you can use to sign other certificates. If used with -r, generates an enterprise root certificate.
-s	Specifies a server identity certificate. The server identity is a combination of a server's private key and public certificate. You reference the server identity certificate when you start Unwired Server (for transport-layer security) or database server (for SQL Anywhere client-server transport-layer security). If used with -r, generates a self-signed server certificate.
-r	Specifies a self-signed root certificate. If used with -s, gencert creates a self-signed server certificate. If used with -c, gencert creates an enterprise root certificate you can use to sign other certificates. If you specify gencert -r with no additional options, gencert creates a certificate you can use as a server certificate or an enterprise root. This option is not compatible with -q.
-q request-file	Sign a pre-generated certificate request. If used with -s, gencert creates a server certificate. If used with -c, gencert creates an enterprise root certificate you can use to sign other certificates. If you specify gencert -q with no additional options, gencert creates a certificate you can use as a server certificate or an enterprise root. The -q option is not compatible with -r.

If you do not specify -s or -c , the certificate contains the functionality provided by both options, so it can be used to sign other certificates or you can use it directly as a server certificate.

Description

You can use the gencert utility to generate trusted public certificates, private keys, and server certificates used to secure Unwired Server synchronizations or SQL Anywhere® client-server communication. This utility creates X.509 certificates (a standard certificate format) for various security configurations.

Gencert prompts you for the following information:

Configure

Field	Description
Cipher	Gencert prompts you to choose an ECC or RSA cipher. If you are generating an ECC certificate, gencert generates an ECC key pair. If you are generating an RSA certificate, it prompts for a key size between 512 and 2048, and then creates a certificate using RSA. (In general, longer keys provide stronger encryption but take longer to process.)
Country, State/Province, and Locality	These values provide general certificate identification. The locality fields are also required by third-party Certificate Authorities if you plan to use globally-signed certificates.
Organization, Organizational Unit, and Common Name	These fields provide additional security that the client is authenticating the correct certificate. On the client side, they correspond to the certificate_company, certificate_unit, and certificate_name protocol options, respectively.
Serial number	You are prompted to choose a serial number for the certificate. The serial number must use alphanumeric characters.
Certificate valid for how many years	You are prompted for the period (in years) that the certificate remains valid. If the certificate expires, all certificates signed by this certificate will also be invalid. Following the specified period, you will need to regenerate the enterprise root, each server certificate, and the public certificates distributed to clients.
Enter password to protect private key	This is the password you will specify in the certificate_password protocol option.
Enter file path to save certificate	Choose a file name and location for the certificate.
Enter file path to save private key	Choose a file name and location for the private key.
Enter file path to save server identity	Choose a file name and location for the server certificate.

Using Unwired Server in a Development Environment

You may be using Unwired Server in a personal development environment or a shared development environment.

In a personal development environment, Unwired Server is installed on your machine with the Sybase Unwired WorkSpace Eclipse development environment, so you can test and refine packages you develop. Default Unwired Server settings should be sufficient for personal development and subsequent deployment. However, should you need to tune settings, you can refer to the Sybase Control Center online help for details.

In a shared development environment, Unwired Server is typically installed by a Unwired Platform administrator (supAdmin) on a separate machine shared by multiple developers. This requires a multiple-seat site license. The supAdmin, is tasked with configuring and maintaining the cluster to which the server has been installed. However, developers can be made domain administrators so they can deploy and test packages they have created. The supAdmin accomplishes this by:

- Creating an administration user for each developer.
- Assigning those administration users to a domain.

For information on how to perform domain administration tasks and which artifacts you can manage within a domain, see the "Systems Administration" chapter in *System Administration*.

Configure

Develop

Develop mobile applications in Eclipse Edition. A mobile application is an end-to-end application, which includes the mobile business object (MBO) definition (back-end data connection, attributes, operations, and relationships), the generated server-side code, and the device application. The device application is the client-side application code that can be created using native coding or the programming APIs.

Use the following topics to guide you through developing mobile applications.

Developing a Mobile Business Object

You can define attributes and operations of a mobile business object (MBO) without immediately binding them to a data source, define them from and bind them to a data source, or create an MBO that does not bind to a data source (local business object).

The attributes and operations that define an MBO must be bound to a data source at some point in the development process, unless it is a local business object. If you already have a connection to the data source through a connection profile, you can quickly generate attribute and operation bindings based on the data source. However, if you do not have access to the required data source, you define the MBO, but bind your operations and attributes to the data source at a later point. The difference between the two development approaches is when you create and bind the attributes and operations:

- Create an MBO and bind to a data source immediately – includes two methods:
 1. Drag and drop the data source onto the Mobile Application Diagram, which launches the appropriate wizards and automatically creates bindings based on the selected data source.
 2. Create an MBO and its operations and attributes using the Mobile Application Diagram and palette that launches a set of wizards and allows you to bind them directly to a data source.
- Create an MBO and defer data source binding – Create an MBO and its operations and attributes using the Mobile Application Diagram and palette that launches a set of wizards and allows you to bind the MBO to a data source at a later time. After you define the data source, you bind the MBO to it from the Properties view.
- Create a local business object – create a local business object by clicking the local business object icon in the palette then click the object diagram. Local business objects can only run on the client and cannot be synchronized. It can contain attributes and operations that run on the device. For example, the local business object could be combined with other MBOs, where the local business object runs an object query against results returned by other MBOs.

See also

- *Working with Mobile Business Objects* on page 118
- *Packaging and Deploying Mobile Business Objects* on page 217
- *Mobile Business Objects* on page 70

Mobile Business Objects

The cornerstone of the solution architecture is the concept of the mobile business object (MBO). Mobile business objects help form the business logic for native Object API applications and mobile workflows by defining the data you want to use from your backend system and expose through your mobile application or workflow.

MBO development involves defining object data models with back-end EIS connections, attributes, operations, and relationships that allow filtered data sets to be synchronized to mobile devices. MBOs are built by developers familiar with the data and transactional requirements of the mobile application, and how that connects to the existing EIS data sources.

A mobile business object (MBO) is derived from a data source (such as a database server, Web service, or SAP® server). MBOs are deployed to Unwired Server, and accessed from mobile device application clients. MBOs:

- Are created using the Unwired WorkSpace graphical tools. These tools simplify and abstract back-end system connections, and provide a uniform view of transactional objects
- Are reusable, allowing you to leverage business logic or processes across multiple device types.
- Future-proof your application; when new device types are added, the same MBO can be used.
- Provide a layer of abstraction from Unwired Server's interaction with heterogenous backends/devices, as shown in the following diagram.



MBOs are developed to include:

- Implementation-level details – metadata columns that include information about the data from a data source.
- Abstract-level details – attributes that correspond to instance-level properties of a programmable object in the mobile client, and map to data source output columns. Parameters correspond to synchronization parameters on the mobile client, and map to data source arguments. For example, output of a SQL SELECT query are mapped as attributes, and the arguments in the WHERE clause are mapped as synchronization parameters, so that the client can pass input to the query.
MBO operations include parameters that map to data source input arguments. Operation parameters determine information a client passes to the enterprise information system (EIS).
- Relationships – defined between MBOs by linking attributes and parameters in one MBO, to attributes and parameters in another MBO.

Developers define MBOs using either a top-down approach—first designing attributes and parameters, then binding them to a data source; or a bottom-up approach—first specifying a data source, then automatically generating attributes and parameters from it.

A mobile application package includes MBOs, roles, and data source connection mappings, and other artifacts that are delivered to the Unwired Server during package deployment.

When the data model is complete, code artifacts are generated. The MBO package, containing one or more MBOs is deployed to Unwired Server. Other MBO artifacts are used to develop a mobile application using Native Object API or HTML5/JS Hybrid App API — when the application is deployed to a device, the MBO data set resides on the device. On device data changes are synchronized to the MBO on the server and then to the EIS back end. Back end changes are communicated to the device via the MBO on the server that sends a notification to the device and updates the MBO data on the device.

See also

- *Creating Mobile Business Objects* on page 78
- *Binding Mobile Business Objects to Data Sources* on page 85
- *Developing a Mobile Business Object* on page 69
- *Working with Mobile Business Objects* on page 118
- *Packaging and Deploying Mobile Business Objects* on page 217

Mobile Application Diagram

Each Mobile Application project has an associated Mobile Application Diagram that provides a graphical representation of all mobile business objects (MBOs) within the project.

Launch the wizards that define MBOs (including operations, relationships, and so on) by selecting the appropriate item from the palette and dropping it on the diagram. The Attributes Creation wizard launches automatically when you create an MBO.

Mobile Application Diagram Palette

Create mobile business objects and attributes, add operations to the object, define relationships between two objects, and so on, by selecting the appropriate palette icon and dropping it on the Mobile Application Diagram.

Table 22. Palette icons and usage

Icon	Description
Select	Selects a mobile business object or relationship.
Zoom	Zooms in or out on a selected element.
Note	Creates a note or note attachment in the Mobile Application Diagram for an existing note.
Mobile Application Diagram	Expand this folder to access the mobile business object related icons. The folder is expanded by default.
Mobile Business Object	Creates a new mobile business object and launches the mobile business object Attributes Creation wizard.
Local Business Object	Creates a new local business object and launches the local business object Attributes Creation wizard.
Relationship	Creates a relationship between two mobile business objects.
Attribute	Creates an attribute for an existing mobile business object.
Operation	Adds an operation to a mobile business object, and allows you to bind the operation to a data source now or later.

Creating a Mobile Application Project

A mobile application project is the container for the mobile business objects that forms the business logic of mobile applications.

You must create a mobile application project before you can create mobile business objects.

1. Select **File > New > Mobile Application Project** from the main menu bar.
2. Enter a:
 - Name
 - Location (if other than the default).
3. Click **Finish**.

An empty Mobile Application Diagram opens.

Opening a Mobile Application Diagram

Use the mobile application diagram to display and manage mobile business objects in a graphical view.

Prerequisites

You must create a mobile application project before you can open the Mobile Application Diagram.

Task

1. Locate the mobile application project you want to open in the WorkSpace Navigator.
2. Right-click the project and select **Open in Diagram Editor**.

Projects

A project is a collection of resources that together accomplish a task.

A project is the first resource that is created because it is the container for all other resources.

For example, in Database development, a project typically stores related SQL files that can be used later for queries or in creating procedural objects.

In Mobile Application development, a project stores mobile business objects and its components.

Copying a Project

You can import a copy of an existing project into your workspace. When a project is copied, its resources are not actually copied, but are a reference to the original resource.

If you make a change to a resource in the project, for example, a service, it is changed both in the original and in the copy.

If you want a copy of a project as an archive, have the project owner export the project to a zip file outside the workspace. Then changes can be made to the working version of the project or services within the workspace. To revert to the archived copy of the project or service, delete the working copy from the workspace, including contents within the project, then unzip the project zip file into the workspace and use the Existing Project Into Workspace option of the Import wizard to import the archived copy of the project back into the workspace.

If you want only a copy of the service in a project, you must copy the entire project. If you only copy the individual service, you are copying the selected file, but not the associated hidden files required to run the service.

Note: If you import a project directly from the file system by pointing to an existing project in the WorkSpace Navigator, you will not have a duplicate copy of the project. Any changes made to this project are made to the original copy.

1. Choose one of the following:

- Copy the exported project folder into the Eclipse/ *workspace folder* of your installation directory.
- Unzip the exported file into the workspace folder in your installation directory.

Note: If an existing project has the same name as the project you are copying, the existing project is overwritten. Change the name of one of the projects before the project is copied.

2. In the Mobile Development perspective, select **File > Import** from the main menu bar.

The Import wizard opens.

3. Select **Existing Project into Workspace** and click **Next**.

4. Follow the instructions to import the project.

The imported project appears in the WorkSpace Navigator and is available for use.

Sharing a Project

You can share a project that resides in a different location. Sharing a project differs from copying a project. When you share a project, the project is not exported from its original location.

Use the Import wizard to point to the shared project as your source directory.

1. In the Mobile Development perspective, select **File > Import** from the main menu bar.

The Import wizard opens.

2. From the General node, select **Existing Project into Workspace**.

3. Click **Next**.

4. Follow the instructions in the wizard to browse for an existing project to share.

5. Click **Finish**.

The project is available for use in the WorkSpace Navigator. However, all project *resources* remain in their original location.

Exporting a Project

You can export a project that can then be imported and shared with other developers.

To share projects with other developers, you can export your project if you are the project owner. You can export the project to a zip file or to the local file system and then notify developers to import the shared project into their project workspace.

Note: You cannot export project resources, such as services. You must export the entire project. However, you can copy, paste, or move services and folders between projects.

1. To refresh the project to ensure that all project files are up to date, select **File > Refresh** from the main menu bar.

- Right-click the project you want to export in the Workspace Navigator and select **File > Export** from the main menu bar.

The Export wizard opens.

- From the General node, select one of the following:

Table 23. Export options

Option	Description
Archive file	Select this method for projects that use double-byte characters in the service name. <hr/> Note: If you do not select this option, the exported project will have corrupted service names.
File system	Select this option to export all other project types.

- Click **Next**.
- To specify the project you want to export, select one of the following:

Table 24. Project types

Project type	Description
File system	Do the following: <ol style="list-style-type: none"> Select the project resources that you want to export. In the To directory field, browse for the directory that you want to export the project into. Select any of the following options: <ul style="list-style-type: none"> Overwrite existing files without warning. Create directory structure for files. Create only selected directories.
Archive file	Do the following: <ol style="list-style-type: none"> Select the project resources that you want to export. In the To archive field, browse for the directory that you want to export the project into. Select any of the following options: <ul style="list-style-type: none"> Save in zip format. Save in tar format. Compress the contents of the file. Create directory structure for files. Create only selected directories.

- Click **Finish** to export the specified project to the destination location.

Importing a Project

You can import a copy of project or projects into your *workspace* as an archive file or directly from a local file system.

Warning! If you import a project directly from the file system by pointing to an existing project in the WorkSpace Navigator, you are not importing a duplicate copy of the project. Changes made to this project are simultaneously made to the original project as well.

1. Set up your exported project for import and do one of the following:
 - Copy the exported project folder into the Eclipse *workspace folder* in your installation directory.
 - Unzip the exported project file into the workspace folder in your installation directory.

Note: If an existing project has the same name as the project you are importing, the existing project is overwritten. Change the name of one of the projects when you do the copy.

2. Select **File > Import** from the main menu bar.

The Import wizard opens.

3. Select **Existing Project into Workspace** and click **Next**.
4. Select the project you want to import selecting one of the following:

Table 25. Import a project

Field	Description
Select root directory	<ol style="list-style-type: none"> 1. Browse for the directory that contains the project you want to import. 2. Select the projects you want to import from the Projects list.
Select archive file	Browse for the archive that contains the project that you want to import.

5. Click **Finish** .

The project is now available for use.

6. After importing the project, right-click the project and select **Update WorkSpace Build Path Entries**.

Running this option ensures that the paths for build and deploy are updated for the workspace to which the project was imported.

Switching Between Developer Profiles

Switch between basic and advanced developer profiles in the Mobile Application Diagram.

Unwired WorkSpace provides two developer profiles, basic and advanced. Basic is the default profile. If you need an Unwired WorkSpace feature (such as a wizard, property, or WorkSpace Navigator item) that is not in the basic profile, switch to the advanced developer profile. For example, to use backend data sources other than those supplied by Sybase Unwired Platform,

you must switch to the advanced developer profile to see the Server Connection Mapping page when deploying the Mobile Business Object package.

If you want to use the advanced profile by default, modify your developer profile preference settings.

- To switch between developer profiles, right-click in the Mobile Application Diagram, select **Switch Developer Profile**, then select either **Basic** or **Advanced**.
- To view or modify your preference settings for the developer profile, click **Window > Preferences > Sybase, Inc. > Mobile Development > Developer Profile**.

Mobile Development Basic and Advanced Developer Profiles

Set Basic and Advanced developer profiles in Unwired WorkSpace.

Unwired WorkSpace provides two developer profiles:

- **Basic** – is a subset of the features available to the Advanced developer, and allows you to develop and deploy mobile business objects (MBOs). Customize the Basic profile so that you see only required properties, wizards, screens, and so on.
- **Advanced** – includes all Unwired WorkSpace features, wizards, and properties, enabling you to perform additional MBO customization not available in the Basic profile.

Determine what profile to use based on the features required to develop your MBOs. If you cannot meet your needs with the Basic profile, use the features available in the Advanced profile, or customize a profile to provide access to specific features.

Naming Conventions

Follow naming convention guidelines to name resources.

Resource	Guideline
Project	Project names cannot contain a space.
Mobile business object	Mobile Business object names must start with an alphabetic character or an underscore. MBOs cannot contain C# or Java reserved words.
Mobile business object parameter	Parameter names: <ul style="list-style-type: none"> • Cannot contain C# or Java reserved words. • Have a maximum length of 64 characters. • Must start with an alphabetic character or an underscore.

See the *Unwired WorkSpace Validation Rules and Error Messages* for a complete list of guidelines.

Creating Mobile Business Objects

Create various types of mobile business objects (MBOs) and bind them to data sources to implement mobile application business logic.

See also

- *Mobile Business Objects* on page 70
- *Binding Mobile Business Objects to Data Sources* on page 85

Creating a Mobile Business Object by Dragging and Dropping a Data Source

Create a mobile business object (MBO), define the attributes and operations, and bind directly to the data source by dragging and dropping the data source onto the Mobile Application Diagram.

1. *Creating a Mobile Application Project*

A mobile application project is the container for the mobile business objects that forms the business logic of mobile applications.

2. *Drag and Drop the Data Source onto the Mobile Application Diagram*

Launch the Quick Create wizard that creates your mobile business object by dragging and dropping the data source onto the Mobile Application Diagram.

3. *Creating Relationships Between Mobile Business Objects*

Use the New Relationship wizard to create relationships between two or more mobile business objects.

4. *Modifying Mobile Business Object Properties*

Edit existing MBO operations, attributes, relationships, and so on, as well as Mobile Application Diagram properties from the Properties view.

5. *Previewing Mobile Business Objects*

Preview mobile business object operations and attributes against the data source to which they are bound Using the Preview and Test Execute dialogs.

Drag and Drop the Data Source onto the Mobile Application Diagram

Launch the Quick Create wizard that creates your mobile business object by dragging and dropping the data source onto the Mobile Application Diagram.

Prerequisites

You must have a Mobile Application project and a connection to the data source before creating the MBO.

Note: When dragging and dropping a non-Sybase data source, you must modify the SQL definition as indicated by the warning dialog to create non-Sybase MBOs or operations. Use either the Mobile Application Diagram (where you can manually enter the SQL definition) or use the Visual SQL dialog to create the SQL definition.

Task

1. Drag a data source entry (for example, a database table from a connection profile) and drop it onto the Mobile Application Diagram.

A MBO is created and data source information is automatically filled in with the appropriate operation-to-command mappings, attribute-to-tabular-view mappings, and parameter-to-argument mappings (except for database views and non-Sybase data sources).

Note:

- If you drag-and-drop a data source that includes a column with a computed default value, parameters for that column are not generated. You must manually modify the SQL definition to add the column from the Properties view after creating the MBO.
 - When dragging and dropping an ASE database that includes a computed column table, Unwired WorkSpace does not generate operations automatically. You must define them with the Operation Creation Wizard.
-

2. Follow the wizard instructions to complete the MBO.

Attribute and operation information varies, depending on the data source to which you bind.

3. Once defined, use the Properties view to modify if necessary.

Creating a Mobile Business Object and Deferring Data Source Binding

Launch the wizards used to create a mobile business object, attributes, and operations from the Mobile Application Diagram. Then bind the mobile business object to a data source from the Properties view when the data source is available.

1. *Creating the Mobile Business Object using the Mobile Business Object Icon*

Use the Mobile Business Object icon to create a mobile business object.

2. *Creating Attributes for a Mobile Business Object*

Use the Mobile Business Object Creation wizard to create a mobile business object and launch the Attributes Creation wizard. A mobile business object may or may not contain attributes.

3. *Creating Operations for a Mobile Business Object*

Use the Operation Creation wizard to create an operation and add it to the mobile business object.

4. *Binding Mobile Business Objects to Data Sources*

Create mappings and bind data source content to mobile business object operations and attributes either when you create them, or later using the Properties view.

Creating the Mobile Business Object using the Mobile Business Object Icon

Use the Mobile Business Object icon to create a mobile business object.

Prerequisites

Before you create a mobile business object, open the Mobile Development perspective and create a Mobile Application project.

Task

1. From the Mobile Application Diagram, select the Mobile Business Object icon from the palette, then click an empty area of the diagram.
The Attributes Creation wizard displays. By default, the mobile business object is named Objectx, where x is 1 if this is the first object in the diagram, 2 if the second, and so on.
2. (Optional) Change the default name of the mobile business object.
3. In the next page select **Bind data source later**.
4. In the last page add attributes and click **Finish**.

See also

- *Mobile Business Object Properties* on page 146
- *Mobile Business Object Attribute Properties* on page 151
- *Mobile Business Object Operation Properties* on page 153
- *Datatype Support* on page 125
- *Old Value Argument* on page 149

Creating Multiple Mobile Business Objects From a Single Operation

Create multiple mobile business objects (MBOs) from a single operation, which allows selection of multiple output objects and tables from a single call to the enterprise information system (EIS) to which the MBO is bound.

Prerequisites

The data source from which you are creating the MBO must contain either:

- Multiple SAP tables – EIS operations that contain multiple business application programming interface/remote function calls BAPI/RFC operations that contain multiple output tables. All selected output tables can be used to generate first/subsequent MBOs. Each SAP output table is regarded as an independent result set used to create a single MBO. Each MBO can be created from a single SAP output table.
- Multiple Web service XSLTs – EIS operations that contain multiple methods bundled with multiple XSLTs. All XSLTs can be used to generate first/subsequent MBOs. You can define more than one XSLT for a single Web service EIS operation. Each XSLT is regarded

as an independent result set used to create a Web service MBO. Each MBO can be created from a single XSLT.

Multiple MBOs that are created from the same data source and share the same EIS operation are treated as a whole unit, which affects certain behavior:

- Executing the EIS operation once updates all MBOs, improving performance compared to calling the operation for each MBO.
- Subsequent MBOs cannot exist without a first MBO. Copying, pasting, or deleting a first MBO performs the same action on subsequent MBOs.
- All subsequent MBOs are included in the search results of the first MBO.

Task

1. Launch the MBO Creation wizard. For example, drag and drop the data source onto the Mobile Application Diagram.
2. On the Definition page, select and define the first MBO.
3. (Optional) On the Parameters page, define the first MBO's default values.
4. On the Attributes Mapping page, select the Web service XSLT or SAP table for the attributes used to specify the Primary MBO.
5. The Create Multiple Mobile Business Objects page appears if:
 - You select multiple output tables on the Definition page for SAP, or
 - There are multiple XSLTs defined on the Definition page for Web services.

The Definition window varies, depending on the data source type, and allows you to determine which SAP output table or Web service XSLT to designate as subsequent MBOs.

Table 26. Multiple mobile business object data source

Data source	Description
SAP	<p>A BAPI/RFC operation that contains multiple output tables. All output tables, except the one defined as the first MBO, can be used to generate subsequent MBOs.</p> <p>In cases where you select an output parameter (either a primitive or structured output parameter), the <HEADER FIELDS> option is available in the Attribute Mapping page in both the creation wizard and the Attributes tab, available from the Properties view.</p> <p>When you select <HEADER FIELDS> as the Select an output table to map attributes option, Unwired WorkSpace generates a MBO that contains the result set from only the output parameters, not from any output tables. The <HEADER FIELDS> result set is a pseudo-table that behaves as a table result set:</p> <ul style="list-style-type: none"> • It is the first MBO by default. • If you select another table's output as the first MBO result set, the <HEADER FIELDS> result set generates a subsequent MBO.
Web service	<p>A method bundled with the multiple XSLTs. All XSLTs, except the one selected as the first MBO, can be used to generate subsequent MBOs.</p>

6. On the Create Multiple Mobile Business Objects page, select any of the subsequent MBOs, and optionally change the default names. By default, all MBOs listed on this page are selected.

Complete definition of the MBOs and select **Next** to configure role mappings (Advanced profile only) or **Finish** to exit the wizard.

7. (Optional) Map the first MBO and operations to logical roles. Subsequent MBOs inherit logical role assignments from the first.

8. Select **Finish** to create the MBOs and exit. You can perform some configuration from the creation wizard, however Sybase recommends that you perform any additional configuration using the Properties view.

Once created, multiple MBOs:

- Are stacked on each other and identified in the Mobile Application Diagram by a dashed line between the first MBO and all subsequent MBOs. You can independently delete subsequent MBOs.
- Have a "shared" relationship, identified by a different decorator (icon) in the WorkSpace Navigator, indicating the operation is shared. If the operation changes, all related MBOs automatically change.

First/Subsequent Mobile Business Object Properties

First/subsequent mobile business object (MBO) properties available from the Properties view.

First MBO refers to the first MBO defined from a single EIS operation from which multiple MBOs can be defined. Subsequent MBO refers to any other MBOs defined from this

operation. To ensure that the properties of the first and subsequent MBOs maintain synchronization, most subsequent MBO properties are read-only. The shared properties (definition, load parameters, and so on) of subsequent MBOs can be modified only through the first MBO.

Table 27. Multiple MBO attribute properties

Property tab	First/subsequent MBO
Share	<p>Identifies the enterprise information system (EIS) operation to which the MBOs are bound.</p> <p>The share tab appears for:</p> <ul style="list-style-type: none"> • Any MBO (except JDBC and REST) with multiple result sets. • First MBOs with at least one subsequent MBO. • Subsequent MBOs. <p>The first MBO Share tab includes Add, Delete, and Delete All buttons, which you can use to add or delete subsequent MBOS.</p> <p>If you delete all subsequent MBOs, the first MBO reverts to a "normal" MBO. If a subsequent MBO is added to a normal MBO, the normal MBO becomes the first MBO in a first/subsequent MBO relationship. In other words, Unwired WorkSpace supports switching MBOs between normal and first MBOs.</p>
Data source	<p>Available only from the first MBO.</p> <ul style="list-style-type: none"> • Change Connection Profile – retains the first/subsequent MBO relationship. • Bind Data Source – drops the first/subsequent MBO relationship and deletes all subsequent MBOs. The first MBO reverts to a normal MBO.
Definition	<p>The definition only can be changed in the first MBO, other than the ability to modify subsequent result set filters.</p> <p>After changing the definition, Unwired WorkSpace attempts to match and merge the changes, which may include adding, refreshing, or deleting subsequent MBOs.</p> <p>First and subsequent MBOs share the same credential properties.</p>

Property tab	First/subsequent MBO
Roles	Subsequent MBOs inherit roles from the first (as defined in the wizard), but you can also add or modify roles for subsequent MBOs.
Load parameters	Available from the first MBO and is read-only for subsequent MBOs. Add load parameters from the Definition tab of the first MBO. For each MBO, the load parameter can only propagate (propagate to attribute) to its own attributes.
Attributes mapping	Modify attribute mappings for subsequent MBOs independently of the first.
Object queries	Create object queries for subsequent MBOs independently of the first.
Cache group	First MBO and all its subsequent MBOs must be in the same cache group, but can be in different synchronization groups.

Creating a Local Business Object

Create a local business object, which is not bound to a data source.

A local business object can be deployed to Unwired Server and perform business functions locally on the device application, for example call object queries or persist application specific configuration information locally across application restarts. Local business objects cannot be filtered or synchronized with the Unwired Server cache database (CDB).

1. Launch the Local Business Object Creation wizard by selecting the Local Business Object icon from the Palette and dragging it onto the Mobile Application Diagram.

Note: You cannot create a local business object by dragging and dropping a data source onto the Mobile Application Diagram, since this method automatically binds the MBO to the data source.

2. On the Attributes Definition page, define the attribute values as you would any other MBO.
3. Select the operations which are allowed to be performed by the local business object.
4. Make any changes from the Properties view, if necessary.

Local business objects support object queries, and are listed (and can be included) in the Code Generation wizard, since local business objects can be referenced in device application code.

Binding Mobile Business Objects to Data Sources

Create mappings and bind data source content to mobile business object operations and attributes either when you create them, or later using the Properties view.

Prerequisites

You must create a mobile business object before you can bind data sources, unless you have created the mobile business object by dragging and dropping a data source onto the Mobile Application Diagram, in which case the binding is performed automatically. The data source to which you are binding must be available. For example, a connection to the database is available through a connection profile.

Task

The binding of data source information to mobile business object attributes and operations varies, depending on the data source to which you are binding. There are several places within Unwired WorkSpace from which you can bind to a data source. Not all options are available from all locations. The Properties view offers the broadest support when binding/rebinding or editing MBO data sources.

Note: Use the Bind Data Source button available from the Properties view to bind to a data source if you are binding after you have created attributes and/or parameters.

1. Bind a data source to mobile business object attributes and operations from:

Option	Description
New attribute or new operation wizard	Bind to a data source when adding operations and attributes to a new mobile business object.
Properties view	For an existing mobile business object that you have created in a top-down fashion (in which you create attributes and/or parameters without binding to a data source immediately), there are a number of places that you can create a data source binding depending on the context you are in (for example, selecting the Attributes tab), then selecting the Bind Data Source button.

2. Follow the wizard instructions to bind the particular data source information to mobile business object attributes and operations.

See also

- *Mobile Business Objects* on page 70
- *Creating Mobile Business Objects* on page 78
- *Creating Operations for a Mobile Business Object* on page 151

Supported Data Sources

This topic describes the various data sources to which you can bind the operations and attributes of a mobile business object. When you bind a mobile business object to a data source, the attribute and operation mappings vary depending on the type of data source to which the mobile business object is bound.

Table 28. Mobile business object data sources

Data source	Description
Database (various types)	A database object (table, view, stored procedure, and so on)
SAP	An SAP BAPI, or RFC operation
Web service and RESTful Web services	A local or remote WSDL file, or a SOAP service

Propagating a Client's Credentials to the Back-end Data Source

Use client credentials to establish enterprise information system (EIS) connections on the client's behalf for all data source types.

To use client credentials, map an EIS connection's username and password properties to system-defined "username" and "password" personalization keys respectively. This creates a new connection for each client and the connection is established for each request (no connection pooling.)

1. During development of the mobile business object MBO/operation, from the data source definition page (available either in the Creation wizard or from the Properties view), in the **Runtime Data Source Credential** section (or **HTTP Basic Authentication** section for a Web Service MBO), enter the client credentials in the User name and Password fields. The runtime data source credential values (user name and password) that Unwired WorkSpace uses for refresh or preview operations is taken in this order:
 - a) Any literal value entered in the User name and Password fields.
 - b) User-defined personalization keys that have non-empty default values.
 - c) User name and password property values contained in the connection profile.
2. During deployment of the package that contains such MBOs, map the design-time connection profiles to the existing or new server connections, but be aware that the username and password portions for the selected server connection is replaced by the username and password propagated from the device application.

Note:

- Do not set client credentials using the Runtime Data Source Credential option for MBO's that belong to a cache group that uses a Scheduled policy, since this is unsupported.

- In general, a MBO operation that uses data source credential settings as connection properties cannot have these settings mapped to an enterprise information system (EIS) during deployment. Instead, they maintain their original settings, which you can map after deployment using Sybase Control Center (SCC).
-

Implementing SSO for SAP

Configure SAP MBOs so they can be used in device applications that implement SSO.

To implement single sign-on for SAP in the development environment, you must bind your MBO to the SAP data source or bind your MBO to a SAP interface exposed as a Web service:

- MBO bound to SAP data source – set "Runtime Data Source Credentials and Connection Properties", by propagating the client's credentials to the back-end data source using the username and password personalization keys.
- MBO bound to SAP Web service – set "HTTP Basic Authentication", by propagating the client's credentials to the back-end data source using the username and password personalization keys.

For the production environment, see the topic *Implementing SSO for SAP* in *System Administration*.

See also

- *Creating an SAP Connection Profile* on page 39
- *Binding an SAP Data Source to a Mobile Business Object* on page 96
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 106

Binding a Database Data Source to a Mobile Business Object

Bind the attributes and operations of a database object to a mobile business object.

1. In the Bind Data Source wizard, enter the following information and click **Next**:

Field	Action
Specify data source	Select and bind to a data source now.
Data source type	Select Database from the drop-down list.
Connection profile	Select the database connection profile to which you are binding your mobile business object attributes. If the required data source is not in the list, click Create to define a new connection profile.

2. Follow the wizard instructions to map and bind data source information the mobile business object attributes and operations.

Field	Action
Data source specific information	<p>In the wizard, enter the SQL statement used to access the database information. Click:</p> <ul style="list-style-type: none"> • SQL Statement Type – either a SQL query statement or a stored procedure. • Validate – to validate the syntax of the SQL statement. • Visual SQL – to launch a Visual SQL dialog, from where you can create the SQL statement. • Preview – to view the results of the SQL statement against the data source.
Runtime Data Source Credential	<p>The User Name and Password required to gain access to the runtime data source.</p> <p>The user name and password can be entered directly, retrieved from a personalization key, or by selecting New key and creating a new personalization key.</p>
Result set Filter	<p>Optionally add a result set filter to the MBO.</p> <p>A result set filter is a custom java class that manipulates the rows or columns of data returned from a read operation for an MBO. To write a filter, developers must have previous experience with Java programming — particularly with the reference implementations for <code>javax.sql.RowSet</code>, which is used to implement the filter interface. See <i>Result Set Filters</i>.</p>
Parameters	<p>From the Parameters page you can configure parameters:</p> <ol style="list-style-type: none"> 1. Link (map) or unlink (unmap) from one parameter to the remote operation's argument. 2. Use Add to add a new parameter, then map it to one of the arguments using the approach described in step one, or select an argument from the Argument drop-down list. 3. Use Delete or Delete all to remove parameters. <hr/> <p>Note: (Properties view only) This is different from unmapping; after unmapping, the parameter still exists and displayed in the bottom table, but the <u>original argument moves down</u>.</p>

Field	Action
Attributes Mapping	<p>Attributes mapping is generated as follows:</p> <ol style="list-style-type: none"> 1. The tabular view columns are generated automatically. 2. If an existing attribute matches the name and data type of one of the columns, they are mapped. 3. Otherwise, you must manually link (map) from the attribute to the tabular view columns. <p>(Properties view only) The methods that you use to map attributes include:</p> <ol style="list-style-type: none"> 1. In the visual portion of the diagram, drag from the square icon of an attribute to the square icon of a tabular view column. 2. In the bottom table, select a column from the drop-down list of the map To cell. <p>From the Attributes Mapping screen, you can modify the mappings by selecting:</p> <ul style="list-style-type: none"> • Delete or Delete All – remove the selected attribute or all attributes. • Add – add a new attribute. • Up or Down – adjust attribute position. <p>The data type of a data source column is read-only, and changes only if the Map to column changes, except for JDBC, in which case the column type is not read-only, and you can change the data type according to the back-end data source.</p>

3. The **Role Assignments** screen allows you to **Create, Add, and Remove** role assignments from the mobile business object.
4. Click **Finish** when done.
The Mobile Application Diagram is refreshed with the new mobile business object attributes.

See also

- *Result Set Filters* on page 211

AutoCommit Option in JDBC Attributes or Operations

When you create MBOs from Adaptive Server® Enterprise stored procedures that use temporary tables, you must select the **AutoCommit** check box in the New Attributes or New Operation definition screen.

An error message displays in Unwired WorkSpace if Auto Commit is not selected. For example, create a temporary table named “tempstores” in this stored procedure:

```
CREATE PROCEDURE dbo.ase_sp
AS
BEGIN
    create table tempstores (temp_row_id integer, temp_id integer )
    insert tempstores select 1, 1
    insert tempstores select 2, 2
    insert tempstores select 3, 3
    select temp_row_id, temp_id from tempstores group by temp_row_id
```

```
drop table tempstores
END
```

If this stored procedure is used as a data source to model an MBO (attributes and operations), when you preview an operation, this error message displays:

```
The 'CREATE TABLE' command is not allowed within a multi-statement
transaction
in the 'tempdb' database.
```

Stored Procedures with Output Parameters and Result Sets

Define a mobile business object (MBO) from a stored procedure's Scalar and Cursor output parameters (and Cursor result sets).

Scalar parameters are common to all databases and Cursor parameters are exclusive to Oracle and DB2 databases.

Defining MBOs from stored procedures with output parameters (SPOP)

From a stored procedure, the developer can map attributes of a MBO to:

- Scalar output parameters
- Cursor as output parameters
- Cursor as result sets

Table 29. Stored procedures with output parameters definition examples

Stored procedure's output parameter	SQL query/definition
Scalar output parameter (ASA)	<pre>{CALL sampledb.dba.testScalarSPOP(:id,:amount)} create PROCEDURE dba.testScalarSPOP (in id INT, out amount INT) BEGIN select * from bonus; select bonus_amount into amount from bonus where emp_id = :id; END</pre>
Cursor as output parameter (Oracle)	<pre>{CALL TESTCURSORSPOP(["pcursor"=:pcursor])} create or replace procedure testCursor- SPOP(p_cursor out types.cursorType) as begin open p_cursor for select * from tblSPOP; end;</pre>

Stored procedure's output parameter	SQL query/definition
Cursor as return result set (DB2)	<pre>{CALL TESTCURSORSPOP()} CREATE PROCEDURE testCursorSPOP() LANGUAGE SQL DYNAMIC RESULT SETS 1 BEGIN DECLARE C1 CURSOR WITH RETURN TO CLIENT FOR SELECT * FROM tblSPOP; OPEN C1; END</pre>

Preview dialog

Only In and Inout parameters appear in the Preview dialog. Out parameters are filtered from the parameter table.

When the mouse hovers over an optional result set in the **Select a result set to preview** field, a tooltip describes the column structure of the result set.

All possible result sets are listed in the corresponding input field, from which you can select one to preview. Result sets can be derived either from an input SQL statement or output parameters of stored procedures, and are represented either as:

- DERIVED – identifies the result set that was derived from a list of stored procedure out parameters. Or,
- RESULT SET- n – identifies a result set returned from a stored procedure or SQL statement. For a stored procedure that returns multiple result sets the index (1, 2, and so on) identifies the result set in the list of result sets returned by the stored procedure.

All Scalar output parameters are grouped into a single result set (DERIVED in the **Select a result set to preview** field) while each Cursor output parameter derives a separate result set.

Attributes Mapping and Properties view

Similar to the Preview dialog, in that when the mouse hovers over the result set in the **Select a result set to preview** field, a tooltip describes the column structure. All possible result sets are listed in the corresponding input field (DERIVED and RESULT SET -n), from which you can select one for attribute mapping.

Whenever you change the result set, the mapping control and mapping table are automatically refreshed.

Creating Multi-level Insert Operations Using Autoincrement Primary Keys

When creating multi-level (chained) insert operations where the primary key of the parent MBO is set to autoincrement, use the "@@identity" parameter in the select statement to provide the chained insert value.

This method of creating a multilevel insert operation is useful if the primary key is set to autoincrement, you are making the relationship between two related Adaptive Server Enterprise/SQL Anywhere database mobile business objects, and you are creating them from the tool palette within the Mobile Application Diagram.

1. Create two MBOs (for example, Customer and Sales_order).

The Customer table has an "id" column which is a primary key of int type with autoincrement default values (or identify type). Sales_order is another table, which has a column named "cust_id" (a primary key of int type).

Since "id" is set to autoincrement, each new row added to the table is uniquely identified. ("id" becomes "@@identity" from the first SQL insert statement).

2. From the Mobile Application Diagram, define the relationship between the MBOs as a **Composite** and **One to many**, and link the Customer table's "id" attribute to Sales_order's "cust_id".

Be sure that the child MBO will be synchronized either independently, or through the parent MBO.

3. Use a **create** statement to insert into the Sales_order MBO and add a **select** statement that returns the "@@identity" row, which is the ID used for the chained insert statement into the Customer MBO.

Example: Chained insert SQL statement

```
INSERT INTO sampledb.dba.customer
(
  fname,
  lname,
  address,
  city,
  state,
  zip,
  phone,
  company_name)
VALUES
('["id"=":id"]',
'["fname"=:fname"]',
'["lname"=:lname"]',
'["address"=:address"]',
'["city"=:city"]',
'["state"=:state"]',
'["zip"=:zip"]',
'["phone"=:phone"]',
'["company_name"=:company_name"]')
```

```
)
SELECT * FROM sampledb.dba.customer WHERE id=@@IDENTITY
```

Note: "id" is a primary key column of identity(or autoincrement) type. Notice that the extra **select** statement and 'id' are not part of the insert statement itself.

Understanding Multi-level Insert Operations

In a multi-level insert, multiple mobile business objects are synchronized in a single operation. The mobile business objects must be bound to a JDBC datasource, have a defined relationship, and the insert parameters must support the relationship.

Some business processes require multiple related enterprise information system (EIS) operations; for example, creating a sales order with line items. The parent/child relationship is often represented by primary key(PK) / foreign key(FK) attributes in the parent and child mobile business objects (MBOs). When you construct these types of MBOs in an offline client application, the primary-key and foreign-key values are transitory. When EIS operations are called to create real data, the EIS systems generate the actual key values, and the primary key of the parent is copied to the related child MBO creation operations. These types of operations are known as "chained insert" or "multilevel insert."

For database MBOs using Sybase databases, dragging and dropping a table that contains autoincrement columns (one mechanism for generating primary keys) automatically creates the appropriate operations for obtaining the parent's generated keys and applying them to the children.

Typically, in a multi-level insert operation, you:

1. Create the parent MBO, and indicate the attributes that constitute that MBO's primary key.
2. Create the child MBO and define a relationship from the parent MBO's primary-key attributes to the child's foreign-key attributes.

Synchronization of the child MBO should occur either independently or through the parent MBO. See the *Client Object API* documentation for details.

3. Define the insert operations for the parent and child MBOs.

The insert operation for the parent MBO must return a single row that contains the primary-key values. The column labels must match the attribute names of the parent MBO. With this information, and the relationship-mapping data, Unwired Workspace modifies the input parameters for the insert operation of the child MBOs by replacing the foreign-key attributes with the ones returned from the parent MBO's insert operation. For example:

```
CREATE TABLE parent(pk int autoincrement primary key, p1
varchar(30), ...)
CREATE TABLE child(fk int references parent.pk, ...)
```

The parent insert MBO is defined as:

```
INSERT INTO parent(p1, ...) VALUES(?, ...); SELECT @@IDENTITY AS
id;
```

Develop

This batch query inserts the new parent row, and returns the newly generated primary key value.

You must understand the key-generation mechanism used by the EIS application from which you are developing, and be able to determine how to retrieve the newly generated keys during the insert operation (frequently, this logic is wrapped in a stored procedure).

This same technique applies to Web service, SAP, and other EIS systems, though the insert-operation definitions differ.

Note:

- The name of the from attribute of the insert operation parameter and parameter of the insert operation do not need to be the same name.
- The insert query returns only the identity column.

The single row that is returned must contain the column referenced in the relationship between the parent MBO and the child MBO, and the label of the column must match the from attribute name of the parent MBO.

Not all columns in the inserted row are required. For example, not all columns are selected or required for a drag-and-drop database operation.

- A multilevel insert records all logs under the parent MBO. All pending actions are also listed under the parent MBO.

Errors may occur if:

- The client sends the parent ID, which does not correspond to the server's interpretation of the parameters of the insert operation.
- The customer's primary key consists of more than one attribute.
If the child has multiple foreign-key attributes pointing to the parent, the relationship should list all relevant parent-to-child attributes. As long as the row returned from the parent insert contains all those columns, the child insert should work; all the foreign-key fields are populated from the parent insert result set.
- The insert operation of the parent fails at the back end.
- There is no association relationship between customer and order in which the source attribute/parameter in customer is a primary key and the target parameter in order is a foreign key to customer.
- The result set generated by the parent's insert operation does not have the required single row with the newly created primary key of that operation.

Note: Unwired Server does not report the specific reason of a multilevel insert failure. If you receive errors, or if the insert fails, check each of these items to try and identify the problem.

Creating Multi-level Insert Operations for Non-autoincrementing Primary Keys

Modify the **create** statement from the Properties view to support a multi-level (chained) insert operation where the primary key does not autoincrement.

If you drag and drop an Adaptive Server Enterprise (ASE) or SQL Anywhere database table to create the **insert/create** operation used in a chained insert operation, the Quick Create wizard creates the mobile business object with default generated operation statements:

- If the database table has a primary key column that is of autoincrement or identity type, then you do not need to modify the **insert** and **select** statements, if the MBO is the parent MBO used later for the chained insert.
- If the primary key is not autoincrement/identity type, modify the **insert** statement manually after the MBO is created to perform the chained insert operation.

1. Drag and drop the data source onto the Mobile Application Diagram. For example, drag and drop a table named customer2 onto the Mobile Application Diagram.
2. Click **OK** in the Quick Create wizard to generate the MBO with default values. The following **create** statement is automatically generated for the database table, and can be viewed and modified in the Properties view, by selecting the **create** operation in the Mobile Application Diagram, then the **Definition** tab in the Properties view.

```
INSERT INTO sampledb.dba.customer2
(id,
fname,
lname,
address,
city,
state,
zip,
phone,
company_name)
VALUES
(
(['id "=" :id'],
['fname "=" :fname'],
['lname "=" :lname'],
['address "=" :address'],
['city "=" :city'],
['state "=" :state'],
['zip "=" :zip'],
['phone "=" :phone'],
['company_name "=" :company_name']
)
```

3. The primary key column "id" is not autoincrement (or identity type), and you must manually enter the appropriate SQL statement, since the Fill from attribute setting in this case is not supplied automatically. From the Definition tab in the Properties view, click **Edit** and modify the statement as follows:

```
INSERT INTO sampledb.dba.customer2
(id,
fname,
```

```

lname,
address,
city,
state,
zip,
phone,
company_name)
VALUES
('["id"=:id]',
["fname"=:fname]',
["lname"=:lname]',
["address"=:address]',
["city"=:city]',
["state"=:state]',
["zip"=:zip]',
["phone"=:phone]',
["company_name"=:company_name]']
)
SELECT * FROM sampledb.dba.customer WHERE id=:id

```

This serves as the parent MBO's insert statement in the relationship, which returns the inserted row.

4. Define the relationship to the second MBO used in the chained insert operation as a **Composite** and **One to many**.

Map the child MBO's foreign key attribute to the parent MBO's primary key, "id" in the above example.

Binding an SAP Data Source to a Mobile Business Object

Bind the attributes and operations of an SAP object or Business Application Programming Interface (BAPI) to a mobile business object.

1. In the Bind Data Source wizard, enter the following information and click **Next**:

Field	Action
Data source type	Select SAP.
Connection profile	Select the SAP connection profile to which you are binding your mobile business object attributes/operations. If the required data source is not in the list, click Create to define a new connection profile.

2. Enter the SAP definition information. You can bind attributes and operations using this screen.

Field	Action
Method definition	<p>Browse for the BAPI or RFC operation from which the attribute or operation is bound. Once selected, the wizard automatically retrieves the meta data (for example, parameters, and input/output tables of the specified BAPI/RFC operation) and builds models for the MBO instance in the table definition.</p> <p>If you select Browse, the wizard displays a tree that represents the data source from which you can access the BAPI or operation. You can enter text to filter specific SAP business objects, or use the Search BAPIs/RFCs dialog box. Once you select the business object of interest, the associated BAPI operations are listed in the table. Select the operation and click OK.</p>
Parameters definition	Select the parameters, structure, input tables, or output tables of the BAPI operation and choose one or more table (or columns inside the tables), or output parameters to be mapped as mobile business object attributes.
Runtime data source credential and connection properties	If the data source is protected, and if the user name and password (data source access credentials) are different than the selected connection profile, you need to enter the User Name and Password that provides access. You can also set default values, or use personalization keys for all JCo connection properties.
Result checker	<p>Optionally add a result checker to the MBO.</p> <p>A result checker is a custom Java class that implements error checking for mobile business objects (MBOs). See <i>Adding a Result Checker</i>.</p>
Result set filters	<p>Optionally add a result set filter to the MBO.</p> <p>A result set filter is a custom java class that manipulates the rows or columns of data returned from a read operation for an MBO. To write a filter, developers must have previous experience with Java programming — particularly with the reference implementations for <code>javax.sql.RowSet</code>, which is used to implement the filter interface. See <i>Result Set Filters</i>.</p>
Preview	Click Preview to preview the newly defined mobile business object attributes.

3. Click **Finish** to use default mappings, or **Next** to modify the columns to attributes mapping, or parameter mapping.
4. The **Parameters Mapping** screen provides a graphical view of the parameter-to-remote operation mappings. Modify the default parameter mappings by using **Add**, **Delete** and **Delete all**. Make a connection between parameters by dragging a line from the source to the target. Parameter properties include:
 - **Parameters**
 - Parameter name – name of the mobile business object parameter. Names cannot contain C# or Java reserved words.
 - Datatype – enter the datatype of the parameter. It must be compatible with the datatype of the remote parameter to which it is mapped.
 - Nullable – accepts Null as a valid value.

- **Data Source**

- Argument – include any arguments to be passed to the parameter used by the MBO operation on the Unwired Server side when executing an MBO operation.
- Datatype – displays the datatype to which the parameter is mapped.
- Nullable – the data source column to which the parameter maps accepts Null as a valid input. When the argument is nullable, you can select <NULL> from the drop-down list. You would not select this option for a primary key.

Click **Next** or **Finish** when done.

5. The **Attributes Mapping** screen provides a graphical view of the columns to table mappings. You can collapse and expand the view and click the navigational buttons to rearrange attributes, remove and add individual attributes, and remove a mapping or all mappings. Click **Next** or **Finish** when done.

You can remove mappings without removing associated attributes from the graphical view only.

6. The **Role Assignments** screen allows you to **Create**, **Add**, and **Remove** role assignments from the mobile business object. Click **Finish** when done.

See also

- *Creating an SAP Connection Profile* on page 39
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 106
- *Implementing SSO for SAP* on page 87
- *Result Set Filters* on page 211

Searching for SAP BAPIs and RFCs

Use regular expression pattern matching to locate SAP BAPIs and RFCs from the Search BAPIs/RFCs dialog box.

The **Search BAPIs/RFCs** dialog used to locate SAP BAPIs and RFCs that bind to mobile business object attributes and operations searches for regular expressions, which should not be confused with wildcard syntax used in other search dialogs (the LIKE clause of a SQL query, or an operating system's file name search mechanism for example).

By convention, all custom RFCs have names beginning with Y or Z. If you were to enter "Z*" as the search string, the BAPIs and RFCs returned would match Z, ZZ, ZZZ, and so on. To search for all BAPIS and RFCs that begin with "Z", enter "Z.*" in the **Search BAPIs/RFCs** dialog. To find all custom RFCs enter "[YZ].*".

Configuring the SAP AutoCommit Feature

Configure an SAP operation, so `commit` is always called after the operation succeeds.

For an SAP operation, if the AutoCommit feature is enabled (`requireCommit` property is set to true), `commit` is always called following a successful operation; if the operation fails, changes are rolled back. By default, the AutoCommit feature is enabled; if disabled, you must explicitly call `commit` after the operation succeeds.

1. In the Mobile Business Object Properties dialog, select **Attributes** or **Operation**, then click the **Definition** tab.
2. Click **Edit**.
3. To enable the AutoCommit feature, check **Commit SAP Operation**; to disable, uncheck **Commit SAP Operation**.
4. Click **OK**.

Modifying SAP Connection Properties

Use SAP connection properties as mobile business object (MBO) parameters when making connections from an SAP MBO to an enterprise information system (EIS).

Modify SAP Java connector (JCo) connection properties when creating or editing an SAP MBO. You can then use these connection properties as parameters, for example, as personalization keys or default values.

Modify connection properties either from the Mobile Business Object Creation wizard or the Properties view.

1. In the Definition window of the Mobile Business Object Creation wizard, expand **Runtime Data Source Credential and Connection Properties**.
2. The Connection Properties table displays the configurable connection properties. The Property column is read-only, but you can modify some SAP JCo connection properties.

For example, you can either input a value or select a personalization key from the drop down list for the property. You can also create a new personalization key for any property. However, you cannot input a new value for language or code page properties.

3. To modify connection properties after the MBO is created:
 - a) From the Properties view, select the **Attributes** tab on the left, then the **Definition** tab.
 - b) Click **Edit**.
 - c) Modify as required and click **OK**.

Once you have created or modified your MBO, you can use these parameters as default values and personalization keys as needed.

At runtime, you can manage SAP connection properties as parameters. For example, if you have an SAP **SalesOrder.CreateFromData1** operation that inserts a sales order for a particular user that occurs in the context of a known SAP user, the user's credentials can be used in the insert operation. User credentials are checked in this order:

1. SAP single sign-on (SSO2) tokens.
2. X.509 single sign-on certificates.
3. Constant and personalization key values for each property.

Modifying SAP BAPIs that Contain Namespaces so they are Valid In Unwired WorkSpace

Unwired WorkSpace does not allow a structure datatype to contain names with special characters (a forward slash '/' in this case), because the forward slash is used as a Java class name after deployment and code generation.

1. After creating MBOs from SAP BAPIs that contain namespaces, modify the definition by removing any forward slashes.

For example, If the BAPI contains a structure

```
/SAPPO/BAPI_ORDER_OPEN
```

, rename the structure.

2. In this case you could remove the /SAPPO/ prefix.

Modify the structure datatype names before making any other changes to the MBO, since renaming the structure type name or structure attribute recreates the parameter of the MBO or operation parameter, causing the previous definition to be lost.

Binding a Web Service Data Source to a Mobile Business Object

Bind the attributes and operations of a Web or SOAP service to a mobile business object.

1. In the Bind Data Source wizard, enter the following information and click **Next**:

Field	Action
Data source type	Select one of: <ul style="list-style-type: none"> • Web Service • SOAP – supported styles of service are document/literal and rpc/literal.
Connection profile	If the data source type is Web Service, select the Web Service connection profile to which you are binding your mobile business object attributes. If the required data source is not in the list, click Create to define a new connection profile.

2. Depending on the selected data source (Web service or SOAP service), enter the attributes definition information in the Attributes wizard. This information translates into the detailed tabular view of the Web service which is mapped to mobile business object attributes.

Field	Action (Web service data source)
Method	The Web service method of the data source that is to be mapped to the mobile business object. To change the method, select a different one from the drop-down list.

Field	Action (Web service data source)
Binding	A binding defines message format and protocol details for operations and messages defined by a particular portType. Since multiple ports can be associated with a single binding, a default port is selected. To change the binding, select a different port from the drop-down list.
Configure XSLT	<p>To map Web service operations to mobile business object attributes, the Web service response message is converted to a table format. Select the Configure XSLT button to display the Define XSLT dialog where you can define the XSLT used to flatten the response messages. Options include:</p> <ul style="list-style-type: none"> • Generate XSLT from response message elements selection – select the elements and attributes to be mapped from the generated list. When a complex type has one or more nested type structures, the first nested structure is automatically selected. Selecting an array (list) of a child node that is not under the same tree is not supported and generates an error. • Define XSLT manually – includes the elements and attributes you have selected in a file format: <ul style="list-style-type: none"> • Save to file – modify by changing the selections and save to a file. • Load from file – since it is difficult to type in the XSLT contents accurately, this option allows you to retrieve the XSLT file from the file system. If you choose this option the XSLT text in the window is replaced by the information contained in the selected file. • OK – saves your changes and exits the dialog. <hr/> <p>Note:</p> <ul style="list-style-type: none"> • A Web service response containing an <code><s:any></code> node may result in two columns being generated. Manually remove the least relevant entry in these cases. • When Unwired WorkSpace generates a default XSLT for transformation of the output of a Web service operation, each result field in the XSLT includes an <code>op_bindpath</code> setting. If you modify the XSLT, leave the <code>op_bindpath</code> intact, since it's value is required to match columns from the transformed result set with fields from the original enterprise information system (EIS)-returned XML response element.
HTTP Basic Authentication	<p>Select this option and enter the user name and password in the corresponding fields to provide basic authentication to the Web service URL before gaining access to the Web service method.</p> <p>The user name and password fields can be entered directly, retrieved from a personalization key, or by selecting New key and creating a new personalization key.</p>
Result Checker	<p>Optionally add a result checker to the MBO.</p> <p>A result checker is a custom Java class that implements error checking for mobile business objects (MBOs). See <i>Adding a Result Checker</i>.</p>

Field	Action (Web service data source)
Result Set Filters	Add a custom java class to manipulate (filter) the rows or columns of data returned to the MBO. See Result Set Filters.
Preview	Click Preview to preview the results of the WSDL method invocation. You must enter values for the WSDL method parameters, and/or optionally set the preview/Test Execute configurations.

Field	Action (SOAP Service data source)
Input SOAP message	The input SOAP message from which the mobile business object attributes are derived.
Destination URL	<p>The URL from which the SOAP message is accessible. Include the Action URI (identifies the intent of the SOAP message) and Method name (identifies the SOAP method/operation).</p> <p>When you fill in the Input SOAP message and Destination URL fields, and implement the XSLT, the Next and Finish buttons are enabled. If you click Finish at this point, a default set of attributes mappings are automatically generated based on the SOAP message definition and you are placed in the Operation editor, from which you can define the mobile business object using the fields described for defining a Web service. Or, complete the remaining fields and click Next.</p>
Configure XSLT	<p>To map Web service operations to mobile business object attributes, the Web service response message is converted to a table format. Select the Configure XSLT button to display the Define XSLT dialog where you can define the XSLT used to flatten the response messages. Options include:</p> <ul style="list-style-type: none"> • Generate XSLT from response message elements selection – select the elements and attributes to be mapped from the generated list. When a complex type has one or more nested type structures, the first nested structure is automatically selected. Selecting an array (list) of a child node that is not under the same tree is not supported and generates an error. • Define XSLT manually – includes the elements and attributes you have selected in a file format: <ul style="list-style-type: none"> • Save to file – modify by changing the selections and save to a file. • Load from file – since it is difficult to type in the XSLT contents accurately, this option allows you to retrieve the XSLT file from the file system. If you choose this option the XSLT text in the window is replaced by the information contained in the selected file. • OK – saves your changes and exits the dialog. <p>Note: A Web service response containing an <code><S:any></code> node may result in two columns being generated. Manually remove the least relevant entry in these cases.</p>

Field	Action (SOAP Service data source)
HTTP Basic Authentication	Select this option and enter the user name and password in the corresponding fields to provide basic authentication to the Web service URL before gaining access to the Web service method. The user name and password fields can be entered directly, retrieved from a personalization key, or by selecting New key and creating a new personalization key.
Result Checker	Use the predefined Default or None result checker, or define and use a Custom result checker. See <i>Adding a Result Checker</i> .
Result Set Filters	Add a custom Java class to manipulate (filter) rows or columns of data returned for an MBO. See <i>Result Set Filters</i> .
Preview	Click Preview to preview the results of the SOAP method invocation. You must enter values for the SOAP method parameters, and/or optionally set the preview/Test Execute configurations.

3. Click **Finish** to use default mappings, or **Next** to modify the columns to attributes mapping, or parameter mapping.
4. The **Parameters Mapping** screen provides a graphical view of the parameter-to-remote operation mappings. Modify the default parameter mappings by using **Add**, **Delete** and **Delete all**. Make a connection between parameters by dragging a line from the source to the target. Parameter properties include:
 - **Parameters**
 - Parameter name – name of the mobile business object parameter. Names cannot contain C# or Java reserved words.
 - Datatype – enter the datatype of the parameter. It must be compatible with the datatype of the remote parameter to which it is mapped.
 - Required – The parameter is required. For example, a deployed MBO that contains user name and password parameters would require this information be supplied by the client.
 - **Data Source**
 - Argument – include any arguments to be passed to the parameter used by the MBO operation on the Unwired Server side when executing an MBO operation.
 - Datatype – displays the datatype to which the parameter is mapped.
 - Default value – enter a default value for the argument, or select <no default value> from the drop-down list.

Click **Next** or **Finish** when done.

5. The **Attributes Mapping** screen provides a graphical view of the columns to table mappings. You can collapse and expand the view and click the navigational buttons to rearrange attributes, remove and add individual attributes, and remove a mapping or all mappings. Click **Next** or **Finish** when done.

You can remove mappings without removing associated attributes from the graphical view only.

6. The **Role Assignments** screen allows you to **Create, Add, and Remove** role assignments from the mobile business object. Click **Finish** when done.

See also

- *Result Set Filters* on page 211

Creating Multi-level Insert Operations for Web Service Mobile Business Objects

Create a multi-level insert operation for two Web service mobile business objects (MBOs).

In this example, you have two MBOs, Order and OrderItem, that both have defined insert (create) operations: the OrderItem.insert operation requires the Order.id, but Order.id is assigned by the enterprise information system (EIS) and not available until the order is created in the EIS. You can create a multilevel insert operation to address this problem. When creating the multilevel insert operation:

- Ensure that Order.insert operation returns a resultSet that has the newly created Order.Id as one of the columns.
- Chain the two insert operations by creating the appropriate relationship.
- Ensure the association from Order to OrderItem is from Order.id.
- Ensure consistent naming: the **Primary key** attribute of Order (ID) must match the ID parameter of OrderItem.insert.

1. Create a Web service connection profile to the data source from which you created the MBOs.
2. Create attributes of the parent MBO (Order). For example, you can drag and drop the Web service data source onto the Mobile Application Diagram, and use the Quick Create wizard to define the MBO.

Define the MBO operation (create/insert).

Note: Web service multilevel inserts support SOAP bindings only.

3. Click **Finish**.
4. Set or verify the **Fill from attribute** setting:
 - a) In the Mobile Application Diagram, double-click the operation that serves as the insert operation for the parent MBO.
 - b) From the left side of the Properties view, select the **Parameters** tab.
 - c) Verify that each parameter name has a corresponding **Fill from attribute** value defined.

All parameters of the create operation in the parent MBO and the child MBO must be set to the related **Fill from attribute** value. By default, the related value is set automatically, but in some cases the value cannot be found, so double check the values.

5. Set or verify the **Primary key** setting:
 - a) In the Mobile Application Diagram select within the header of the MBO to view the MBO properties in the Properties view.
 - b) From the left side of the Properties view, select the **Attributes** tab located on the left, then the **Attributes Mapping** tab located on the top.
 - c) Locate and select the **Primary key** check box for the attribute that serves as the primary-key equivalent for the parent MBO (for example, Id).
6. Create the child MBO (OrderItem) the same way you created the parent – drag and drop the data source onto the Mobile Application Diagram, and follow the Quick Create wizard instructions to create the attributes and operations.
7. From the Properties view, verify that each operation's (insert) parameter name has a corresponding **Fill from attribute** setting.
8. In the Mobile Application Diagram, click **Relationship**, and use the wizard to define a relationship between the MBOs. For example, link the Source object Order "Id" attribute to the Target object OrderItem "Id." Select **Composite** and **One to many**.

Synchronization of the child MBO should occur either independently or through the parent MBO. See the *Client Object API* documentation for details.

Web Service Mobile Business Object Limitations

Understand Web service mobile business object (MBO) limitations.

This section describes known limitations when binding an MBO to Web service data sources.

Unsupported types

These Web service types are currently unsupported:

- Recursive definitions of element types.
- Soapenc encoded arrays.
- Derived types using List/Union constructs.
- anyType
- gDay, gYear, gMonth, gYearMonth
- HexBinary datatypes are not supported, instead use base64Binary.
- Unsupported datatypes are either ignored or recognized as a String.

Unsupported derived or complex types

These complex datatype scenarios are currently unsupported for Web service MBOs:

- This structure is not supported:

```
Structure A[]:
x of type struct B[].
(y of type struct C[])
```

but does support:

```
Struct A[]:
```

```
< scalar type > x,  
y of type Struct B[]
```

Unsupported schema constructs

- Schema used as element data (i.e. an element is a schema).

Unsupported Web service operations

- One way operation

MBO Mapping restrictions

MBO attribute to Web service column mapping restrictions include:

- If an attribute maps to a table column and is intended to be an alternate key, specify a smaller maxlength value for it as compared to any other non-key attribute (try to match the actual length to the enterprise information system (EIS) column to which it maps). This is a client-side implementation requirement, because when the row size (addition of all column sizes(maxlengths)) exceeds the page size, the client converts columns with higher maxlengths to `long varchar`. This creates a run-time exception if the client creates an index on each column which is or part of an alternate key.
- Do not define alternate keys on columns that are `long varchar/binary` datatypes since device databases (Ultralite) do not support it.
- For WSDL and SOAP Web service data sources , if an Update operation that uses the **Apply results to the cache** cache update policy does not have an XSLT, an error message displays:

```
The 'UPDATE' operation 'Customer- > updateStudent()' with 'Apply results to the cache' cache update policy does not have XSLT.
```

Configuring an SAP Exposed Web Service MBO to Use Credentials

Enable SSO with X.509 certificates or SSO2 tokens for SAP BAPIs that are exposed as Web services.

1. From Unwired WorkSpace, define a connection profile for the Web service MBO:
 - a) In the Web Service Connection Details dialog, select **From URL** and enter the URL of the SAP BAPI that is exposed as a Web service.
 - b) Select **Enable HTTP authentication**.
 - c) Enter a user name and password used for authentication.
2. Define the attributes of the Web service MBO:
 - a) Connect to the Web service connection profile.
 - b) Expand the connection profile and drag-and-drop the interface for which you are creating an MBO.
 - c) From the Definition scree, select **HTTP Basic Authentication**.

- d) In the HTTP Basic Authentication fields, specify the system provided default **username** and **password**.

Next

Deploy the MBO and configure the Workflow application to use either X.509 or SSO2 credentials.

See also

- *Creating an SAP Connection Profile* on page 39
- *Binding an SAP Data Source to a Mobile Business Object* on page 96
- *Implementing SSO for SAP* on page 87

Accessing a Web Service from an HTTPS Port

To access an SAP BAPI exposed as a Web Service from an HTTPS port, add the same SAP server certificate you imported into the Unwired Server truststore into an Unwired WorkSpace truststore.

Perform this configuration on the Unwired WorkSpace host, or an SSL security exception is returned when trying to connect to the connection profile of the HTTPS WSDL URL.

1. Add `truststore.jks` to the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Eclipse` directory.
2. Use the Java **keytool** command to add the same SAP server certificate you imported into the Unwired Server truststore into `truststore.jks`.
3. Add this argument to the `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Eclipse\UnwiredWorkspace.bat` file:

```
-Djavax.net.ssl.trustStore=%ECLIPSE_ROOT%\truststore.jks
```

4. Create and run a `setup-truststore.bat` file that sets up your truststore environment. For example:

```
copy ..\JDK1.6.0_23\jre\lib\security\cacerts truststore.jks
@echo *
@echo * Answer Yes to "Trust this certificate?".
@echo *
keytool -import -keystore truststore.jks -file sap-doe-
vml.sybase.com.PEM.crt -storepass changeit
@echo *
@echo * Add to eclipse vmargs: -Djavax.net.ssl.trustStore=
%ECLIPSE_ROOT%\truststore.jks
@echo *
```

5. Restart Unwired WorkSpace.

Binding a REST Web Service Data Source to a Mobile Business Object

Bind the attributes and operations of a REST (Representational State Transfer) Web service to a mobile business object.

The main difference with binding to a REST Web service data source and any other Web service data source is the definition page. Other pages (parameter mapping, attributes mapping, role mapping, and so on), are the same. Successfully invoking a REST Web service requires:

- A URI template.
- An HTTP method.
- A request representation using XSD.
- A response representation using XSD. For Read operation attributes, the response representation is mandatory. For Create, Update, Delete and Other operations, the request and response representation is optional.
- An optional XSLT if necessary, except for Read operation attributes, for which an XSLT is mandatory. You can also can define multiple-XSLTs for a commingled MBO (multiple MBOs from a single operation).
- HTTP headers declaration.
- Optional authentication configuration.

1. In the Bind Data Source wizard, enter the following information and click **Next**:

Field	Action
Data source type	REST Web Service
Connection profile	Select the REST Web Service connection profile to which you are binding your mobile business object attributes. If the required data source is not in the list, click Create to define a new connection profile.

2. In the Definitions page, enter the attributes definition information.

This information translates into the detailed tabular view of the REST Web service that is mapped to mobile business object attributes:

Tab	Description
Resource base URL	The base URL for a REST Web service resource URL, required to construct the resource URI. For example, http://example.com. The base URL was defined in the connection profile, and is shown (read-only) in the definition page for attributes or operations.

Tab	Description
Resource URI template	<p>The URI template is appended to the base URL (for example, customers/{id}). The template determines how the URI is parsed so all possible parameters are retrieved. All parameters are treated as enterprise information system (EIS) operation arguments, enabling MBO create, read, update, and delete (CRUD) operations on the REST Web service resources.</p> <p>You can specify the datatype for parameters defined in the URI template using this format: {paramName (xsdType) } For example, '/getCustomer/{id(int)}'. A string datatype is used if you do not explicitly specify the datatype. You can also specify support of nullable for the parameter as: '/getCustomer/{id(int?)}'. These datatypes can be specified in the URI template and are case-sensitive:</p> <ul style="list-style-type: none"> • boolean • string • binary • char • byte • short • int • long • integer • decimal • float • double • date • time • dateTime <p>You cannot specify the length in the URI template for parameters, but can modify parameter length after the REST MBO is created from the Properties view.</p>
HTTP method	The method called, either GET, PUT, POST, or DELETE.
Expected status code	<p>All available HTTP status codes are shown in the drop-down list. The default is 200. If you change it to NONE, it means that the returned status code is not checked.</p> <p>For each EIS REST Web Service operation, you can choose an HTTP status code as the expected status code. If the returned status code is the same as the specified expected status code, the EIS operation was executed successfully.</p> <p>If the returned status code does not match the specified expected status code, the EIS operation fails.</p>

Tab	Description
Representation	<p>Allows you to specify the request or response representation. Select the representation type:</p> <ul style="list-style-type: none"> • Request • Response <p>Enter or select:</p> <ul style="list-style-type: none"> • Name – name of the representation, which must be unique. • Referenced representation – All existing representations defined for the same MBO are listed and can be selected, but only the same Type of representation can be referenced. For example, if you previously defined a request representation for a create EIS operation, you can use it as a referenced request representation to define a new update representation for a EIS operation. <p>Click Edit to specify the XML schema for a request or response representation, instead of referencing an existing representation:</p> <ul style="list-style-type: none"> • Name – name of the XSD. • Type – Request or response. • XSD URL – URL to an available XSD. • XSD file – allows you to import the XSD from a local file. • Root element – The root element used as the XSD request input. • Load element – Select this after specifying the XSD URL or XSD file, to retrieve all available load elements defined in the XSD. <p>Only one XSD can be defined for a given representation.</p> <p>Define XSLT manually – for response representation, after the XSD definition and root element was specified, Unwired WorkSpace automatically generates a default XSLT for the response representation. You can open the XSLT editing dialog to modify the XSLT. Modifying multi-XSLTs for one response representation to support commingled MBOs is also supported. All available XSLT definitions are listed in the table in the XSD definition dialog:</p> <ul style="list-style-type: none"> • Save to file – modify by changing the selections and save to a file. • Load from file – since it is difficult to type in the XSLT contents accurately, this option allows you to retrieve the XSLT file from the file system. If you choose this option the XSLT text in the window is replaced by the information contained in the selected file. <p>Select OK to save your definition and exit the dialog.</p>

Tab	Description
Authentication	<p>Supports HTTP Basic Authentication:</p> <ul style="list-style-type: none"> • User name – authenticated user • Password – password of the authenticated user. <p>The user name and password fields can be entered directly, retrieved from a personalization key, or by selecting New key and creating a new personalization key.</p>
HTTP header	<p>Captures all properties needed to execute or manage the REST Web Service.</p> <p>You can enter the existing HTTP header as defined in the HTTP specification, or declare your own HTTP header. For each element of the HTTP header you can:</p> <ul style="list-style-type: none"> • Input/Output – specify if it is used as an input, output or both. • Variable – specify a literal value as the HTTP header's value. Specify a "Variable" to denote it as a 'parameterized' header. For parameterized headers, Unwired WorkSpace parses and treats the value as an input argument or output column of the EIS operation (depending on whether the definition is input or output). • Value – specify the header value. For example application/xml. <p>The Input/Output/Both setting determines if Variable and Value are enabled:</p> <ul style="list-style-type: none"> • Input – Variable is enabled: <ul style="list-style-type: none"> • If you select Variable, the Value field is disabled and cleared. • If you unselect Variable, the Value field is enabled, and you must specify a value. • Output – Variable is an automatically selected read-only field. The Value field is disabled and cleared. • Both – Variable is an automatically selected read-only field. The Value field is disabled and cleared.

3. Click **Finish** to use default mappings, or **Next** to modify the columns to attributes mapping, or parameter mapping.
4. The **Parameters Mapping** screen provides a graphical view of the parameter-to-remote operation mappings. Modify the default parameter mappings by using **Add**, **Delete** and **Delete all**. Make a connection between parameters by dragging a line from the source to the target. Parameter properties include:
 - **Parameters**
 - Parameter name – name of the mobile business object parameter. Names cannot contain C# or Java reserved words.
 - Datatype – enter the datatype of the parameter. It must be compatible with the datatype of the remote parameter to which it is mapped.

- **Required** – The parameter is required. For example, a deployed MBO that contains user name and password parameters would require this information be supplied by the client.
- **Data Source**
 - **Argument** – include any arguments to be passed to the parameter used by the MBO operation on the Unwired Server side when executing an MBO operation.
 - **Datatype** – displays the datatype to which the parameter is mapped.
 - **Default value** – enter a default value for the argument, or select <no default value> from the drop-down list.

Click **Next** or **Finish** when done.

5. The **Attributes Mapping** screen provides a graphical view of the columns to table mappings. You can collapse and expand the view and click the navigational buttons to rearrange attributes, remove and add individual attributes, and remove a mapping or all mappings. Click **Next** or **Finish** when done.

You can remove mappings without removing associated attributes from the graphical view only.

6. The **Role Assignments** screen allows you to **Create**, **Add**, and **Remove** role assignments from the mobile business object. Click **Finish** when done.

REST Web Service Mobile Business Object Limitations

Understand REST Web service mobile business object (MBO) limitations.

- If an Update operation using the **Apply results to the cache** cache update policy does not have a response representation, an error displays:

```
The 'UPDATE' operation 'Customer- > updateStudent()' with 'Apply results to the cache' cache update policy does not have response representation.
```

- If the response representation of an Update operation using the **Apply results to the cache** cache update policy does not have an XSLT, an error message displays:

```
The response representation of the 'UPDATE' operation 'Customer- > updateStudent()' with 'Apply results to the cache' cache update policy does not have an XSLT defined.
```

Rebinding Data Sources to Mobile Business Objects

Use the Bind Data Source wizard in the Properties view to change a mobile business object's binding to a different data source.

The binding of data source information to MBO attributes and operations varies, depending on the data source to which you are binding.

1. In the Properties view, select the **Attributes** or **Operations** tab, then click **Bind Data Source**.

A prompt informs you that the attributes are already bound to a data source.

2. Click **Yes** to rebind to a different data source.
3. Follow the Bind to a Data Source wizard instructions to bind to a different data source.
You can bind to the same or different type of data source depending on the original MBO and binding. The default mappings change accordingly.

Changing a Data Source's Connection Profile

Change a data source binding from one data source, to another data source of the same type, using the Properties view.

Prerequisites

Change a connection profile binding by selecting a different connection profile of the same data source type. The connection profile to which you are binding must be available.

Task

1. You can modify connection profile to MBO attributes and operation bindings from the Properties view: there are a number of places from which you can change the connection profile to which the MBO is bound, depending on the context you are in, by selecting the **Change Connection Profile** button.

The Change Data Source dialog displays. The **Data source type** field is read-only in this dialog, you can only change the connection profile.

2. Select a connection profile from the drop-down list to switch to a different connection profile of the same data source type.

Adding a Result Checker

Add a result checker to implement operation result processing logic in custom code. The deployed result checker logically validates operation results and produces log records for device clients or the server package log.

See the *Developer Guide for Unwired Platform* for information about writing a custom result checker.

Add a result checker when you edit Attribute or Operation properties for a mobile business object derived from a data source. Add a result checker after you have either written a custom one or use a predefined one in Unwired Workspace (the latter of which can be configured when you create an object).

1. In the New Attributes or New Operation wizard, in the Result checker section, select from these options:

Option	Description
Default	<p>The result checker depends on the data source type:</p> <ul style="list-style-type: none"> • SAP – <code>com.sybase.sup.sap3.DefaultSAPResultCheck</code>. If a RETURN parameter is found in the selected operation, this option is automatically selected. • Web service (SOAP) – <code>com.sybase.sup.ws.soap.DefaultWSResultCheck</code>. The default checker always returns the status as successful. • Web service (RESTful) – <code>com.sybase.sup.ws.rest.DefaultRestResultCheck</code>. The default checker always returns the status as successful.
None	<p>Return the status as successful with no message. The result checker used depends on the data source type:</p> <ul style="list-style-type: none"> • SAP – <code>com.sybase.sup.sap3.NoOpSAPResultCheck</code> • Web service (SOAP) – <code>com.sybase.sup.ws.soap.NoOpWSResultCheck</code> • Web service (RESTful) – <code>com.sybase.sup.ws.rest.NoOpRestResultCheck</code>
Custom	<p>Specify a custom result checker.</p> <p>See the topic <i>Writing a Custom Result Checker</i> in the Developer Guide for Unwired Server for information about writing a custom result checker.</p>

2. (Optional) If you have not yet created the result checker classes, select **Custom** in the Result checker area of the New Attributes or New Operation dialog, and click **Create** to run the New Java Class wizard.
3. If prompted, add a Java nature.
 - a) (Recommended) Click **Yes** to add a Java nature. In Eclipse, a Java nature adds Java-specific behavior to projects.

In the New Java Class wizard, enter:

Option	Description
Source folder	By default, this is the <code>Filters</code> folder from your project. Click Browse to locate the source folder for the Java class.
Package	<p>Click Browse to locate the package for the new Java class.</p> <p>Note: Sybase recommends that you do not leave this field blank. Otherwise, the <code>JDK<version></code> Java class in the default package cannot be resolved in other packages.</p>

Option	Description
Enclosing type	Choose a type in which to enclose the new class. You can select either this option or the Package option, above. Enter a valid name or click Browse .
Name	Enter a name for the result checker class.
Modifiers	Select the Java class modifiers. The default modifier is public.
Superclass	<ol style="list-style-type: none"> 1. Click Browse. 2. In the Superclass Selection dialog, enter: <ul style="list-style-type: none"> • Choose a Type • Matching Items 3. Click OK.
Interfaces	<p>By default, this is populated with the corresponding interface:</p> <ul style="list-style-type: none"> • SAP – com.sybase.sup.sap.SAPResultChecker <hr/> <p>Note: The default SAP interface does not work. See <i>Using the Correct Java Connector Version For the SAP Custom Result Checker</i>.</p> <ul style="list-style-type: none"> • Web service (SOAP) – com.sybase.sup.ws.soap.WSResultChecker • RESTful Web services – com.sybase.sup.ws.rest.ResultChecker <p>Click Add to select interfaces implemented by the new class.</p>
Which Method Stubs Would You Like to Create	<ul style="list-style-type: none"> • Public Static Void Main • Constructors From Superclass • (Default) Inherited Abstract Methods
Do You Want to Add Comments	Select Generate Comments to add comments. From here, you can modify the preferences of the code templates by clicking Configure templates and default values .

- b) Click **No** if you do not want to add the Java nature to the selected mobile application project.

- c) Click **Finish** to compile the java skeleton source file and add the skeleton Java checker class to the MBO.
The result checker appears next to the Custom option.
4. In the Result checker section, next to the Custom option, click **Browse** to find an existing result checker class.
 - a) In the Select Result Checker Class dialog, select the result checker class and click **OK**.
The result checker class appears next to the Custom option.
5. Validate the result checker:
 - a) To reuse input values you have already saved for previous previews, select **Existing Configuration**. Otherwise, load defaults, or create a new set of input values expressly for this preview instance.
 - b) Click **Preview**.

If the data runs successfully, *Execution Succeeded* appears at the top of the Preview dialog and data appears in the **Preview Result** window.

Using the Correct Java Connector Version For the SAP Custom Result Checker

SAP Java Connector (JCo) libraries are installed with Sybase Unwired Platform: JCo 3.x libraries are installed in Unwired Server and JCo 2.x libraries with Unwired WorkSpace. This effects how you implement a custom result checker for SAP enterprise information systems (EIS).

1. Add the `sapjco3.jar` and `sup-ds.jar` files to the Java build path of the mobile application project to which you are adding the custom result checker:
This requires access to the Unwired Server installation, since it contains the `sapjco3.jar` file.
 - a) From Unwired WorkSpace, right-click the project in WorkSpace Navigator and select **Properties**.
 - b) If a custom result checker does not yet exist, you must add the Java Nature to the project. Select **Project Natures > Java > OK**, then right-click the project again and select **Properties**.
 - c) Select **Java Build Path**.
 - d) Select the **Libraries** tab, then **Add External JARs**.
 - e) Browse to and add these JAR files, if not already in the library path:
 - `<UnwiredPlatform_InstallDir>\sapjco\v3\32bit\sapjco3.jar`
 - `<UnwiredPlatform_InstallDir>\Servers\UnwiredServer\lib\ext\sup-ds.jar`
2. Add the custom result checker by editing the MBO definition, not when creating the MBO. For example:

- a) Open the mobile application project and select an existing MBO in the mobile application diagram.
- b) In Properties view, select **Operations**, the operation of interest, and click **Edit**.
- c) From the Edit Operation screen, select **Edit**.
- d) Expand **Result Checker** and select **Custom > Create**.
- e) In the Interfaces section of the New Java Class screen, remove the `SAPResultChecker` interface.
- f) Click **Add** and enter `SAPResultChecker`. Select the **com.sybase.sup.sap3 SAPResultChecker**.
- g) Enter a name for the result checker and, optionally, a package name.
- h) Click **OK** until you exit the screens, then save the project.

The Java stub file is created in the `Filters` folder within the project.

3. Remove the `sup-ds.jar` and `sapjco.jar` files that are automatically installed to the Java build path:
 - a) Right-click the project and select **Properties**.
 - b) Select **Java Build Path > Libraries**.
 - c) Select the `sup-ds.jar` and `sapjco.jar` files that were automatically installed and select **Remove**.

These files have `Unwired_WorkSpace` in their respective paths.

4. Add your custom code and deploy the project.

Editing the Result Checker

Change the result checker settings using the Change Definition dialog.

1. Right-click inside the Mobile Application Diagram and select **Show Properties View**, or select **Window > Show View > Properties**.
2. In the Properties view, click the **Attributes** or **Operations** tab, then the **Definition** tab.
3. Click **Edit**.
4. Make your changes in the Change Definition dialog, and click **OK**.

If you see this error message after defining an SAP custom result checker from an attribute and refreshing `Unwired WorkSpace`, click **OK** and ignore the message: `can't be cast to com.sybase.sup.sap.SAPResultChecker`.

SAP custom result checkers do not support the `Unwired WorkSpace Preview` option.

Deploying Result Checker Classes to Unwired Server

Deploy result checker classes to Unwired Server when you deploy mobile business objects (MBOs) that contain them.

If any of your MBOs includes a result checker (or other user-defined classes), the **Package User-defined Classes** page of the deployment wizard prompts you for the location of the Jar/class files to deploy to Unwired Server.

1. (Optional) To maintain and deploy a single file, create a Java archive of all your classes, or add an existing JAR file.
2. When finished with MBO development, deploy the MBOs to Unwired Server. Follow the prompts to include the result checker classes.

Refactoring a Result Checker

When a result checker is deleted, renamed, or moved, update its references automatically.

Deleting References to a Result Checker

Delete all references to a result checker from the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the result checker, and expand the **Filters** folder.
2. Right-click the result checker and select **Delete**.
3. In the Confirm Delete dialog, verify the selected references, and click **OK**.

Renaming a Result Checker

Rename a result checker, and update its references in the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the result checker you want to rename, and expand the **Filters** folder.
2. Right-click the result checker, and select **Refactor > Rename**.
3. In the Rename Type dialog, verify the changes, and click **Finish**.

Moving a Result Checker

Move a result checker to another location, and update its references in the workspace.

1. In WorkSpace Navigator, select the mobile application project that contains the Web result checker you want to move, and expand the **Filters** folder.
2. Right-click the result checker, and select **Refactor > Move**.
3. In the Move dialog, click **OK**.

Working with Mobile Business Objects

Create attributes and operations for mobile business objects, create relationships between mobile business objects, bind them to a back-end data sources, modify and test them.

See also

- *Developing a Mobile Business Object* on page 69
- *Packaging and Deploying Mobile Business Objects* on page 217
- *Mobile Business Objects* on page 70

Modifying Mobile Business Object Properties

Edit existing MBO operations, attributes, relationships, and so on, as well as Mobile Application Diagram properties from the Properties view.

1. Right-click inside the Mobile Application Diagram and select **Show Properties View**, or select **Window > Show View > Properties**.
2. Display and edit any of these properties from the Properties view by clicking in the Mobile Application Diagram.

Table 30. Properties view context

Edit	Action
Mobile business object	Click inside the MBO (or inside the title area), but do not click an attribute or operation.
Attributes compartment	Click the attributes compartment to show a different representation than the 'mobile business object' context – the Data source, Definition, Attribute Mapping, Parameters, and Roles tabs are laid out as vertical tabs instead of horizontal tabs. Note: If the mobile business object attributes are not bound to a data source, only the Data Source, Attributes, and Roles tabs display.
Attribute	Click an MBO attribute.
Operation	Click an MBO operation.
Relationship	Click the line (relationship) that connects two MBOs.
Mobile Application Diagram	Click inside the Mobile Application Diagram, but outside any object it contains.

3. Modify the properties and save your changes. You can select **File > Save**, enter CTRL-S, or select the Workbench save button (the floppy disk icon).

Any changes you make in the Properties view are immediately reflected in the Mobile Application Diagram, however you must save the changes or they are lost once you exit the Mobile Application Diagram.

See also

- *Creating Relationships Between Mobile Business Objects* on page 158
- *Previewing Mobile Business Objects* on page 155

Mobile Business Object General Properties

Perform general tasks related to mobile business objects (MBOs) and the Mobile Application Diagram.

Copying a Mobile Business Object

Create a copy of an existing mobile business object, attribute, or operation, which you can then modify as needed.

1. In the Mobile Application Diagram, right-click the mobile business object, attribute, operation or multiple objects (to select multiple MBOs, hold down SHIFT, then use the left mouse button to select additional MBOs) you want to copy so that the entire object is selected. Select **Edit > Copy**(or Ctrl-C).
A copy of the mobile business object is created in the Mobile Application Diagram.
2. Select **Edit > Paste**(or Ctrl-V) to create a copy of the original object or objects.
3. Change the name of the duplicated mobile business object, and modify any other contents as needed using the Properties view.

Deleting a Component of a Mobile Business Object

Delete a mobile business object, operation, attribute, or relationship.

From the Mobile Application Diagram, right-click the MBO, operation, attribute, or relationship, and select **Delete**.

Searching for Mobile Business Objects

Search for existing mobile business objects based on name, attributes, operations, relationships, and so on. Search results display in the Search view.

1. From the workbench menu, select **Search > Unwired WorkSpace**, or select the **Ctrl+H** keys.
2. In the Search dialog, define your search criteria and click **Search**.

Table 31. Search criteria

Option	Description
Search string	Enter a search string, including wildcard characters, used as the name pattern used as search criteria for your other selections. Use wildcards, "*" ; matches all, "?" matches single character.
Search for	Searches for a matching mobile business object, attribute, or operation.
Limit to	Limit the search to declarations, references, or all occurrences of the specified search.
Data source reference	Filter a search based on data source information; search for items that are bound to a data source, data source type, or connection profile name.
Role assignments	Select mode (And or Or) and enter specific role names. Both "*" and "?" wild cards are supported. Use ";" as the delimiter when entering multiple search strings. For example, "roleA;*B;C?".

Option	Description
Personalization key reference	Select an existing personalization key from the drop-down list. Both "*" and "?" wild cards are supported.
Resultset filter reference	Select an existing filter from the drop-down list. Both "*" and "?" wild cards are supported.
Scope	Limit the scope by mobile application project: <ul style="list-style-type: none"> • All – searches all projects • Enclosing – defines the scope as the currently selected elements • Single – select the project from the list.

Search results are displayed in the Search view.

Editing Multiple Rows of Table Information

Batch edit multiple rows of table information by selecting the desired rows, making the change, and applying the change to the selected rows.

You can batch edit table row information in these locations:

- Attribute mapping table (available from the wizard during MBO creation, and the Properties view during editing)
- Parameter mapping table (available from the wizard during MBO creation, and the Properties view during editing)
- Preview Dialog (for input parameters and input table data)

1. To edit a field in multiple table rows, select the rows, by either:
 - a) Left-clicking while pressing "Ctrl" to select individual rows, or
 - b) Selecting "Shift" and left-clicking a row to include all rows in between the two selections.
2. Make changes in a cell of one of the selected rows.
For example, if you are changing the datatype for all selected rows, right-click the datatype cell in any of the selected datatype rows and select the datatype.
3. To apply changes to the selected rows, change the focus by clicking outside the selected rows. Your changes are applied to all rows.

To undo changes made in the Properties view, select **Edit > Undo**. To undo your changes when creating the MBO or in the Preview dialog, repeat steps one and two.

Mobile Application Diagram Properties

Modify Mobile Application Diagram settings.

Table 32. Mobile Application Diagram properties

Tab	Contents
Appearance	From the Properties view, define various aspects of the Mobile Application Diagram's appearance, including: <ul style="list-style-type: none"> • Fonts and Colors – changes the appearance of the text displayed in the Mobile Application Diagram. • Title fonts and colors – changes the appearance of the titles displayed in the Mobile Application Diagram.

Resizing mobile business object compartments

You can resize the attributes and operations portion of a mobile business object from the Mobile Application Diagram by dragging the line that separates the compartments to make the compartment larger or smaller.

Managing Mobile Application Diagram Filters and Logical Groups

A filter defines a list of selected mobile business objects. If a filter is applied, only the mobile business objects defined in that filter appear. Logical groups keep selected mobile business objects and other logical groups together.

Because there is only one Mobile Application Diagram for all mobile business objects in a given project, filters and logical groups can be useful for viewing only mobile business objects of interest. You can create two types of filters:

- Filter by Mobile Business Objects – include only selected mobile business objects in the filter.
- Filter by Logical Group – include entire subfolders and their contents in the filter.

Creating a Mobile Application Diagram Filter

For a given Mobile Application Diagram, create a filter (diagram filter) to view only mobile business objects of interest.

Filters and diagram filters both perform the same function, the difference is creating a diagram filter allows you to select the MBOs, while creating a filter pre-selects MBOs.

1. If you are creating a diagram filter:
 - a) Right-click in an empty area of the Mobile Application Diagram and select **Diagram Filter**.
 - b) Click **Select Filters**, then **New** to launch the New Filter wizard.
 - c) Select the filter type:

- Mobile business objects – select individual mobile business objects
 - Logical group – select logical groups (subfolders) and the mobile business objects that the logical group contains.
- d) Enter a descriptive **Filter name**, and select the individual mobile business objects (or logical groups) to include in the filter. For logical groups, select **Recursively include all the sub logical groups** to include subfolder contents in the filter.
 - e) Click **Finish** to create the filter.
2. If you are creating a filter:
 - a) Select multiple MBOs in the Mobile Application Diagram (hold the Shift key while selecting individual MBOs).
 - b) While holding the Shift key (after selecting the last MBO), right-click and select **New > Filter**.
 - c) Enter a filter name. Verify, add, or remove MBOs to be included in the filter.
 - d) (optional) Select **Apply the filter immediately** to apply the filter as soon as you click Finish.
 - e) Click **Finish** to create the filter.

Editing a Mobile Application Diagram Filter

Modify an existing mobile application diagram filter.

When editing an existing mobile application diagram filter, you can change only the contents of the filter, not the type of filter.

1. Right-click in an empty area of the Mobile Application Diagram and select **Diagram Filter**.
2. Double-click the filter you want to modify or click **Edit**.
3. Modify the name or contents of the filter by selecting (or unselecting) the mobile business objects or logical groups that you want to add (or remove) from the filter, and click **OK**.
4. Click **OK** when done.

Selecting a Mobile Application Diagram Filter

Determine which mobile business objects to view in the Mobile Application Diagram by selecting a mobile application diagram filter.

1. Right-click in an empty area of the Mobile Application Diagram and select **Diagram Filter**.
2. Select **No Filter** to view all mobile business objects within the Mobile Application Diagram, or select **Select Filters** and select a filter to view only the mobile business objects defined by that filter within the Mobile Application Diagram.
3. Click **OK**.

You cannot arrange MBOs using any of the Arrange options when a diagram filter is applied.

Deleting a Mobile Application Diagram Filter

Delete a mobile application diagram filter.

1. Right-click in an empty area of the Mobile Application Diagram and select **Diagram Filter**.
2. Select the filter you want to delete and click **Remove**.
3. Click **OK**.

Creating a Logical Group

Create a logical group to keep selected mobile business objects together. Only mobile business objects and other logical groups can be part of a logical group.

You can create logical groups from the File menu, the Mobile Application Diagram, or WorkSpace Navigator.

1. If you are creating a logical group from the File menu:
 - a) Select **File > New > Logical Group**.
 - b) Select the Mobile Application folder to act as the parent folder of the logical group. By default the logical group is created in the parent's Mobile Business Objects subfolder. But you can also select an existing logical group folder as the parent. Enter the **Logical group name**.
 - c) Click **Finish** to create the logical group.
2. If you are creating a logical group from the Mobile Application Diagram:
 - a) Right-click an MBO and select **New > Logical Group**.
 - b) Select the MBOs to be included in the logical group and click **Next**.
You can right-click one MBO to create a logical group, or select several MBOs and right-click to create a logical group, to which the selected MBOs are included in the Logical Group wizard by default.
 - c) Select the Mobile Application folder to act as the parent folder of the logical group. By default the logical group is created in the parent's Mobile Business Objects subfolder. But you can also select an existing logical group folder as the parent. Enter the **Logical group name**.
 - d) Click **Finish** to create the logical group.
3. If you are creating a logical group from WorkSpace Navigator:
 - a) Expand the project folder, the Mobile Business Objects folder. Right-click any MBO and select **New > Logical Group**.
 - b) Select the MBOs to be included in the logical group and click **Next**.
You can right-click one MBO to create a logical group, or select several MBOs and right-click to create a logical group, to which the selected MBOs are included in the Logical Group wizard by default.
 - c) Select the Mobile Application folder to act as the parent folder of the logical group. By default the logical group is created in the parent's Mobile Business Objects subfolder.

But you can also select an existing logical group folder as the parent. Enter the **Logical group name**

- d) Click **Finish** to create the logical group.

The logical group displays under the Mobile Business Objects folder in WorkSpace Navigator.

Adding Mobile Business Objects to a Logical Group

A logical group can be used to manage a project that contains a large number of mobile business objects (MBOs).

1. From WorkSpace Navigator, expand the project folder and Mobile Business Objects folder that contains the logical group folder.
2. Drag-and-drop the MBOs to the logical group folder.

You can also drag MBOs from other projects, or drag an MBO out of a logical group folder into another logical group, or into the Mobile Business Objects folder.

Deleting a Logical Group

Delete a logical group and its contents.

1. From WorkSpace Navigator, expand the project folder and Mobile Business Objects folder that contains the logical group folder.
2. Right-click the logical group folder and select **Delete**.

Note: Deleting a logical group deletes its entire contents. If you want to save its contents (MBOs or any logical groups) drag them to another folder before deleting the logical group.

Mobile Business Object Data Properties

Modify mobile business object (MBO) data properties to customize mobile application business logic.

Datatype Support

Unwired WorkSpace supports a variety of datatypes, from a simple type to an array of objects.

Mobile business object (MBO) attributes and parameter datatypes map to data source datatypes. Select the datatype of a given parameter or attribute from the datatype drop-down list, which maps to the data source's datatype. You define attribute and parameter datatypes in a number of Unwired WorkSpace locations, depending on the MBO development phase, including:

- Creating MBOs – when creating MBO operations and attributes in these locations:
 - Attributes Mapping wizard

- Operation Parameters page (when deferring binding to a data source)
- Attributes page (when deferring binding to a data source)
- Editing MBOs – when editing MBO attributes and parameters from the Properties view in these locations:
 - Parameters tab
 - Load Parameters tab
 - Attributes tab
 - Synchronization tab (for synchronization parameters)
 - Object Queries tab and Object Query creation wizard
- Testing MBOs – use the Test Execute and Preview dialogs for testing mobile business object operations or previewing attributes.
- Creating and editing personalization keys – holds the personalization parameters and supports the same datatypes as MBO attributes and parameters.

Since attributes and parameters depend on the data source to which the MBO maps, not all attributes and parameters support all datatypes. Generally, if a datatype does not display in the drop-down list, it is not supported for that MBO.

When defining the default value for a parameter with the maxlength setting, the maximum length also applies to any localized (i18n) values (Including some double-byte character languages, such as Chinese).

Unwired WorkSpace supports various categories of datatypes.

Table 33. Datatype categories

Category	Description
Simple	int, string, date, and so on.
List of simple types	an array of simple types: int[], string[], date[], and so on.
Object (complex)	Implemented with a structure, and includes: <ul style="list-style-type: none"> • SAP input structure or input table • Nested complex types in Web Services that handle type structures as input (repeating and nested elements) <hr/> Note: Parameters, personalization keys, and default values all support complex types. Attributes do not, except for those that represent relationships.
List of object types	an array of objects: customer[], account[], and so on.

Table 34. Simple datatype description

Datatype	Description
binary, binary (%n)	<p>Select either:</p> <ul style="list-style-type: none"> • Input manually – enter a base64 encoded string directly in the input field. • Import from file – browse to a file from which the input string is retrieved. <p>If you are choosing a binary datatype, you can set the length if you require something other than the default set in the Miscellaneous Preferences dialog. To set the length, click the cell you require in the datatype column and select binary(%n). Press enter and then type the size you require. For example, you could:</p> <ol style="list-style-type: none"> 1. Click the particular cell in the datatype column, and select binary(%n) from the list of datatypes. 2. Enter the value for the binary length by highlighting %n with your cursor, and replacing it with the size you require. 3. Press enter to set the binary length. For example, if you entered 10 as the binary length, you see binary(10) in the datatype cell. <hr/> <p>Note: The maximum allowable length for binary datatypes is 2G bytes. If the MBO's attribute is a primary key, the maximum allowable length for binary datatypes is 2048 bytes. For MBO parameters, binary default value cannot exceed 16384 bytes.</p>
date	Select the day in the provided calendar. By default the local time zone is selected. The Time zone field is read only.
dateTime	By default the current date, time and time zone display in the calendar.
time	Enter the time, and enter the local time zone in the Time zone field.

Datatype	Description
string, string(%n)	<p>If you are choosing a string datatype, you can set the length of the string if you require something other than the default set in the Preferences dialog. To set the length, click the cell you require in the datatype column and select string(%n). Press enter and then type the size you require. For example, you could:</p> <ol style="list-style-type: none"> 1. Click the particular cell in the datatype column, and select string(%n) from the list of datatypes. 2. Enter the value for the string length by highlighting %n with your cursor, and replacing it with the size you require. 3. Press enter to set the string length. For example, if you entered 10 as the string length, you see string(10) in the datatype cell. <hr/> <p>Note: The maximum allowable length for string datatypes is 2G bytes. If the MBO's attribute is a primary key, the maximum allowable length is 512 bytes.</p>
All others	Enter the appropriate value in the Value field.

See also

- *Creating Attributes for a Mobile Business Object* on page 150
- *Creating Operations for a Mobile Business Object* on page 151
- *Creating the Mobile Business Object using the Mobile Business Object Icon* on page 80
- *Previewing Mobile Business Objects* on page 155
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a Synchronization tab from which you configure synchronization behavior.
- *Adding a Load Parameter to a Mobile Business Object* on page 177

Time Zone Datatype Behavior

Because enterprise resources are frequently located in different time zones, you need to understand the restrictions of using time-related datatypes when developing mobile business objects (MBOs).

Zone-offset independent field based time

Unwired Platform date, time, and dateTime datatypes hold time zone independent field-based time, as defined in *Incremental versus Field-Based Time*, making zone offsets invalid. For example, if you specify a default value for a synchronization parameter, it is not valid to include the zone offset:

```
2009-08-28T00:00:01+08:00
```

If a device application needs an attribute to store zone offset or zone name data, then define another MBO attribute to contain it. When previewing or testing date, time, and dateTime datatypes, values related to zone-dependent fields in Web services undergo the conversions described in this document. You need to account for any adjustments in the expected results, and may need to adjust any MBO default values you set.

Receiving values from Web services

Zone-offset behavior of date, time, and datetime datatypes:

- `xsd:date` – if Unwired Server receives a date value from a Web service that includes a zone offset (ending with "Z", "+XX:XX", or "-XX:XX"), Unwired Server ignores and drops the zone offset suffix. DATE values stored in the Unwired Server cache database (CDB), and sent to the client, do not include any zone offset. For example, if Unwired Server receives an `xsd:date` value:

```
2000-01-01+12:00
```

it converts that value to a DATE:

```
2000-01-01
```

- `xsd:time` – if Unwired Server receives a time value from a Web service that includes a zone offset (ending with "Z", "+XX:XX", or "-XX:XX"), Unwired Server ignores and drops the zone offset suffix. TIME values stored in the CDB, and sent to the client, do not include any zone offset. For example, if Unwired Server receives an `xsd:time` value:

```
14:00:00+12:00
```

it converts that value to a TIME:

```
14:00:00
```

- `xsd:dateTime` – if Unwired Server receives a dateTime value from a Web service that includes a zone offset (ending with "Z", "+XX:XX", or "-XX:XX"), Unwired Server adjusts the fields to convert the value to UTC (+00:00) and then drops the zone offset suffix. DATETIME values stored in the CDB, and sent to the client, do not include any zone offset. For example, if Unwired Server receives an `xsd:dateTime`:

```
2000-01-01T14:00:00+12:00
```

it converts that value to a DATETIME:

```
2000-01-01 02:00:00
```

If Unwired Server receives a value from a Web service that does not include a zone offset, then Unwired Server uses the unchanged value.

Sending values to Web services

By default, Unwired Server appends a "Z" to any date/time value that it sends to a Web service. If a Web service expects to receive zone-independent field based time, then use a derived simpleType with a pattern restriction in the XML schema description to indicate to the XML parser that only zoneless representation is accepted by the Web service. For example, you could define these simpleTypes:

```
<s:simpleType name="ZonelessDate">
  <s:restriction base="s:date">
```

```

        <s:pattern value="[0-9]{4}-[0-9]{2}-[0-9]{2}" />
    </s:restriction>
</s:simpleType>

<s:simpleType name="ZonelessTime">
    <s:restriction base="s:time">
        <s:pattern value="^[0-2][0-9]:[0-5][0-9]:[0-5][0-9]
            (.([0-9]{3}))?$" />
    </s:restriction>
</s:simpleType>

<s:simpleType name="ZonelessDateTime">
    <s:restriction base="s:dateTime">
        <s:pattern value="^[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-2]
            [0-9]:[0-5][0-9]:[0-5][0-9](.([0-9]{3}))?$" />
    </s:restriction>
</s:simpleType>

```

Then use the corresponding derived simpleType instead of using "xsd:date", "xsd:time", or "xsd:dateTime" as an element's type. These patterns are only examples and not "prescriptive."

Special consideration for client applications

As an example, you are writing a device application, and know that the enterprise information system (EIS) Web service always expects to receive values with zone offset. You also know (from above) that Unwired Server always sends a value with a "Z" suffix to the Web service. How do you then pass the appropriate values (for MBO attributes and/or operation parameters) from the device application?

In this example, the client device is located in New Zealand (12 hours ahead of UTC), and an event occurs at device-local date/time "2010-05-12T11:24:00+12:00". Since the client using the Object API can only pass zoneless values to MBO attributes or operation parameters, the client application must convert the fields to UTC, for example, "2010-05-11T23:24:00+00:00", and drop the zone offset to pass "2010-05-11T23:24:00" into an MBO attribute or operation parameter. When the client uploads this value to Unwired Server, it appends "Z" which results in "2010-05-11T23:24:00Z", which is then sent to the EIS Web service. Since "2010-05-11T23:24:00Z" is an equivalent point in time to "2010-05-12T11:24:00+12:00", no information is lost.

In other words, if the EIS expects to receive values with zone offsets, the client application might need to do zone offset conversions to UTC. Conversely, if the EIS expects to receive zoneless values, then the client application does not need to perform any conversions from device-local time, other than dropping the zone offset.

Load parameters and timezone support for Web service MBOs

Avoid using date/datetime datatypes as load parameters, since you could get unexpected results. If want to use Web service MBO operations that have time zone offsets, convert the date/datetime value to UTC, before sending it to Unwired Server. When the date/datetime value is returned from Unwired Server, change it back. Other considerations to be aware of:

- Date/Datetime datatype personalization keys for Web services should be avoided – if the client is in a different time zone, the same time change to UTC may be different, requiring a conversion to the personalization key when ever entering a different time zone.
- Default values for Date/Datetime datatypes in Web service operations with time zone offset should be avoided – the default values are set in the MBO at design time, the developer cannot determine which time zone the client uses, so the UTC conversion is impossible.

Datatype Default Values

This topic provides information about datatype default values that can be set for mobile business object (MBO) attributes and parameters.

You can provide a default value for attributes and parameters that are compatible with their datatype (and used by the device application to pass to the MBO), where ever you specify a datatype (Properties view, Preview dialog, and so on).

Note: When possible, the default value is retrieved with an appropriate value from the data source when you bind to the data source, which you can then modify.

NULL and empty default values

It is important to understand the differences between the default values NULL and no default (leaving the default value empty):

- NULL – not all datatypes support NULL as a value (int, decimal, double, float, and so on). After an MBO is created, NULL may be an available default value, but should not be selected if NULL is invalid for that datatype. If NULL is selected, and is invalid for the datatype, errors occur either when you deploy the MBO to Unwired Server, or when a device application interacts with the deployed MBO. These examples illustrate how a device application behaves when an MBO contains a synchronization parameter equal to NULL:
 - Where NULL is supported – the device application receives the rows where the attribute in the MBO is NULL. If a synchronization/load parameter is NULL, then data refresh is performed using the value NULL.
 - Where NULL is not supported – if associating synchronization parameter X with attribute X, the download cursor is similar to:


```
select ... from my_table t where t.last_modified >= ... and t.x
>= :X
```

If X is NULL, no rows are returned, since (t.x >= :X) is false in the database if X is NULL.
- empty default value – an empty string is not the same as no default. For string and binary datatypes, an empty string is a valid default value. For other datatypes, an empty string is invalid and generates an error.

The default value is set according to the nullability and datatype of the argument, synchronization parameter, or personalization key. For nullable types, the initial default value

is set to NULL, for non-nullable types, a valid value is set according to the datatype (for example, string "", boolean "false", decimal "0", integer "0", float "0", and so on).

The default datatype length, if you do not specify one, is:

- string – 300
- binary – 32767

Default values for Adaptive Server Anywhere uniqueidentifiers

Keep these points in mind when working with Adaptive Server Anywhere uniqueidentifiers:

1. Uniqueidentifiers are treated as binary datatypes in Unwired WorkSpace. If the enterprise information system (EIS) on which the uniqueidentifier resides:
 - Accepts NULL as a valid value, you can set the default value to NULL.
 - Does not accept NULL as a valid value, you need to set the correct default value. By default Unwired WorkSpace uses an empty string as the default value for binary datatypes.
2. To avoid entering values for the UNIQUEIDENTIFIER columns during insert operations, modify the default SQL definition as this example illustrates:
 - a. Create an Adaptive Server Anywhere table with SQL:

```
CREATE TABLE "dba"."YorkTable" (  
  "UniqueColumn" UNIQUEIDENTIFIER NOT NULL DEFAULT NEWID(),  
  "Characters" VARCHAR(10) NOT NULL,  
)  
IN SYSTEM  
;  
ALTER TABLE "dba"."YorkTable"  
ADD CONSTRAINT "ASA126" PRIMARY KEY NONCLUSTERED  
("UniqueColumn")  
;
```

- b. Drag-and-drop the table to create the new MBO.

The auto-generated Create operation SQL is:

```
"INSERT INTO sampledb.dba.YorkTable (UniqueColumn, Characters)  
VALUES  
(:UniqueColumn, :Characters)"
```

- c. To avoid entering values for the UNIQUEIDENTIFIER column during insert operations, change the SQL definition to:

```
"INSERT INTO sampledb.dba.YorkTable (Characters) VALUES  
(:Characters)"
```

Structure Objects

Structure objects (complex datatypes), represents an object datatype that includes attributes that define the datatype.

Creating Structure Objects using the Structure Icon

Structure objects hold complex object types, for example, an SAP input structure or input table. Structure objects do not bind to data sources or contain operations. If an MBO

references an structure object, a dashed line connects the two objects in the Mobile Application Diagram.

1. From the Mobile Application Diagram, select the Structure icon from the palette, then click an empty area of the diagram. The New Structure wizard displays.
2. Enter definition information and click **Next**:
 - Name – name of the structure object
 - Comment – an optional comment
3. From the Attributes page, define the simple or complex datatypes that define the structure by using the **Add**, **Delete**, **Delete All**, **Up**, and **Down** buttons, and click **Finish** when done.

The datatype of a structure attribute can include other structures (complex types).

Complex Datatypes

Structure objects hold complex object types (data structures), for example, an SAP input structure or input table.

When created, structure objects (or complex types) are generated into a class. The complex type contains one or more attributes. Every attribute contains type and name information.

Type	Valid element
Simple type	String Double Binary Integer Date DateTime Time Boolean Long Float Decimal Byte Short
List of simple types	An array of any of the supported simple types. For example: String[]

The name of the complex type attribute is used as the generated class name and should follow attribute naming conventions. The complex type is used in many places so naming is important for identification. For example, the complex type:

```
Address
  State
  City
  Street
```

could have this value, represented by this structure:

```
[State="Ca":City="Dublin":Street="Sybase Drive"]
```

Complex Datatype Limitations

Understand complex datatype (structures) restrictions and limitations.

Complex datatype default value limitations

- You can only set a default value for the complex datatype, not values for individual fields within the complex datatype.
- If multiple parameters refer to the same structure object, and individual default values are set for the parameters, Unwired WorkSpace passes only the first value to the structure and ignores the other values. To avoid this situation:
Instead of having multiple parameters reference the same MBO, copy and paste the structure object so that each reference is to a single structure object.

Complex datatype limitations

- Complex datatypes which are not bound to any MBO operation or component are not included in the generated code
- If you create a nested tree with three levels (structures) and you delete the last node in the tree (level three), no error message displays indicating that the attribute type of one of the attributes on level two is non-existent
- If there is a type mismatch between a synchronization parameter type and personalization key type, the error is not visible in the header area of the properties view.
- If you attempt to model a structure with a field/attribute of the same type you either get a `StackOverflowError` or if you try to model an attribute type as a list type of the same structure, the change is ignored.
- If you attempt to model two structures each with attributes of the other type (for example, structure5 contains an attribute of type structure6, and structure6 contains an attribute of type structure5) a `StackOverflowError` occurs.

Deleting structures

A structure can be referenced by a personalization key or a parameter or other structure's attribute. It can be deleted only if it's not referenced by any entity. If a personalization key references the structure, the deletion of the structure (either from object diagram or from workspace navigator) generates an error similar to:

```
"Structure type:'' < structure_name > '' can't be deleted,
because it is still referenced by personalization key : '' < PK_name
> ''".
```

A Cut operation in the object diagram or Delete operation generates a similar error message if it is still referenced by a parameter or other structure.

Editing Structure Type Default Values

Edit complex datatype (structured MBO) default values from the Structure Type Values editor, which provides a table view that includes an expandable tree that allows access to the structures and types.

The Structure Type Values editor allows you to edit structured MBOs from various places, including personalization key default values. You can:

- Set the default value of one attribute of a complex datatype to NULL
 - Set the default value of a complex datatype object to NULL
 - Add a new object to a list of complex datatypes and set the object's default value
1. Editing a structured type depends on the context from which you access the editor. For example:
 - Personalization keys – if creating a new personalization key that includes a structured datatype as the **Type**, select the ellipses (...) next to the **Default values** field. For existing personalization keys, right-click the personalization key and select **Properties**, and select the ellipses (...) next to the **Default values** field.
 - Attributes – from the Properties view, select the Attributes Mapping tab. If the attribute is a structured type, select the corresponding ellipses (...) in the **Default value** field.
 - Others – similar to attributes, if datatype supports structured types, and allows you to set default values, select the ellipses (...) in the **Default value** field.
 2. Make changes from the editor as needed.

The editor provides a table view that includes an expandable tree that allows access to the structured datatype. The table view includes these columns:

- Structure – the expandable root of the structure, which includes all structure members.
 - Type – the datatype of each of the structure members. A member can be a simple or structured type.
 - Values – the default value for each of the structure members. Modify changes for individual members, not at the root of the structure. Changes made to individual members are reflected in a comma separated list of the root.
3. Use the navigation buttons (**Up** and **Down**) to move structures. You cannot move individual members within a structure. Use the **Add**, **Delete**, and **Delete All** buttons to add or delete members or entire structures.
 4. Click **Ok** when finished.

Adding Structure Objects to Mobile Business Objects

Add complex object types to mobile business objects (MBOs) through structure objects.

When creating an MBO from a data source that contains complex object types, for example, multiple SAP output tables, Unwired WorkSpace generates a structure object and initially sets it as the datatype of both the argument and the derived parameter, which you can change from the drop-down list.

1. Create the MBO. For example, drag-and-drop an SAP data source onto the Mobile Application Diagram.
2. The MBO parameter to remote operation mappings appear in the Parameters page of the New Attributes wizard, including any generated structure objects.
3. Accept the default mappings and click **Finish**, or modify the parameter datatype by selecting something other than the default from the drop-down list:
 - For a structured argument, there is only one complex-typed mapping, instead of a number of simple-typed mappings.
 - In the Datatype drop-down, you can select from any of the available simple types, list of simple types, object types, or list of object types.

Editing List Type Default Values

Edit list (array) datatype default values from the List Values editor, which provides a table view that includes all members of the list.

The List Values editor allows you to edit array datatypes from various places, including personalization key default values. You can:

1. Editing an array datatype depends on the context from which you access the editor. For example:
 - Personalization keys – if creating a new personalization key that includes an array datatype as the **Type**, select the ellipses (...) next to the **Default values** field. For existing personalization keys, right-click the personalization key and select **Properties**, and select the ellipses (...) next to the **Default values** field.
 - Attributes – from the Properties view, select the Attributes Mapping tab. If the attribute is an array, select the corresponding ellipses (...) in the **Default value** field.
 - Others – similar to attributes, if datatype supports arrays, and allows you to set default values, select the ellipses (...) in the **Default value** field.
2. Make changes from the editor as needed.

The editor provides a Value column where you can define a value for array members, and use the **Add**, **Delete**, and **Delete all** buttons to create or delete array members as needed.

The type of array determines what values you can select. For example, if you add a Datetime type, select a default value from the calendar.

3. Click **Ok** when finished.

Unwired Platform to Enterprise Information System Datatype Mappings

These tables provide mapping information for various EIS types into Unwired Platform data types.

Table 35. JDBC types

MBO datatype	Generic JDBC type	Java type
BINARY	java.sql.Types.BINARY java.sql.Types.VARBINARY java.sql.Types.LONGVARBINARY java.sql.Types.BLOB	java.lang.Byte[]
BOOLEAN	java.sql.Types.BOOLEAN java.sql.Types.BIT	java.lang.Boolean
BYTE	java.sql.Types.BOOLEAN java.sql.Types.BIT	java.lang.Byte
CHAR	java.sql.Types.BOOLEAN java.sql.Types.BIT	java.lang.Character
DATE	java.sql.Types.DATE	java.util.Calendar
DATETIME	java.sql.Types.TIMESTAMP	java.util.Calendar
DECIMAL	java.sql.Types.DECIMAL java.sql.Types.NUMERIC	java.math.BigDecimal
DOUBLE	java.sql.Types.DOUBLE	java.lang.Double
FLOAT	java.sql.Types.FLOAT java.sql.Types.REAL	java.lang.Float
INT	java.sql.Types.INTEGER	java.math.BigInteger
INTEGER	java.sql.Types.INTEGER	java.math.BigInteger
LONG	java.sql.Types.BIGINT	java.lang.Long
SHORT	java.sql.Types.BIGINT	java.lang.Short

MBO datatype	Generic JDBC type	Java type
STRING	java.sql.Types.CHAR java.sql.Types.NCHAR java.sql.Types.VARCHAR java.sql.Types.NVARCHAR java.sql.Types.LONGVARCHAR java.sql.Types.LONGNVARCHAR	java.lang.String
TIME	java.sql.Types.TIME	java.lang.String

Table 36. Web service types

MBO datatype	XSD type	Java type
BOOLEAN	xs:Boolean	java.lang.Boolean
BYTE	xs:Byte	java.lang.Byte
BINARY	xs:Base64Binary xs:HexBinary	java.lang.Byte[]
DOUBLE	xs:Double	java.lang.Double
FLOAT	xs:Float xs:Int	java.lang.Float
CHAR	xs:UnsignedShort	java.lang.Character
LONG	xs:Long xs:UnsignedInt	java.lang.Long
SHORT	xs:Short xs:UnsignedByte	java.lang.Short

MBO datatype	XSD type	Java type
STRING	xs:String xs:Duration xs:GYearMonth xs:GYear xs:GMonthDay xs:GDay xs:GMonth xs:NOTATION xs:Token xs:NormalizedString xs:Language xs:Name xs:NMTOKEN xs:NCName xs:ID xs:IDREF xs:ENTITY xs:NMTOKENS xs:IDREFS xs:ENTITIES	java.lang.String
DECIMAL	xs:Decimal	java.math.BigDecimal
INT	xs:Integer xs:NonPositiveInteger xs:NonNegativeInteger xs:NegativeInteger xs:UnsignedLong xs:PositiveInteger xs:AnyURI (java.net.URI.class)	java.math.BigInteger

MBO datatype	XSD type	Java type
INTEGER	xs:Integer xs:NonPositiveInteger xs:NonNegativeInteger xs:NegativeInteger xs:UnsignedLong xs:PositiveInteger xs:AnyURI (java.net.URI.class)	java.math.BigInteger
DATETIME	xs:DateTime	java.util.Calendar
TIME	xs:Time	java.lang.String
DATE	xs>Date xs:QName	java.util.Calendar

Table 37. SAP RFC types

MBO data-type	JCo version 3 type code	ABAP type	Java type
BINARY	JCoMetaDa- ta.TYPE_BYTE JCoMetaDa- ta.TYPE_XSTRING	X Y	java.lang.Byte[]
BOOLEAN	JCoMetaDa- ta.TYPE_BYTE JCo- MetaDa- ta.TYPE_XSTRING	X Y	java.lang.Boolean
BYTE	JCoMetaDa- ta.TYPE_INT1	b	java.lang.Byte
CHAR	JCoMetaDa- ta.TYPE_CHAR	C	java.lang.Character
DATE	JCoMetaDa- ta.TYPE_DATE	D	java.util.Calendar
DATETIME	JCoMetaDa- ta.TYPE_DATE	D	java.util.Calendar

MBO data-type	JCo version 3 type code	ABAP type	Java type
DECIMAL	JCoMetaDa- ta.TYPE_BCD	P	java.math.BigDecimal
DOUBLE	JCoMetaDa- ta.TYPE_FLOAT	F	java.lang.Double
FLOAT	JCoMetaDa- ta.TYPE_FLOAT	F	java.lang.Float
INT	JCoMetaDa- ta.TYPE_INT	I r	java.lang.Integer
INTEGER	JCoMetaDa- ta.TYPE_INT	I	java.math.BigInteger
LONG	JCoMetaDa- ta.TYPE_INT	I	java.lang.Long
SHORT	JCoMetaDa- ta.TYPE_INT2	s	java.lang.Short
STRING	JCoMetaDa- ta.TYPE_STRING TYPE_NUM TYPE_FLOAT TYPE_DECF34 TYPE_DECF16	g N F e a	java.lang.String
TIME	JCoMetaDa- ta.TYPE_TIME	T	java.lang.String

Mobile Business Object to Mobile Device Platform Datatype Mappings

This table provides mapping information for various MBO datatypes to those of the mobile device target language.

The optional "?" suffix indicates the datatype supports nullability. In some cases, a nullable target language type might be used for a non-nullable MBO type. In either case, the nullability indicator should always be specified if the target type must support nulls.

Any referenced type name that does not appear in the table is expected to be one of the following:

- The name of a class defined within the same package.
- The fully qualified name of a class defined in a previously compiled package.

Develop

- The name of an imported class.
- The name of an external class.

Table 38. MBO to Java RIM datatype mappings

MBO datatype	Java RIM types
BOOLEAN	boolean
BOOLEAN?	java.lang.Boolean
STRING	java.lang.String
STRING?	java.lang.String
BINARY	byte[]
BINARY?	byte[]
CHAR	char
CHAR?	java.lang.Character
BYTE	byte
BYTE?	java.lang.Byte
SHORT	short
SHORT?	java.lang.Short
INT	int
INT?	java.lang.Integer
LONG	long
LONG?	java.lang.Long
INTEGER	java.math.BigInteger (javamx.math.BigInteger)
INTEGER?	java.math.BigInteger (javamx.math.BigInteger)
DECIMAL	java.math.BigDecimal (javamx.math.BigDecimal)
DECIMAL?	java.math.BigDecimal (javamx.math.BigDecimal)
FLOAT	float
FLOAT?	java.lang.Float
DOUBLE	double
DOUBLE?	java.lang.Double

MBO datatype	Java RIM types
DATE	java.util.Date
DATE?	java.util.Date
TIME	java.util.Date
TIME?	java.util.Date
DATETIME	java.util.Date
DATETIME?	java.util.Date

Table 39. MBO to C# device datatype mappings

MBO datatype	C# type
BOOLEAN	bool
BOOLEAN?	bool?
STRING	string
STRING?	string
BINARY	byte[]
BINARY?	byte[]
CHAR	char
CHAR?	char?
BYTE	byte
BYTE?	byte?
SHORT	short
SHORT?	short?
INT	int
INT?	int?
LONG	long
LONG?	long?
INTEGER	decimal
INTEGER?	decimal?

MBO datatype	C# type
DECIMAL	decimal
DECIMAL?	decimal?
FLOAT	float
FLOAT?	float?
DOUBLE	double
DOUBLE?	double?
DATE	System.DateTime
DATE?	System.DateTime?
TIME	System.DateTime
TIME?	System.DateTime?
DATETIME	System.DateTime
DATETIME?	System.DateTime?

Table 40. MBO to VB.NET datatype mappings

MBO datatype	VB.NET type
BOOLEAN	boolean
BOOLEAN?	nullable(of boolean)
STRING	string
STRING?	string
BINARY	byte()
BINARY?	byte()
CHAR	char
CHAR?	nullable(of char)
BYTE	byte
BYTE?	nullable(of byte)
SHORT	short
SHORT?	nullable(of short)

MBO datatype	VB.NET type
INT	integer
INT?	nullable(of integer)
LONG	long
LONG?	nullable(of long)
INTEGER	decimal
INTEGER?	nullable(of decimal)
DECIMAL	decimal
DECIMAL?	nullable(of decimal)
FLOAT	single
FLOAT?	nullable(of single)
DOUBLE	double
DOUBLE?	nullable(of double)
DATE	date
DATE?	nullable(of date)
TIME	date
TIME?	nullable(of date)
DATETIME	date
DATETIME?	nullable(of date)

Mobile Business Object Properties

Modify mobile business object properties from the Properties view.

Table 41. Mobile business object properties

Tab	Contents
General	<ul style="list-style-type: none">• Name – the name of the mobile business object (MBO).• Comment – describes the mobile business object.• Generate metadata options – select to generate the metadata class for the corresponding MBO attributes and operations. This option is unselected by default. Generate metadata when you want to further customize generated MBO code. There are also Generate metadata options in the Code Generation wizard. If selected, those options override the MBO level options.

Tab	Contents
Attributes	<p>If the mobile business object attributes are bound to a data source, the contents of the Attributes tabs include:</p> <ul style="list-style-type: none"> • Data Source – information about the data source from which the attributes are derived and bound. Select Change Connection Profile to change the connection profile of the same data source type, or Bind Data Source to bind or rebind the mobile business object to a data source of any supported type. • Definition – displays the defined attributes and any runtime credential/authentication requirements, which vary depending on the data source type. Select Edit to launch the Change Definition dialog and modify the definition. You can also create a resultset filter skeleton or add an existing resultset filter class. Select Preview to preview the mobile business object. <p>Result Filters shows resultset filters and their paths.</p> <ul style="list-style-type: none"> • Attributes mapping – Includes: <ul style="list-style-type: none"> • Attributes <ul style="list-style-type: none"> • Name – name of the attribute. • Datatype – attribute datatype • Nullable – a Null value for the attribute is valid. • Primary key – analogous to a table's primary key, when set, MBO data can be searched/found using the attribute's value. <p>When set, generated device client code contains FindBy methods (along with associated parameters and return types). At runtime, FindBy methods return a collection of objects that match the specified search criteria defined in the object query.</p> <hr/> <p>Note: The Unwired Server cache database (CDB) does support identical rows. If a SQL query statement generates identical rows, the CDB identifies and stores one unique row, resulting in only one row displaying on the device after synchronization.</p> <hr/> • Data Source <ul style="list-style-type: none"> • Map to – data source column to which the corresponding attribute is mapped. • Datatype – data source column datatype to which the corresponding attribute is mapped. • Nullable – select this option only for enterprise information system (EIS) arguments that support NULL as a valid value. <hr/> <p>Note: For SAP and Web service data sources, the Nullable and Datatype columns are read-only. You cannot change them, instead, Unwired WorkSpace correctly identifies and sets them from the back-end data source. But for a JDBC data source, Unwired WorkSpace may not be able to correctly identify the Datatype and nullability, so you need data source knowledge to correct them if needed.</p> <hr/>

Tab	Contents
	<ul style="list-style-type: none"> • Click Refresh to refresh the metadata (parameters and columns). Use the Add, Delete, Delete All buttons to remove attributes. • Click Remap to automatically generate new mappings for unmap-ped attribute columns based on metadata changes of the data source to which the mobile business object (MBO) is bound. • Show figure button – when selected, the visual display of the at-tribute to tabular view column mapping displays. • Load Parameters – see <i>Adding a Load Parameter</i> for details. • Roles – lists all logical roles assigned to the mobile business object along with all available roles. Use Add and Add All to move available roles to assigned roles, and Remove and Remove All to remove roles from a mobile business object (or double-click a role to add or remove it). Select Create to define a new role. • Object Queries – displays the auto-generated object queries based on which attributes are primary key attributes. Also allows you to add, edit, or delete object queries. See <i>Object Queries</i> for details.
Operations	<p>Lists all operations defined for the mobile business object. Select Add to define a new operation, highlight an existing operation and select Edit to modify an existing operation, or use Delete and Delete All to remove operations.</p> <p>Select Bind to bind an operation to a data source, or Rebind to change an existing binding to a different data source. see <i>Mobile Business Object Operation Properties</i> for details.</p>
Relationships	<p>Lists all relationships defined between this and other mobile business ob-jects. Select Add to define a new relationship, highlight an existing rela-tionship and select Edit to modify an existing relationship, or use Delete and Delete All to remove relationships.</p>
Synchronization	<p>Define various synchronization aspects of the MBO from the Advanced Developer profile. See <i>Defining Synchronization Properties</i>.</p>
Appearance	<p>Modify certain look-and-feel aspects of the mobile business object from the Advanced Developer profile.</p>

See also

- *Creating Attributes for a Mobile Business Object* on page 150
- *Creating Operations for a Mobile Business Object* on page 151
- *Creating the Mobile Business Object using the Mobile Business Object Icon* on page 80
- *Previewing Mobile Business Objects* on page 155
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a Synchronization tab from which you configure synchronization behavior.

- *Adding a Load Parameter to a Mobile Business Object* on page 177

Old Value Argument

Use the Old Value Argument field to map a mobile business object(MBO) parameter to a second (old) argument in update operations.

Use the **Old value argument** field to change an Update operation's parameter from its current value to a new value, while still allowing access to the parameter through the previously defined argument. For example, if you use the Update operation of an MBO to modify the lname parameter's argument from Jones to Smith, the **Argument** value is Smith, and if you choose to, you can map the **Old value argument** to Jones.

The old value is available from a drop down list, in the form *old.argument_name* (where *argument_name* is the name of the original argument, lname in the above example. The Un-map option unmaps the old value.

The **Old value argument** field is available from a number of Properties view locations, including:

- Parameters tab of an Update operation's Property view.
- Update operation Editing dialog.
- Test execute dialog supports selection of *old.argument* parameters when testing Update operations.

Note: Although the parameter can be mapped to both **Argument** and **Old value argument**, when you select **Show Figure**, only the connection to the **Argument** displays.

Avoiding synchronization conflicts with the old value argument

If a mobile business object (MBO) performs an update operation, the device sends additional parameters to the server that contain the original values of the database columns mapped to the object's parameters. These original values are shared with the enterprise information system (EIS) server in specially-named arguments. That is, if an argument is named A, and if the original value is available, it is provided in the argument named `old.A`. By checking whether or not the original values has become stale, the EIS update operation can avoid conflicting updates.

Consider a database that contains table Person with three columns: `socialsecurity_num (pk)`, `fname`, and `lname`. Then consider what happens in case of updates by different device applications.

1. Two devices, D1 and D2, have downloaded a row 999-55-1212, 'Joe', 'User'.
2. D1 updates the `fname` to 'Jane' and succeeds.
3. D2 updates `lname` to "Yooser", and consequently supplies the original values 'Joe' and 'User'.

Because the current `lname ('Jane')` is not same as `old.fname ('Joe')` the update for D2 does not occur.

See also

- *Creating Attributes for a Mobile Business Object* on page 150
- *Creating Operations for a Mobile Business Object* on page 151
- *Creating the Mobile Business Object using the Mobile Business Object Icon* on page 80
- *Previewing Mobile Business Objects* on page 155
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a Synchronization tab from which you configure synchronization behavior.
- *Adding a Load Parameter to a Mobile Business Object* on page 177

Creating Attributes for a Mobile Business Object

Use the Mobile Business Object Creation wizard to create a mobile business object and launch the Attributes Creation wizard. A mobile business object may or may not contain attributes.

1. When you create a mobile business object in the Mobile Application Diagram, the Attributes Creation wizard automatically starts. You can also add an attribute to an existing mobile business object by selecting the Attribute icon from the palette and clicking inside the attribute section of an existing mobile business object.
2. Follow the wizard instructions to create attributes for the mobile business object. Attributes vary, depending on the data source to which you are binding.
3. Select either:

Option	Description
Specify a data source	Specify the data source to which the mobile business object attributes map. This creates a default mapping depending on the data source to which you are mapping. You must have a connection profile associated with the data source to which you are binding to use this option.
Bind data source later	Manually define the attributes and role assignments without binding to the data source.

See also

- *Mobile Business Object Properties* on page 146
- *Mobile Business Object Attribute Properties* on page 151
- *Mobile Business Object Operation Properties* on page 153
- *Datatype Support* on page 125
- *Old Value Argument* on page 149

Mobile Business Object Attribute Properties

Modify a mobile business object that contains a single attribute using the Properties view.

Table 42. Single attribute properties

Tab	Contents
General	<ul style="list-style-type: none"> • Name – the name of the attribute. You cannot use keyword or reserved words as the attribute name. • Datatype – the attribute's datatype. • Nullable – select this option to allow null as a valid value. This option may not be available if the datatype does not support null. • Map to – the data source to which the attribute maps. • Primary key – identifies the data source column as a primary key.

See also

- *Creating Attributes for a Mobile Business Object* on page 150
- *Creating Operations for a Mobile Business Object* on page 151
- *Creating the Mobile Business Object using the Mobile Business Object Icon* on page 80
- *Previewing Mobile Business Objects* on page 155
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a Synchronization tab from which you configure synchronization behavior.
- *Adding a Load Parameter to a Mobile Business Object* on page 177
- *Load Parameters* on page 176
- *Combining Load and Synchronization Parameters* on page 180

Creating Operations for a Mobile Business Object

Use the Operation Creation wizard to create an operation and add it to the mobile business object.

Prerequisites

You must first create a mobile business object before adding an operation, unless you create the mobile business object directly from the data source, in which case you can drag-and-drop the data source on to the Mobile Application Diagram which launches the **Quick Create** wizard for automatic creation of attributes and operations.

Task

1. Launch the **New Operation** wizard from the Mobile Application Diagram by selecting the **Operation** icon from the palette, then selecting the operation section of the mobile business object to which you are adding the operation.
2. Follow the wizard instructions to add an operation to the mobile business object. Include the operation type that allows you to Create, Update, or Delete data on the back-end data source, or Other for other types (or undefined) operations.

Note: The New Operation wizard varies depending on the type of data source to which the operation is bound. See the corresponding binding topic for detailed information.

3. Select either:

Option	Description
Specify a data source	Specify the data source to which the mobile business object operation maps. This creates a default mapping depending on the data source to which you are mapping. You must have access to the data source (for example, a data source connection profile) to which you are binding to use this option.
Bind data source later	Manually define the operation, parameters, and role assignments without binding to the data source.

See also

- *Creating Attributes for a Mobile Business Object* Use the Mobile Business Object Creation wizard to create a mobile business object and launch the Attributes Creation wizard. A mobile business object may or may not contain attributes.
- *Binding Mobile Business Objects to Data Sources* on page 85
- *Mobile Business Object Properties* on page 146
- *Mobile Business Object Attribute Properties* on page 151
- *Mobile Business Object Operation Properties* on page 153
- *Datatype Support* on page 125
- *Old Value Argument* on page 149

Mobile Business Object Operation Properties

Modify mobile business object operation properties using the Properties view.

Table 43. Operation properties

Tab	Contents
General	<ul style="list-style-type: none"> • Operation name – the name of the operation. • Operation type – the operation to be performed. Operation types include Create, Update, Delete, and Other. Use Other only when there are no attributes associated with the MBO (disallowing the Fill from Attribute field to determine the operation's parameters). • Comment – describes the operation.
Data Source	<p>Information about the data source from which the operation is derived. Select Change Connection Profile to bind the operation to a different connection profile of the same data source type, or Bind Data Source to bind the operation to a data source, or to rebind to a data source.</p>
Definition	<p>View, modify, and test operations:</p> <ul style="list-style-type: none"> • View – display the operation (SQL statement, or Web service method, for example) in a read-only window. • Edit – modify the operation definition. You can change the type of operation as long as it is supported by the data source, and validate your changes. Enter credentials, if required, to access the data source. • Test execute – tests the operation against the data source to which it is bound. The Test execute dialog allows you to load any existing test configurations and preview the results. <hr/> <p>Note: Selecting Test execute can modify data stored on the data source to which the operation is bound. However, SQL statements are automatically rolled back and do not modify data.</p>

Tab	Contents
<p>Parameters</p>	<p>View or edit the operation parameters that map to data source arguments. Operation parameters determine how the client passes information to the enterprise information system (EIS). Select Refresh to update the parameters.</p> <p>Click Remap to automatically generate new mappings for unmapped parameter arguments based on metadata changes of the data source to which the mobile business object (MBO) is bound.</p> <p>Select:</p> <ul style="list-style-type: none"> • Parameters <ul style="list-style-type: none"> • Name – name of the mobile business object parameter. Names cannot contain C# or Java reserved words. • Datatype – the parameter's datatype • Nullable – accepts null as a valid value. Unselect this option if the argument to which this parameter is mapped does not support null as a valid value. • Updatable – allows the parameter to be updated on the device. • Required – indicates a required parameter. • Personalization key – select a personalization key to map to the parameter, which provides the value. • Fill from attribute – fills the parameter's value with that of the selected attribute. <p>If a create, update, or delete (CUD) operation parameter is a fill from attribute parameter and is bound to a personalization key, you cannot set <NULL> value for this parameter. If you want to pass <NULL> value for a CUD operation parameter which is fill-from-attribute, model additional operations without personalization keys or default values bound to the parameters.</p> <ul style="list-style-type: none"> • Old value argument (update operations only) – maps a parameter to a second (old) argument. See <i>Old Value Argument</i>. • Data Source <ul style="list-style-type: none"> • Argument – data source argument name to which the parameter is mapped. • Datatype – Datatype of the data source argument to which the parameter is mapped. • Nullable – accepts null as a valid value. • Default value – the default value for the argument, or select <NULL> from the drop-down list. • Show Figure – display a graphical representation of the operation's parameter mappings. Modify the mapping by clicking and moving the arrow that identifies the mapping.

Tab	Contents
Roles	Lists all logical roles granted to the operation along with all available roles. Use Add and Add All to move available roles to granted, and Remove and Remove All to remove roles from an operation. Select Create to define a new role.
Cache Update Policy	Determine how the results of an MBO operation are applied to the Unwired Server cache. See <i>Cache Update Policy</i> .

See also

- *Creating Attributes for a Mobile Business Object* on page 150
- *Creating Operations for a Mobile Business Object* on page 151
- *Creating the Mobile Business Object using the Mobile Business Object Icon* on page 80
- *Previewing Mobile Business Objects* on page 155
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a *Synchronization* tab from which you configure synchronization behavior.
- *Adding a Load Parameter to a Mobile Business Object* on page 177

Previewing Mobile Business Objects

Preview mobile business object operations and attributes against the data source to which they are bound Using the Preview and Test Execute dialogs.

See also

- *Modifying Mobile Business Object Properties* on page 119
- *Mobile Business Object Properties* on page 146
- *Mobile Business Object Attribute Properties* on page 151
- *Mobile Business Object Operation Properties* on page 153
- *Datatype Support* on page 125
- *Old Value Argument* on page 149

Previewing Mobile Business Object Attributes

Configure attribute parameters and preview the results against the data source to which they are bound. Optionally create a reusable configuration for future previews.

1. From the Mobile Application Diagram, select the mobile business object you want to test by clicking once anywhere in the mobile business object.
2. Right-click the mobile business object and select **Preview**.

Note: You can also access the Preview dialog when defining attributes for a newly created mobile business object. Or you can click the Definition tab of Attributes in the Properties view then click the **Preview** button to see the Preview dialog.

- From the Preview dialog, configure the attribute parameters used to test and preview the mobile business object and preview the results.

Table 44. Preview dialog instructions

Field	Description
Attributes	<ul style="list-style-type: none"> • Argument name – (read-only) the name of the argument. • Datatype – (read-only) the parameter's datatype. • Nullable – (read-only) accepts NULL if selected. • Required – (read-only) requires a value if selected. • Value – supply a value for the attribute's parameter compatible with the datatype, to pass to the data source. To select NULL as the value, click in the field next to the parameter, then click the down arrow on the right to select NULL as the value from the drop-down. If the datatype is binary, date, datetime, or time, select the input dialog in which you can select a value. <p>If the datatype is TABLE, click in the Value field, then click the ellipsis (...) within the Value field to launch the Input Table Data dialog. The Input Table Data dialog contains columns that correspond to that of the selected parameter.</p> <p>the parameter displays only if you have defined it, otherwise the table is empty.</p> <p>Click Add to add a row and click inside each column of the new row to enter preview data that corresponds to the columns datatype. If you have multiple rows, you can order them using Up and Down buttons, and delete rows using the Delete and Delete All buttons.</p> <p>You can order the attribute parameters by selecting the column heading by which you want to order.</p>
Existing Configurations	<p>Select either an existing configuration that contains attribute parameter value settings, or preview without using a configuration:</p> <ul style="list-style-type: none"> • If you continue without a configuration, then the preview uses the parameter data you provided. Optionally, provide a configuration name then click Save to save your settings as a configuration. If you enter a name without clicking Save, the configuration is not created. • Select an existing configuration from which you preview the attributes. • Delete – deletes the selected configuration. • Delete all – deletes all saved configurations. • Load default – loads the default configuration.

Field	Description
Preview	<p>Select Preview to pass the parameters to the data source and view the results in the Preview Result screen.</p> <hr/> <p>Note:</p> <ul style="list-style-type: none"> • If the MBO definition requires a default value, and you do not provide one, the data source returns an error which Unwired WorkSpace displays, when you select Preview. • By default, 20 rows display at a time. You can change the default in the Miscellaneous Preferences dialog (Maximum rows to retrieve). But if this number is too high, the preview may take a long time to complete.

See also

- *Result Set Filters* on page 211

Testing Mobile Business Object Operations

Configure operation parameters and test them against the data source to which they are bound. Optionally create a reusable configuration for future testing.

1. From the Mobile Application Diagram, select the operation you want to test.
2. Right-click the operation and select **Test Execute**.

Note: You can also access the Test Execute dialog when creating mobile business object operations.

3. From the Test Execute dialog, configure the operation parameters used to test the operation.

Table 45. Test Execute dialog instructions

Field	Description
Operation parameters	<ul style="list-style-type: none"> • Argument name – (read-only) the name of the data source argument. • Datatype – (read-only) the parameter's datatype. • Nullable – (read-only) accepts NULL if selected. • Required – (read-only) requires a value if selected. • value – supply a value for the attribute's parameter compatible with the datatype, to pass to the data source. To select NULL as the value, click in the field next to the parameter, then click the down arrow on the right to select NULL as the value from the drop-down. If the datatype is binary, date, datetime, or time, select the input dialog in which you can select a value. <p>You can order the operation parameters by selecting the column heading by which you want to order.</p>

Field	Description
Existing Configurations	<p>Select either an existing configuration that contains test settings, or test without using a configuration:</p> <ul style="list-style-type: none"> • If you continue without a configuration, then the preview uses the parameter data you provided. Optionally provide a configuration name and click Save to save your settings as a configuration. If you enter a name without clicking Save, the configuration is not created. • Select an existing configuration from which you test the operation. • Delete – deletes the selected configuration. • Delete all – deletes all saved configurations. • Load default – loads the default value for each parameter.
Test Execute	<p>Select Test Execute to pass the parameters to the data source and view the results in the Preview Result screen. A warning message appears. Click OK to continue.</p>

Creating Relationships Between Mobile Business Objects

Use the New Relationship wizard to create relationships between two or more mobile business objects.

Prerequisites

At least two mobile business objects and at least one attribute must be defined for either mobile business object before you can create a relationship between them. You can also create relationships between MBOs through parameters.

Task

1. Launch the New Relationship wizard from the Mobile Application Diagram by selecting the Relationship icon from the palette, then make a connection between two mobile business objects by clicking in the name or attribute section of one mobile business object (parent/source), and then dragging the line to the other (child/target).
 You cannot create relationships between local business objects and MBOs bound to a data source.
2. Follow the wizard instructions to define a relationship between the mobile business objects. Include:
 - **Relationship name** – if necessary, change the default name to a unique, descriptive name.
 - **Source object** – read-only and automatically filled in. Enter the source object **Attribute** name, which identifies the name of the source of the relationship, and is independent of any MBO attribute.

- **Target object** – select a target of the relationship from the drop-down list. Enter the target object **Attribute** name, which identifies the name of the target of the relationship, and is independent of any MBO attribute.
- **Comment** – (Optional)
- **Relationship type** – either:
 - **One to many** – the source for this relationship is one-to-many (default). For example, one manager manages multiple employees.



- **One to one** – a one-to-one relationship between the target and the source. For example, a manager manages only one department.



- **Many to one** – the source of this relationship is many-to-one. For example, many employees work in one department.



- **Bi-directional** – indicates a two-way relationship. That is, changes can be propagated in either direction.



- **Composite** – create, update and delete operations on a parent MBO automatically cascades changes to the child entity. For example, deleting the parent Customer MBO cascades directly to the child MBO Sales Order. This option is disabled for many-to-one relationships.
3. Create the mappings from the source mobile business object to the target by selecting a source object attribute, parameter, or structured parameter and dragging it to the corresponding target to which you want to map. Structured parameter relationships are not true relationships, but allows mapping of structures. Unwired WorkSpace also supports attribute-to-parameter, parameter-to-attribute, and parameter-to-parameter mappings.
 4. Click **OK** when finished.

Relationships are identified in the Mobile Application Diagram by a line connecting the related MBOs, with a different type of arrow for each type of relationship (one-to-one, one-to-many, or many-to-one).

See also

- *Drag and Drop the Data Source onto the Mobile Application Diagram* Launch the Quick Create wizard that creates your mobile business object by dragging and dropping the data source onto the Mobile Application Diagram.
- *Modifying Mobile Business Object Properties* on page 119

Mobile Business Object Relationship Properties

Modify the relationship of two mobile business objects using the Properties view.

Table 46. Relationship properties

Tab	Contents
<p>General</p>	<ul style="list-style-type: none"> • Source object – the source MBO. • Source attribute – identifies the name of the source of the relationship, and is independent of any MBO attribute. • Target object – the target MBO. • Target attribute – identifies the name of the target of the relationship, and is independent of any MBO attribute. • Comment – a description of the relationship. • One-to-many, one-to-one, or many-to-one – select the type of relationship. By default, the relationship is one-to-many. • Composite – create, update and delete operations on a parent MBO automatically cascade changes to the child entity. For example, deleting the parent Customer MBO cascades directly to the child MBO Sales Order. This option is disabled for many-to-one relationships. • Bi-directional – indicates a two-way relationship and optional for all relationships, which enables selection of unidirectional, one-to-many, and many-to-one relationships.
<p>Mapping</p>	<p>View or change the mappings for this relationship. While attribute-to-attribute mapping is explicitly supported, you can also map:</p> <ul style="list-style-type: none"> • Parameter-to-parameter mapping, including structured parameters (structured MBOs) – requires that you set a propagate to attribute for the parameter, then map the parameter from the relationship mapping tab. For example, the Customer MBO has an attribute named state and a parameter named state_name, from the attributes Load Parameters tab, select state as the propagate to attribute for the state_name parameter. Then map the state_name parameter in a relationship. You must map sub-parameters when mapping structured parameters. For example, if Customer[] is the structured parameter, and it contains state as a sub-parameter, you must map the state sub-parameter, and not the Customer[] parameter. • Attribute-to-parameter and parameter-to-attribute mappings are also supported.

Tab	Contents
Appearance	<p>Modify the appearance of the text and line that represents the relationship. For example:</p> <ul style="list-style-type: none">• Fonts and Colors – changes the appearance of the relationship text displayed in the Mobile Application Diagram.• Routing – changes the routing of the relationship connector line displayed in the Mobile Application Diagram.• Line and arrows – changes the style of lines and arrows.• Smoothness – changes the smoothness of the relationship line.• Jump links – changes the placement and appearance of any jump links.

Relationship Guidelines

Follow these guidelines when configuring MBOs and enterprise information systems (EISs) used in relationships.

Table 47. Relationship guidelines

Guideline	Example
<p>These examples use SQL Anywhere to illustrate how to properly configure the EIS so delete operations behave as expected</p>	<ul style="list-style-type: none"> Example one consist of two MBOs: <pre>Address: addr_id, street, city, country</pre> <pre>Employee: emp_id, name, addr_id (foreign key references Address.addr_id)</pre> <p>The MBOs have a relationship defined, for example, a composite, uni-directional, one-to-one relationship: <pre>Employee(Parent) --> Address(Child)</pre> <p>At runtime, if you delete Employee, Unwired Server attempts to delete the child Address first, which fails because of the Employee.addr_id-to-Address.addr_id reference. For the delete operation to perform as expected, the addr_id foreign key must be defined in the EIS database as on delete set null to enable deletion.</p> <p>This requirement applies regardless of the type of relationship defined between the MBOs.</p> </p> Example two also uses the Employee and Address MBOs in a relationship that has mutual references: <pre>Address: addr_id, street, city, country, emp_id(foreign key references Employee.emp_id)</pre> <pre>Employee: emp_id, name, addr_id(foreign key references Address.addr_id)</pre> <p>You must set both foreign keys in the EIS database as on delete set null or on delete cascade, regardless of the type of relationship you define for Address and Employee, or neither of them can be deleted.</p>
<p>If the foreign key in a relationship points to a <code>char(n)</code> column, and it is the only column in the table, use the rtrim(column name) function in the MBOs column definition</p>	<p>This example has a relationship defined as: <pre>States(Parent) --> Sales_region(Child)</pre> <p>Sales_region contains one column named Region with a data type of char(7). The MBO definition should be: <pre>SELECT rtrim(region) as region FROM sampledb.dba.sales_regions</pre></p> </p>

Guideline	Example
MBOs used in a relationship must belong to the same Synchronization group	MBOs in a relationship need to share the same synchronization characteristics for proper synchronization of all MBOs within the relationship, therefore verify that all MBOs are in the same synchronization group.
By default, all MBOs in a relationship are assigned to the same cache group	You can assign different cache groups to MBOs within a relationship, however, doing so generates a warning message. You may want to, for example, separate cache groups with different cache policies for a <code>sales</code> MBO, which changes frequently, that has a relationship with the <code>catalog</code> MBO, which changes infrequently.
Unwired WorkSpace does not support Circular relationships	<p>Do not create circular relationships that work from parent-to-child then, eventually, back to the parent. For example:</p> <p>A relationship that includes three MBOs: A, B, and C, where</p> <pre>A----->B----->C----->A</pre> <p>is not supported.</p>

Guideline	Example
<p>Composite relationship behavior</p>	<ul style="list-style-type: none"> • If a parent MBO's primary key is auto-incremental and you want to perform multi-level insert operations, then the relationship must be composite. Otherwise the child MBO can only be created on an existing parent MBO. • When Composite is selected, all child operations are performed through the parent. • Foreign key attributes are updated automatically when setting a related parent/child object. Do not directly update the child's attribute (row) identified by the foreign key. • Multi-level insert (MLI) operations require composite relationships. • For successful composite deletes, all children MBOs in the composite relationship must have a delete operation, which includes only one parameter that has Fill from Attribute to the primary key attribute and mapped to the primary key argument. • A child MBO can have multiple parents (many to one), but only one parent can have a composite relationship to the child. <p>In a composite relationship, when one child MBO does not have a synchronization parameter, and has multiple parents, each of which has a synchronization parameter, the child inherits synchronization behavior from both of the parents. If a composite relationship does not exist, Unwired WorkSpace randomly selects a parent through which the child synchronizes. Sybase recommends that you define one parent MBO as the composite parent to avoid unintended behavior. For example:</p> <p>There are two one-to-many relationships: A<--->>C and B<--->>C. Both MBOs A and B have a synchronization parameter while MBO C has none. In this case, one of the relationships should be a composite.</p>
<p>First/subsequent MBO relationship behavior</p>	<p>Multiple MBOs created from a single operation result:</p> <ul style="list-style-type: none"> • Subsequent MBOs as child entities (if the relationship mapping is based on a subsequent MBO load parameter) are supported, and the relationship is treated the same as a relationship on an attribute, meaning it does not affect MBO loading. • Relationships from subsequent to first MBOs where the relationship mapping is based on the load parameter of the first MBO are not supported.
<p>Local business objects</p>	<ul style="list-style-type: none"> • A relationship between two local business objects that does not have a primary key set is invalid. <p>You cannot create relationships between local business objects using a composite primary key.</p>

Guideline	Example
Multi-level insert operation behavior	Multi-level insert operations do not support unidirectional one to many relationships – newly created rows in the child MBO do not display prior to synchronization in a unidirectional one to many relationship. To view details prior to synchronization from the device application, go to the pending operations screen.
Automatic setting of primary keys	Relationships require all primary keys on the source or target MBO (source or target depends on the relationship type) to be referenced in the relationship. If you design a relationship that does not include all the primary keys, a warning prompt allows you to decide whether to automatically set reference attributes to primary keys, and unset the unreferenced attributes primary keys. If you select Yes , Unwired WorkSpace automatically sets the primary key, if not, Unwired WorkSpace maintains the existing primary key setting, and displays a validation message according to the relationship rules.
Personalization key and default value removed from child parameter.	When creating a relationship from an attribute of a parent MBO to a load parameter of a child MBO, the personalization key and default value cell of the load parameter is disabled, and the personalization key and the default values, if any, are cleared, since those values are not used to load the child's value from the EIS at run-time.

Remapping Attributes and Parameters

The remap option generates new mappings for unmapped attribute columns or parameter arguments based on the metadata of the data source to which the mobile business object (MBO) is bound.

For MBOs that already have mappings, selecting **Remap** remaps parameters and attributes to the data source based on the new data source definition.

Remap is available from both the Attributes and Parameters tabs in the Properties view, and also affects how Preview refreshes attributes and parameters. Selecting Remap:

- Queries the metadata for unmapped column or argument lists.
- Creates new attributes or parameters for unmapped columns or arguments using metadata information (name, type, default value).
- Generates new data source mappings for attributes or parameters.
- Establishes "Fill from attribute" links for parameters.

In the Preview dialog, Remap works automatically when refreshing MBOs if the definition of the MBO changes. For example, after you change the definition in the Definition dialog and select **Refresh**, not only is the data source metadata refreshed, a remapping of any unmapped attributes/parameters is automatically performed based on the refreshed metadata.

You have an MBO with attributes bond to a database table defined as:

```
Customers[ ID, Surname, GivenName, Address, Phone ]
```

As the developer, you want to eliminate the “Address” column and better represent the address information. You modify the database schema to:

```
Customers[ID, Surname, GivenName, Street, City, State, Country, PostalCode, Phone]
```

The MBO attributes become out of date because of the schema change, requiring the developer to **Refresh** the metadata. After you refresh, the "Address" attribute is unbound. Without remap, you would manually add attributes and map them to the columns in the tabular view. Selecting **Remap** creates the new attributes and maps them to the appropriate columns automatically based on the modified metadata.

Managing Personalization Keys

Personalization keys allow the mobile user to define (personalize) certain input field values within the mobile application, by associating a name (key) with a simple or complex datatype value.

Mobile development supports two types of personalization keys:

- User-defined – you can define these when developing a mobile business object. Before using these keys in a device application, each user sets their own values. For example; name, address, zip code, currency, location, customer list, and so on.
- System defined (username/password) – refers to the user's login credentials used to access enterprise information system (EIS) data. Unlike preference attributes, username/password is read-only and reset each time the user logs in or changes their password. The values are typically used as personalization attributes or other data source runtime credentials. The username/password system personalization keys do not support NULL values.

Creating a Personalization Key

Create a personalization key using the Personalization Key wizard.

1. Launch the Personalization Key wizard from either the:

- Workspace Navigator – right-click the Mobile Application project, the Personalization Keys folder within the Mobile Application project, or an existing personalization key within the Personalization Keys folder and select **New > Personalization Key**.
- Mobile Development perspective – select **File > New > Personalization Key**, or **File > New > Other > Mobile Application Project > Personalization Key**.

The New Personalization Key wizard displays:

2. Follow the wizard instructions and click **Finish** to create the new personalization key:

- Mobile Application project (available only if invoked from the Mobile Development perspective) – the project in which this personalization key is created. The newly created personalization key will be displayed under the Personalization Key folder of the selected Mobile Application project.

- Name – the name of the personalization key
- Type – select the supported data type of the personalization key value. If the type is an array, for example String[], then it supports a list of values.
- Nullable – accepts null as a valid value.
- Protected – obfuscates the personalization key value, making it more secure.
- Default value(s) – supports multiple values. Select the ellipsis (...) to **Add** or **Delete** multiple default values in these situations:
 - The type is BINARY/DATE/DATETIME/TIME.

Note: If you add a default value, for these data types, the default value is initially read only. To edit this value, click the ellipsis (...) in the Edit Properties dialog of the wizard. For the DATETIME and TIME types, note that you can add or edit the millisecond value. The precision of millisecond is limited to three digits. For example, a value of .1 is rendered as .100.

- The type is a primitive list. For example, STRING[], BINARY[], and so on.
- The type is a structure or list of structures.

All default values must be of the same data type as specified in the Type field.

- Storage – determines where the key values are stored and maintained. Options include:
 - Server – on Unwired Server
 - Client – on the device client
 - Transient – only saved in memory of the current login session
- Description – (optional) the description of this personalization key

Copy and Pasting Personalization Keys

Create a new personalization key by copy and pasting an existing key.

1. Navigate to the existing personalization key you want to copy by expanding the Mobile Application project folder, then the Personalization Keys folder.
2. Right-click the personalization key you want to copy and select **Copy**.
3. Navigate to the Mobile Application project to which you want to paste the personalization key. Right-click the Personalization Keys folder and select **Paste**.

If the pasted personalization key has the same name as an existing personalization key in the Personalization Keys folder to which you are pasting, rename the personalization key. If you are copying and pasting a personalization key that contains a structure, the same structure must be available in the project to which the personalization key is copied.

Modifying Personalization Key Properties

Modify common personalization key properties from the personalization key's Properties dialog box.

1. Access the Properties dialog box by right-clicking the personalization key you want to modify and selecting **Properties**.

Develop

2. From the Common tab, modify any of these properties:

- Name
- Type
- Nullable (checkbox)
- Protected (checkbox)
- Default value(s)
- Storage
- Description

3. Click **OK** to save your changes.

Deleting a Personalization Key

Delete an existing personalization key from the Personalization Keys folder.

To determine the mobile business objects, attributes, and operations referenced by the personalization key you are deleting, right-click the personalization key and select **References > option** (where option is the reference type of interest).

1. Navigate to the personalization key you want to delete, by expanding the Mobile Application folder then the Personalization Keys folder.

2. Right-click the personalization key and select **Delete**.

A confirmation dialog appears with a list of all mobile business objects, attributes, and operations referenced by the personalization key.

3. Click **Ok** to delete the personalization key.

All MBO and operation assignments are removed and the personalization key is deleted from the Personalization Keys folder.

Personalization Key Limitations

Personalization keys should not exceed 512 bytes.

If total length of personalization keys in a package exceeds 512 bytes, the warning

```
The total length of personalization keys in the
package exceeds maximum safe length limit of 512 bytes
```

displays in the Eclipse Problems view, and the project and deployment package contains a warning icon.

If the length of a specific personalization key's default value exceeds 512 bytes, the warning

```
The length of personalization key 'PersonalizationkeyName'
exceeds maximum safe length limit of 512 bytes
```

displays in the Eclipse Problems View and Personalization key Properties page.

Managing Roles and Permissions

Logical roles provide authorization for mobile business objects and operations during development.

The Sybase Unwired Platform supports two types of roles:

- Logical roles – defined and used within the development environment to provide security to mobile business objects and MBO operations during development, and eventually mapped to physical roles during deployment. Mobile device users have no interaction with logical roles.
- Physical roles – enterprise-oriented and managed by the Unwired Server administrator or through some other enterprise security provider. Physical roles control the access to the back-end data sources of an MBO during runtime.

When deploying MBOs to a Unwired Server, you can map logical roles to existing physical roles that are located on the Unwired Server. The mapping transfers authorization and other properties from the physical role to the logical role.

The need for role mapping exists because, in most cases, any role-based authorization used while developing an MBO is invalid once the MBO is deployed to the Unwired Server, since it is likely the Unwired Server uses a different security mechanism/set of roles, or the data source changes and uses different authorization than used during development.

If development and Unwired Servers do use the same set of roles, you can map the logical role name directly to the physical role when deploying the MBO.

The Roles folder contains user-defined roles, that can be modified and reused.

Role assignments are not propagated from the mobile business object to the operations it contains. If you want to control access to any operation, you must explicitly set the appropriate role (by default, if no role is assigned then the operation is assigned the role 'everybody' which allows unlimited access).

Creating Logical Roles

Use the New Role wizard to create a logical role in the Mobile Application project's Roles folder.

The context from which you launch the New Role wizard determines the default project location of the new role.

1. Launch the New Role wizard by selecting **File > New > Other > Sybase > Mobile Development > Role**.
2. Specify:

Table 48. Create a new logical role

Field	Description
Mobile Application project	Select the Mobile Application project to which this role is added. The role is stored in the Roles folder of the selected Mobile Application project.
Name	Enter a name for the new role.
Description	(Optional) Enter a description for the role.
Default role	(Optional) set this role as the default role assigned to all mobile business objects and operations created from that point forward for the specified Mobile Application project.

3. Click **Finish.**

The new logical role is now available from the Roles folder of the Mobile Application project in which it was created.

Copy and Pasting Logical Roles

Create a new logical role by copying and pasting an existing role.

1. Navigate to the existing role you want to copy by expanding the Mobile Application project folder, then the Roles folder.
2. Right-click the role you want to copy and select **Copy**.
3. Navigate to the Mobile Application project to which you want to paste the role. Right-click the Roles folder and select **Paste**.

If the pasted role has the same name as an existing role in the Roles folder to which you are pasting, rename the role.

Modifying Logical Role Properties

Modify common role properties from the role's Properties dialog box.

1. Access the Properties dialog box by right-clicking the role you want to modify and selecting **Properties**.
2. Select **Common**.
3. Modify:
 - Name – role name.
 - Description – role description.
 - Default role – set and unset this role as the default.
4. Click **OK** to save your changes.

Setting and Unsetting the Default Logical Role

Optionally set a default role that is assigned to all mobile business objects and operations created from that point forward for the Mobile Application project.

1. Navigate to the role you want to set as the default by expanding the Mobile Application folder then the Roles folder.
2. Right-click the role and select **Set as Default**.
3. To remove the default setting, select the default role and click **Set as Default** again.

Assigning Roles to Mobile Business Objects and Operations

Assign logical roles to mobile business objects and operations using the role's Properties dialog box.

1. Right-click the role you want to assign and select **Properties**.
2. Select **Assignments**.
3. Modify role assignments by selecting the tab that corresponds to the assignment you want to make:
 - **Mobile Business Objects** – displays all MBOs that are contained in the same Mobile Application project as the role. Select individual MBOs to which you assign the role, or use the **Select All** and **Deselect All** buttons. Assigning a role to an MBO assigns the role to all operations and attributes of that MBO.
 - **Operations** – displays all MBO operations that are contained in the same Mobile Application project as the role. Select individual operations to which you assign the role, or use the **Select All** and **Deselect All** buttons.

Note: You can also assign a role by dragging-and-dropping the role to the operation or MBO.

4. Click **OK** to save your changes.

Finding Role References

Locate mobile business objects and operations that the role references.

1. Navigate to the role whose references you want to find by expanding the project folder then the Roles folder.
2. Right-click the role, select **References**, then select:
 - **Mobile Business Object** – displays mobile business objects referenced by this role.
 - **Operation** – displays mobile business object operations referenced by this role.
 - **All References** – displays mobile business objects and operations referenced by this role.

Referenced objects display in the Search view. Double-clicking objects in the Search view changes the focus in the Mobile Application Diagram to the object.

Deleting a Logical Role

Delete an existing logical role from the Roles folder.

1. Navigate to the role you want to delete, by expanding the project folder then the Roles folder.
2. Right-click the role and select **Delete**.
3. Click **Yes** to delete the role.

The Delete dialog box displays all mobile business objects and operations referenced by the role.

All mobile business object and operation assignments are removed and the role is deleted from the Roles folder.

Mobile Business Object Mobility Properties

Mobility properties determine data movement within the enterprise, once mobile business objects are deployed to Unwired Server and device applications access MBO data.

Synchronization

Determine the amount of data (filter), and under what conditions (timing and triggers), mobile business objects (MBOs) upload data to and download data from Unwired Server.

Synchronization properties are unavailable for MBOs in cache groups that use an Online policy.

Synchronization Parameter Considerations

Modeling of synchronization parameters implicitly generates data partition keys within the Unwired Server cache. Partition keys define subsections of data within an MBO, enabling parallel data access to large MBO data sets.

For example, you can refresh multiple partitions in parallel, or query one partition while another is being refreshed. In general, partitions prevent serialized access to the cache. Some best practices for defining partition keys include:

- Synchronization parameters should be defined and mapped to all result-affecting load parameters. Failure to do so results in partitions being continually overwritten/deleted which leads to unexpected results in the mobile client.
- Result-affecting load parameters are those parameters of the EIS read operation that affect the results of the operation. Some parameters may be information needed by the EIS but do not actually affect the results of the read operation. For example; suppose an MBO is modeled using a Web Service operation “getAllBooksByAuthor(AuthorName, userKey) where userKey is simply a mechanism to authenticate a user and does not effect the results of the operation. For a given “AuthorName” the service will return the same list of books

regardless of the “userKey” value. In this case “userKey” is not a result-affecting parameter and therefore should not be mapped to a synchronization key.

Defining Synchronization Properties for Individual Mobile Business Objects

Each mobile business object (MBO) that can be synchronized includes a Synchronization tab from which you configure synchronization behavior.

1. From the Mobile Application Diagram, right-click the MBO for which you are configuring synchronization, and select **Show Properties View**.

Note: MBOs not bound to a data source or that use an Online cache policy do not support synchronization.

2. In the Properties view, select the **Synchronization** tab.

Note: If you do not see the Synchronization tab, switch to the Advanced Developer profile.

3. Complete the synchronization definition:

Use the **Add**, **Delete**, and **Delete All** buttons to create or remove parameters. You can also supply additional parameter information for each parameter, including:

- **Parameter name** – by default each new parameter name is parameter N (N is the number of the parameter), which you can change.
- **Datatype** – the datatype of the parameter. Datatypes in the Personalization key and Mapped-to columns must be compatible with the datatype selected here.
- **Nullable** – accepts null as a valid value. Unselect this option if the argument to which this parameter is mapped does not support null as a valid value.
- **Default value** – enter a valid value.
- **Personalization key** – maps the synchronization parameter to a personalization-key value. If you specify a **Personalization Key**, the client is not required to provide a value, but must define a personalization value for the key before successfully using the MBO.
- **Mapped to** – maps the synchronization parameter to an existing attribute. During synchronization, the application's synchronized local cache contains only entities that match the synchronization parameter value with that of the mapped attribute.
- **Query limiting – deprecated**. Instead use dynamic device-side queries to limit synchronization to the last value entered by the device application user.

4. Select **Customized download data** to generate the SQL statement that defines the synchronization filter. Unselecting this option clears the statement.

By default, the internally generated SQL statement, based on the synchronization parameters, includes only the '=' operator. To properly generate the SQL for any MBO that is bound to a JDBC data source that includes a **where** clause operator other than '=' (>, <, <>, !=, >=, <=, !=>, !=<, and so on), update the generated SQL statement after selecting **Customized download data**. For example, change the '=' operator to '>'.

Next

(Optional) For a parameter to serve as both a load parameter and a synchronization parameter, go to the attribute's **Load Parameters** tab after you define the synchronization parameter, and select **Synchronization parameter** for the parameter to which the synchronization parameter is mapped.

Understanding Synchronization Parameters

Synchronization parameters restrict the rows that are transferred from the Unwired Server cache database (CDB) to the device to match values the client provides. A synchronization parameter does not affect enterprise information system (EIS) interaction with the CDB, unless you specify the Synchronization Parameter setting for a given load parameter.

After you bind a mobile business object (MBO) to a data source, MBO attributes map to database columns. You control the amount of data synchronized (filtered) between the CDB and device application by defining synchronization parameters that map to attributes. For example, mapping a synchronization parameter to the "state" attribute, returns customer records for a particular state based on the value entered by the device application user.

Another example is an MBO named "sales_order" with a synchronization parameter mapping to the "region" attribute, and deploy the MBO to Unwired Server, executing this query from the device application:

```
SELECT * FROM sales_order
```

returns a complete copy of all sales orders in the CDB. If the application user provides a region when synchronizing, for example "Eastern", the client sees sales orders only for the Eastern region.

Using a synchronization parameter to filter results may be particularly useful for MBOs that have large amounts of data that do not change frequently, making periodic bulk loads and a longer cache interval more appropriate. For example, use select * from customer to bulk-load all customers. Then design a synchronization parameter that maps to the "state" attribute. To load only California customers, the device application user passes in the "CA" parameter.

Synchronization Parameter Definition Guidelines

Understand guidelines and restrictions when defining synchronization parameters.

Guideline	Description
Datetime and time synchronization parameters	<p>Synchronization may fail if you use SQL Anywhere as the Unwired Server cache database (CDB) when the synchronization parameter is a datetime or time datatype, since datetime and time columns contain three digits for the fraction portion, making direct comparison incompatible with Unwired WorkSpace.</p> <p>To compare a datetime or time datatype to a string as a string, use the DATEFORMAT function or CAST function to convert the datetime or time datatype to a string before comparing. For example, this SQL statement is the generated download cursor when attribute <code>c2(datetime)</code> is specified as the synchronization parameter:</p> <pre>SELECT x.* FROM Mydatetime x WHERE ((x.c2=:c2) OR ((x.c2 IS NULL) AND (:c2 IS NULL)))</pre> <p>Change the statement to:</p> <pre>SELECT x.* FROM Mydatetime x WHERE ((dateformat(x.c2, 'yyyy-mm-dd hh:mm:ss') = dateformat(:c2, 'yyyy-mm-dd hh:mm:ss')) OR ((x.c2 IS NULL) AND (:c2 IS NULL)))</pre>
Datatype and nullable default values	When a synchronization parameter is mapped to an attribute, to maintain consistency between the two, the parameter datatype and nullable fields follow that of the attribute to which it is mapped and become read-only.

Synchronization Groups

A synchronization group specifies the synchronization behavior for every mobile business object (MBO) within that group.

Creating Synchronization Groups

Synchronization group folders define the synchronization schedules for the mobile business objects (MBOs) within them. Create as many synchronization groups as required to meet the varying synchronization schedules of the MBOs for a given mobile application project.

1. To launch the New Synchronization Group wizard, either:
 - Right-click any folder from the WorkSpace Navigator, and select **New > Synchronization group**, or
 - From Unwired WorkSpace, select **File > New > Synchronization group**.
2. Specify properties for the synchronization group and click **Finish**.
 - **Name** – name of the synchronization group.

- **Change detection interval** – the frequency, in hours, minutes, and seconds, with which Unwired Server is notified of data changes within the synchronization group.
- **Description** – an optional description.
- **Use as default synchronization group** – identifies the synchronization group as the project's default. All MBOs automatically belong to the default synchronization group when created, except MBOs that are not bound to any data source, and cannot be synchronized.

Unwired WorkSpace includes a read-only "Default" synchronization groups folder (that uses a change detection level of 10 minutes), which ensures there is always at least one folder available.

The synchronization group folder you just added appears under the Synchronization groups parent folder in the mobile application project for which it was created.

Deleting Synchronization Groups

Deleting a synchronization group folder assigns all mobile business objects (MBOs) currently in the folder to the default Synchronization group folder.

You cannot delete the predefined "Default" synchronization group.

1. From WorkSpace Navigator, expand the project folder that contains the synchronization group folder.
2. Right-click the synchronization group folder of interest and select **Delete**.

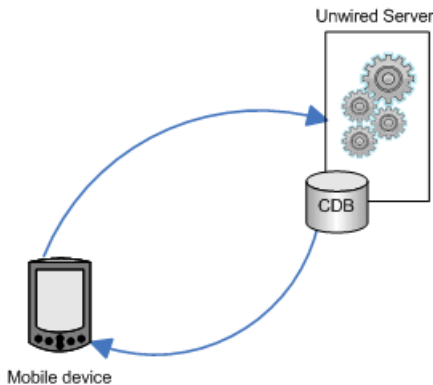
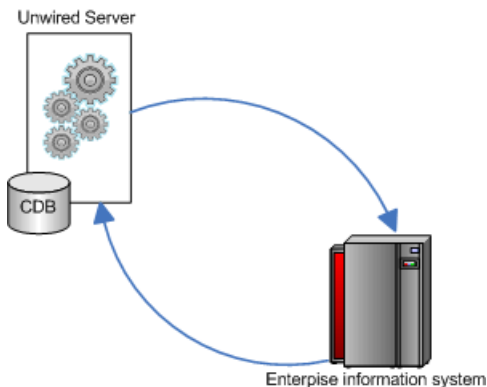
All MBOs in the deleted synchronization group folder are assigned to the default synchronization group folder.

Load Parameters

Load parameters control the amount of data refreshed between the enterprise information system (EIS) and the cache database (CDB), and creates partitions based on load parameter values. In contrast, synchronization parameters filter CDB data during device application synchronization.

Set load parameters in the Properties view, from the Load Parameters tab. Set synchronization parameters from the Synchronization tab. It is important to understand both their differences and how they work together to load (data refresh) and filter (synchronize) data. For example, you can define:

- A synchronization parameter and a separate load parameter – refresh data based on a parameter independent of synchronization, or
- A load parameter that maps to a synchronization parameter – use the same parameter for both refreshing and synchronizing data.

Figure 1: Synchronization parameter**Figure 2: Load parameter****See also**

- *Mobile Business Object Attribute Properties* on page 151
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a *Synchronization* tab from which you configure synchronization behavior.

Adding a Load Parameter to a Mobile Business Object

Load parameters control the amount of data refreshed between the enterprise information system (EIS) and the Unwired Server cache database (CDB). You can also map them to synchronization parameters, unless the MBO for which you are defining the load parameter is in a cache group that uses an Online policy.

You can use mobile business object (MBO) load parameters to control data refresh, create partitions, and, if mapped to a synchronization parameter, filter data that is returned to the mobile device. If you drag and drop a data source, or otherwise create MBO attributes that

automatically generate attribute parameter-to-argument mapping, steps 1–3 below are optional. Do not confuse load parameters with operation parameters, which are defined or modified for individual operations.

1. In the Mobile Application Diagram, right-click the MBO, and select **Show Properties View**. Alternatively, click in the MBO title header to open the associated MBO properties view.
2. In the Properties view, select the **Attributes** tab, then select the **Load Parameters** tab.

Any parameters defined as part of the MBO definition automatically appear as load parameters. For example, if your MBO is bound to the sample database's customer table, and the SQL definition is:

```
SELECT customer.id,
       customer.fname,
       customer.lname,
       customer.address,
       customer.city,
       customer.state,
       customer.zip,
       customer.phone,
       customer.company_name
FROM customer
WHERE customer.state = :state_name
```

then **state_name** is a load parameter and **state** is the **Propagate to attribute**. You can modify this load parameter or add additional load parameters from the Load Parameters tab.

3. Click **Add**.

A new parameter is created with the default name of "parameter1." The next parameter you create is called "parameter2," and so on.

4. For each parameter, specify:

Property	Description
Parameters	
Name	Name of the mobile business object parameter. Names cannot contain C# or Java reserved words.
Datatype	Select the parameter datatype. Datatypes include string, date, int, datetime, decimal, and so on, and map to the data source's datatype.
Nullable	Select this option if Null is a valid value for the parameter, the corresponding data source Argument to which it maps, and the Propagate to attribute .

Property	Description
Personalization key	(Optional) You can select a personalization key to map to the parameter, which provides an argument value. For example, you can create and specify the "state" personalization key, and return values that match a specific value, such as "California." If Synchronization parameter is selected, this option is unavailable.
Synchronization parameter	(Optional and unavailable for MBOs in cache groups that use an Online policy) Map the load parameter to an existing synchronization parameter (defined from the Synchronization tab) to filter the results that are downloaded to mobile devices based on the value. The Default value and Personalization key options are ignored if you select synchronization parameter.
Propagate to attribute	(Optional) Select an attribute to use its value for that of the parameter.
Data Source	
Argument	Data source argument name to which the parameter is mapped. If you select Unmap as the data source argument, an extra unnamed row is added to the load parameter list. This is not an error. The data source requires an argument mapping for these parameters. Selecting Refresh , or remapping the parameter removes the extra row.
Datatype	Datatype of the data source argument to which the parameter is mapped. You can change the argument's datatype only if it maps to a parameter bound to a JDBC data source.
Nullable	Select this option only for EIS arguments that support NULL as a valid value.
Default value	(Optional) Enter a default value for the parameter. If Synchronization parameter is selected, this option is unavailable.

Property	Description
Show/Hide figure	Once you define a parameter, select Show figure to view a graphical representation of the parameter-to-data source argument mapping. You can modify the mapping by dragging the connector line from the parameter to a different argument, which automatically updates the datatype to that of the argument.

See also

- *Mobile Business Object Properties* on page 146
- *Mobile Business Object Attribute Properties* on page 151
- *Mobile Business Object Operation Properties* on page 153
- *Datatype Support* on page 125
- *Old Value Argument* on page 149

Combining Load and Synchronization Parameters

Combine parameter settings to control how data is cached in the CDB, and filtered and returned to a device application.

Since data refresh and synchronization is controlled through parameters, design and implement load and synchronization parameters for efficient data control, especially where large sets of data are involved, or Unwired Server and device applications may perform poorly. For example:

- Unwired Server – defining load parameters that load too much enterprise information system (EIS) data into the CDB at a given time can impact performance and, in some cases, generate memory errors (`java.lang.OutOfMemoryError`). Define load parameters to divide data into smaller segments so data loads efficiently. Otherwise, you may have to increase the server’s JVM heap size to accommodate the extra load.
- Device application – synchronization parameters limit CDB to device application data. If you do not define synchronization parameters, a client may download all data in the CDB (for a particular MBO), and, in the worst case, cause the device application to crash. Define synchronization parameters to divide data into manageable segments so every synchronization finishes quickly.

These results are based on the customer table in the `sampledb` database in the My Sample Database connection profile.

Parameters	Result
No load or synchronization parameters are defined.	All table data is downloaded to the device. The SQL definition is: <pre>select * from sampledb.dba.customer</pre>
The region synchronization parameter is defined, but not used as a load parameter.	Only customers from a specific region are downloaded to the device. Typically, region is paired with a personalization key. For example, a sales representative living and working in the western region is interested only in customers from that region. The SQL definition is: <pre>select cust_id, cust_name, region from sampledb.dba.customer where region=:region</pre>
The region parameter is defined, used as a load parameter, and as a synchronization parameter.	Occurs if data refresh requires a region parameter and a synchronization parameter. This scenario is more likely for Web service and SAP MBOs than for database MBOs.
Load parameters are mapped to username and password personalization keys, and a separate synchronization parameter is mapped to region.	Refresh data only for the authenticated user, but synchronize based on the region: <pre>select cust_id, cust_name, region from sampledb.dba.customer where region=:region, for user A.</pre>

See also

- *Mobile Business Object Attribute Properties* on page 151
- *Defining Synchronization Properties for Individual Mobile Business Objects* Each mobile business object (MBO) that can be synchronized includes a *Synchronization* tab from which you configure synchronization behavior.

Mapping a Load Parameter to a Synchronization Parameter

Load parameters affect enterprise information system (EIS) caching on Unwired Server when a mobile business object (MBO) is accessed by a device application and filters (if it is also mapped to a synchronization parameter) data that is downloaded to the device application.

1. From the Mobile Application Diagram, right-click the MBO for which you are configuring synchronization and select **Show Properties View**.
2. In the Properties view, select the **Attributes** tab from the left side, then select the **Load Parameters** tab, located on the top.
3. To use the load parameter for synchronization, select the **Synchronization parameter** from the drop-down list.

If the load parameter maps to a synchronization parameter, the value can be supplied by the client each time it synchronizes the MBO. During subsequent synchronizations, the client may provide different values for the parameter, which affects EIS data refresh results from Unwired Server.

If the parameter includes a default value, a client-supplied value is optional.

Example: Parameters and Stored Procedures

Learn how to create a mobile business object (MBO) from a stored procedure that contains parameters.

This example creates a stored procedure with two parameters in the `sampledb` that serves as a customer address book. Passing in the first and last names returns address information. For example, you can use SQL Scrapbook to create a stored procedure using this statement:

```
create PROCEDURE dba.getCustomerAddress
    (@lname_parm varchar(15), @fname_parm varchar(15))
AS
BEGIN
    select address,
           city,
           state,
           zip
    from customer
    where lname = @lname_parm and fname = @fname_parm
END
```

Create an MBO from the stored procedure. For example, drag and drop the **getCustomerAddress** stored procedure onto the Mobile Application Diagram. The SQL query for this statement, which calls two parameters is:

```
{CALL sampledb.dba.getCustomerAddress(:lname_parm, :fname_parm)}
```

which are the default load parameters.

Cache Partitions

Partitioning the Unwired Server cache (CDB) divides it into segments that can be refreshed individually, which gives faster system performance than refreshing the entire CDB.

Cache partitioning is determined by a partition key, which is a load parameter used by the operation to load data into the cache from the enterprise information system (EIS).

Define a partition key from a mobile business object (MBO) attribute column, or define a composite partition key from multiple attributes. A device application user specifies a value for the load parameter when synchronizing his or her client application, possibly through a personalization key that is also used as a synchronization parameter.

All cache partitions require a load parameter:

- Create cache partitions through a load parameter specified by the client, for example, a load parameter that uses the synchronization parameter option and maps to a personalization key.
- Refresh a cache partition if data in the partition is:
 - Expired
 - Invalidated
 - Inconsistent – if a client has multiple partitions, refresh all partitions even if only one partition expires.
- If the MBO is defined with something other than "=" in the **where** clause, manually edit, in the synchronization tab, the SQL code for the customized download data.
- Although you create a partition key based one or more MBO attributes, you rarely use the primary-key column, since doing so creates a partition for every row in the CDB. The goal of partitioning is to place only the data you need in a partition.

On-demand versus scheduled refresh

Cache policy refresh options affect cache partitions, in that they determine the frequency with which the CDB is updated from the EIS (data refresh). The scheduled refresh option refreshes the cache based on a clock. The on-demand refresh option, which is based on client actions, is discussed here, with an emphasis on how to determine if cache partitions behave as expected.

To validate CDB data partition refresh behavior:

1. Define multiple partitions and multiple clients.
2. Define a cache group policy and confirm cache refresh behavior. Set the cache policy with a cache interval that is long enough to allow you to perform updates to the EIS and synchronize both clients.
3. Wait until the cache interval passes, then resynchronize the clients. The second synchronization should reflect EIS changes, since the cache policy dictates that the CDB must now refresh.
4. Inspect CDB log files for time stamps in the "LAST_REFRESH" and "LMD" columns for your package to confirm that the partitions and rows of data for the associated partitions have refreshed as expected.

To confirm the correct partitions have refreshed:

1. Make updates to data from multiple partitions in the EIS.
2. Synchronize one client so only one partition refreshes. Make sure the values for the length of the cache interval and the last time the partition refreshed indicate that the data in that partition is stale and needs to refresh when you synchronize.
3. The synchronized client retrieves refreshed rows only from the partition of interest. Data from other EIS partitions do not update the CDB, even though data has changed in the EIS (because no client has requested a synchronization of that partition).
4. When the second client synchronizes the second partition, only that partition refreshes.

Examples: Parameters and Cache Partitions

Create cache partitions based on mobile business object (MBO) and load parameter definitions.

Creating cache partitions based on personalization-key values

These examples use the employee table in the My Sample Database connection profile.

Create a mobile business object (MBO) that uses a parameter and personalization key to partition the Unwired Server cache database (CDB). The general process is:

- In Unwired WorkSpace, create a MBO with a parameter from the employee table, add a personalization key, and define the load parameters that define how the CDB is partitioned
- Client (device application) partitions are created as new clients connect. Users set the personalization key in their application, and then synchronize and download data.
- The CDB loads data that satisfies the MBO definition using the load parameter value, which is the personalization key value in this example, passed by the client, and returns only those rows that matches the client's personalization key value.

For this example:

1. Drag and drop the employee table, and edit the definition to include the state_param parameter:

```
SELECT emp_id,
       manager_id,
       emp_fname,
       emp_lname,
       dept_id,
       street,
       city,
       state,
       zip_code,
       phone,
       status,
       ss_number,
       salary,
       start_date,
       termination_date,
       birth_date,
       bene_health_ins,
       bene_life_ins,
       bene_day_care,
       sex FROM sampledb.dba.employee
where state = :state_param
```

2. Create a personalization key with these values:
 - **Name** – state_pk
 - **Type** – string(4)
 - **All other entries** – accept default values
3. From the Attributes Load Parameters tab, define the load parameter:

- **Name** – state_param
 - **Datatype** – string(4)
 - **Nullable** – no
 - **Propagate to** – state
 - **Personalization key** – state_pk
 - **Argument** – state_param
 - **Datatype** – string
 - **Nullable** – no
4. Deploy the package to Unwired Server.
Client and CDB behavior is:
- Client 1 sets state_pk to "TX ", while client 2 uses "GA ". Two rows (one for each parameter) are added to the parameter table in the CDB. Two trailing white spaces are added to pad the total length to four, since state_pk is defined as string(4).
 - The CDB partition table contains values that define the partition key for each partition (TX and GA).
 - The partition refresh table tracks the most recent refresh for each partition.

Only the data in the partition of interest refreshes. This is an important performance consideration for large tables.

Creating cache partitions based on compound parameter values

This example shows how to create a partitioned cache for the employee table where the partitions are defined by a compound partition key that uses two attributes: city and state.

Manually edit the SQL definition. For a query that does not require exact matches for the state or city parameters, use this MBO definition as the download query:

```
SELECT emp_id,
       manager_id,
       emp_fname,
       emp_lname,
       dept_id,
       street,
       city,
       state,
FROM sampledb.dba.employee
WHERE state LIKE :state_param + '%'
AND city LIKE :city_param + '%'
```

Configuring Mobile Business Objects for Mobile Workflow Online Data Access

You can define load parameters that determines what data to return to a Mobile Workflow application.

You can map load parameters to either transient personalization keys or propagate-to attributes to define load parameters used within a Workflow application, the difference is:

- personalization keys – the value is passed in as personalization key values. For example, a user name and password that returns a result set for that particular user.
- propagate-to attribute – when this method is used, and the MBO uses an Online cache group policy, the value of the load parameter always comes from findByParameter object query parameter. And the data is real-time enterprise information system (EIS) data. That is, every call to the object query results in an immediate data refresh, and delivery of requested data to the client.

Defining Mobile Workflow Load Parameters From Mapped Transient personalization Keys

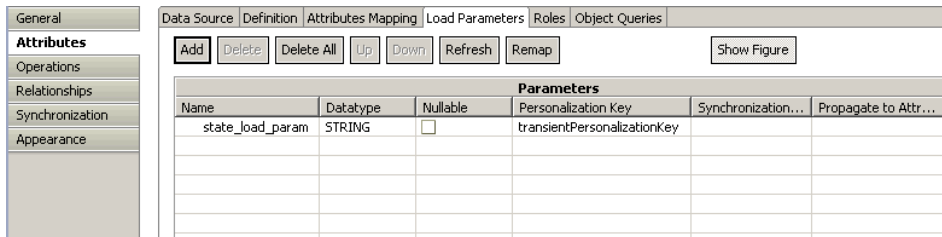
Map load parameters to transient personalization keys. Specify values for those personalization keys when you invoke an operation or object query from the mobile workflow form.

The basic task flow for mapping and specifying values for transient personalization keys from Unwired WorkSpace is:

1. Create a mobile business object that has load parameters. For example, the Customer MBO could have this definition:

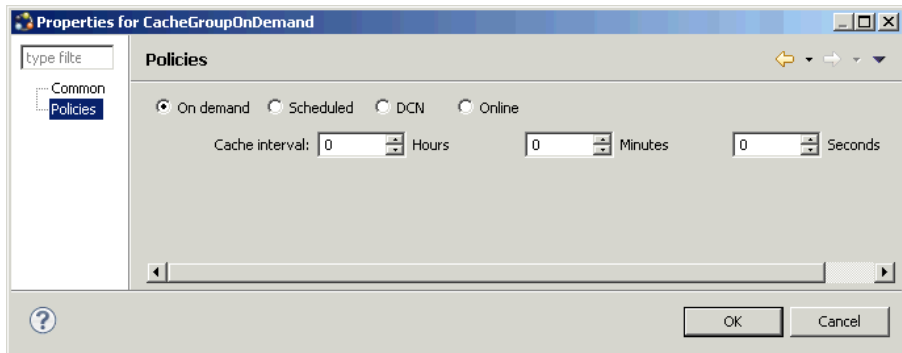
```
SELECT id,
       fname,
       lname,
       address,
       city,
       state,
       zip,
       phone,
       company_name FROM sampledb.dba.customer
where state = :state_load_param
```

2. Create a corresponding personalization key for each load parameter where the Storage type is **Transient**.
3. Map each load parameter to the corresponding personalization key. Set the load parameters in the MBO Properties view, from the Load Parameters tab.



4. (Optional – specify the cache group policy that meets the Workflow client's needs. The policy defined here, results in a immediate data refresh for every client request). Set the **On demand** cache group policy for the MBO with a **Cache interval** of zero.

- a) Add the MBO to a Cache Group that uses the On demand cache group policy. For example, create a new cache group named CacheGroupOnDemand and set the policy to **On demand** and the **Cache interval** to zero.



- b) Drag and drop the MBO to the CacheGroupOnDemand cache group. At runtime, when the client passes in their transient personalization key values (user name and password for example), the entire cache for the MBO refreshes prior to synchronization. While this ensures the device receives up-to-date EIS data, it is more costly in terms of resources compared to a cache policy with a longer cache interval.
5. Deploy the project that contains the MBO to Unwired WorkSpace. Select Message-based in the deployment wizard.

Defining Mobile Workflow Load Parameters from Mapped Propagate to Attributes

Create an MBO with at least one load parameter, map parameters as propagate to attributes, then assign the MBO to a cache group that uses an Online policy.

1. From Unwired WorkSpace, create an MBO that has at least one load parameter. For example, you could define an Emp MBO as:

```
SELECT  id,
        empName,
        empDeptId FROM sampledb.dba.emp
WHERE empDeptId = :deptIdLP
```

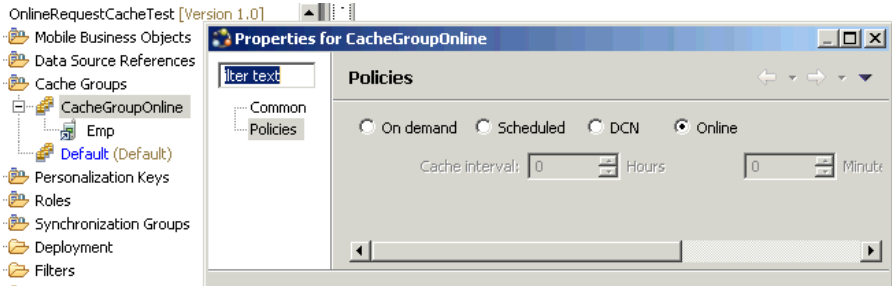
2. In the MBO Properties view, select the **Load Parameters** tab, map each load parameter to be used as an operation load parameter for the Mobile Workflow package to a Propagate to Attribute. This example requires you to map the deptIdLP load parameter to the empDeptId attribute. You must also verify that data types are INT and the default value is a valid INT.

General	Data Source	Definition	Attributes Mapping	Load Parameters	Roles	Object Queries															
Attributes	<div style="display: flex; justify-content: space-between; align-items: center;"> Add Delete Delete All Refresh Remap Show </div>																				
Operations	Parameters																				
Relationships	<table border="1" style="width: 100%;"> <thead> <tr> <th>Name</th> <th>Datatype</th> <th>Null...</th> <th>Propagate to Attribute</th> <th>Argument</th> </tr> </thead> <tbody> <tr> <td>deptIdLP</td> <td>INT</td> <td><input type="checkbox"/></td> <td>empDeptId</td> <td>deptIdLP</td> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table>						Name	Datatype	Null...	Propagate to Attribute	Argument	deptIdLP	INT	<input type="checkbox"/>	empDeptId	deptIdLP					
Name	Datatype	Null...	Propagate to Attribute	Argument																	
deptIdLP	INT	<input type="checkbox"/>	empDeptId	deptIdLP																	
Appearance																					

Unmapped parameters are set to NULL, and get their value from the default value, if specified, or from the personalization key value to which they are mapped, if specified. If the key is unmapped, and the parameter has no default value and is not mapped to a personalization key value, the parameter value is empty (NULL for string, 0 for numeric, and so on).

3. Set the Online cache group policy for the MBO.

- a) Add the MBO to a cache group that uses the Online cache group policy. For example, create a new cache group named CacheGroupOnline and set the policy to **Online**.



- b) Drag and drop the MBO to CacheGroupOnline.

The findByParameter object query is automatically generated based on all load parameters that have propagate-to attributes:

Name	Return type	Create an index	Parameters	Query Definition
findByParameter	Multiple objects	<input checked="" type="checkbox"/>	deptIdLP	SELECT x.* FROM Emp x WHERE x.empDeptId = :deptIdLP

4. Deploy the project that contains the MBO to Unwired Server. Select **Message-based** in the deployment wizard.

Cache Groups

A cache group specifies the data refresh behavior for every mobile business object (MBO) within that group.

During development, you can group MBOs based on their data refresh requirements. Some terms and concepts you should be familiar with are:

- **Cache group** – includes a cache policy and the MBOs that share that policy. An MBO can belong to only one cache group.
- **Cache** – MBO data in the Unwired Server cache (CDB) can be refreshed according to a cache policy, along with other mechanisms, such as data change notification (DCN).
- **Cache policy** – defines the cache refresh behavior and properties for the MBOs within the cache group based on the policy:
 - **On demand** – the cache expires after a certain period of time such as 10 minutes. The cache is not updated until a request is made of the cache and the cache has expired. If a

request is made of the cache and it is expired, there may be a delay responding to the request while the cache is refreshed.

- **Scheduled** – the cache is refreshed according to a schedule such as 7:00 am, 1:00 pm, or 6:00 pm. Note that load parameters filled from transient personalization keys can not be used with a scheduled cache type.
- **DCN** – the cache never expires. Data refresh is triggered by an enterprise information system (EIS) Data Change Notification. The cache interval fields are disabled when DCN is selected. See the *Developer Guide for Unwired Server* for details about implementing DCN.
- **Online** – only can be used with message-based mobile workflow applications. See *Online Cache Group Policy*.

Each cache group contains a cache policy, which in turn contains cache refresh/update properties. When a refresh occurs, the Unwired Server calls the default read operation (for each MBO in the cache group), and all of the rows that are returned from the enterprise information system (EIS) are compared to existing rows in the CDB as follows:

- If the CDB is empty, all rows are inserted.
- If any rows exist in the CDB, Unwired Server processes the row-set and checks (using the primary key) to determine if the row already exists in the cache:
 - If it does, and all columns are the same as the EIS, nothing happens. When a client synchronizes to request all rows that have changed since the last synchronization, only rows that have changed are included, which is important for performance and efficiency.
 - If the row does not exist, it is inserted and the next synchronization query retrieves the row.

Cache Group Considerations

Define multiple cache groups for data based on specific usage patterns and consistency tolerance. For example, for transactional data that has little tolerance for EIS data that is stale, an onDemand cache policy with a large coherence window is a more appropriate cache solution than an hourly schedule-based refresh.

Some best practices for defining cache groups and allocating MBOs to those groups include:

- Place all MBOs that are modeled with composite relationships in the same cache group.
- Do not place Reference data and transactional data in the same cache group. Typically, reference data has different data consistency requirements than transactional data.
- Avoid circular dependencies between cache groups.
- Avoid loading of an MBO in one cache group based on the attributes of an MBO in another cache group.

Creating Cache Groups

Cache group folders define the data refresh schedules for the mobile business objects (MBOs) within them. Create as many cache groups as required to meet the varying data refresh needs of the MBOs for a given mobile application project.

1. Switch to the Advanced developer profile by right-clicking in the Mobile Application Diagram and selecting **Switch developer profile > Advanced**.
2. To launch the New Cache Group wizard, either:
 - Right-click any folder from the WorkSpace Navigator, and select **New > Cache Group**, or
 - From Unwired WorkSpace, select **File > New > Cache Group**.
3. Specify general properties and the data refresh schedule for the cache group and click **Finish**.
 - **Mobile application project** – the mobile application project to which this cache group belongs. This option does not appear if you launch the wizard from the WorkSpace Navigator.
 - **Name** – name of the cache group.
 - **Description** – an optional description.
 - **Policy** – select the policy to be used by all MBOs within the cache group. The cache group policy determines the method and timing by which the Unwired Server cache is refreshed from the enterprise information system (EIS) for all MBOs within the cache group:

Policy	Description
On demand	Application logic combined with the cache interval determines when a cache refresh is triggered. The cache is not updated until a request is made of the cache and the cache has expired. If a request is made of the cache and it has expired, there may be a delay responding to the request while the cache is refreshed. You can select Partition by Requester and Device Identity only if using an On demand policy. To enable, select this option from the Cache Group Property Policies tab.
Scheduled	The cache is refreshed when a scheduled task executes, and can be defined by the Unwired Server administrator or by setting the cache interval. Note that load parameters filled from transient personalization keys can not be used with a scheduled cache type.
DCN	The cache never expires. Data refresh is triggered by an EIS Data Change Notification. The cache interval fields are disabled when DCN is selected.

Policy	Description
Online	Used strictly with Mobile Workflow applications where access to real-time EIS data is required, bypassing the Unwired Server cache (CDB). See <i>Online Cache Group Policy</i>

- **Cache interval** – used by both On demand and Scheduled, the cache interval allows you to associate an interval (hour, minute, seconds, and so on) for the cache group. If an application tries to synchronize:
 - Before the cache interval expires – the client application receives the data currently in the Unwired Server cache database (CDB).
 - After the cache interval expires – the CDB is refreshed from the enterprise information system (EIS), and the client synchronizes with the data in the CDB. If the cache interval is set to zero, each device-initiated synchronization refreshes the entire cache prior to synchronization. While this ensures the device receives up-to-date EIS data, it is costly in terms of resources. A Scheduled cache policy does not support a zero interval. Examples include:
 - An On demand policy with a 0 cache interval – each client request for data results in the data being retrieved from the EIS and delivered to the client through the cache, but the cache is immediately invalidated, ensuring the most current EIS data is available to clients.
 - An On demand setting with a two hour cache interval – after each cache refresh, the cache is valid for two hours. Each client request for data is serviced by the cache. When a client requests data at least two hours after the last data refresh, the cache is refreshed from the EIS and the cache interval resets for two hours.
 - A Scheduled setting with a 24 hour cache interval setting – the data is refreshed every 24 hours.
- **Use as default cache group** – identifies this cache group as the project's default. All MBOs automatically belong to the default cache group when created, except for local business objects, which cannot belong to a cache group.

Modifying Cache Group Properties

Modify common and policy properties for cache groups.

Each mobile application project contains a Cache Groups folder that contains all cache groups for the mobile business objects (MBOs) belonging to that project.

1. In WorkSpace Navigator, expand the mobile application project to access the Cache Groups folder.
2. Expand the Cache Groups folder, right-click the cache group you want to view or modify, and select **Properties**.
3. In the left pane, select either the **Common** or **Policies** tab, and view or modify:

Table 49. Cache group properties

Property	Description
Common	<ul style="list-style-type: none"> • Name – name of the cache group. • Description – a description of the cache group. • Use as default cache group – sets this cache group as the default.
Policies	The cache group policy determines the method and timing by which the Unwired Server cache is refreshed from the enterprise information system (EIS) for all MBOs within the cache group.

4. Click **OK** to save changes and exit, or **Cancel** to undo any changes.

Setting the Data Partitioning by User and Device Identity Property

Select Partition by Requester and Device Identity to partition mobile business object (MBO) data by a requester’s identity (user id plus device id). This option ensures that data loaded into the Unwired Server cache (CDB) using an On demand cache policy is accessible only by that requestor.

The requestor identity is derived from the device id and the user from which a request originates, meaning that two users on the same device have different requestor identities, and the same user on different devices has a different requestor identity for each device.

1. Verify that the MBO belongs to a cache group that uses an On demand policy.
2. From the Policies window of the cache group select **Partition by Requester and Device Identity**.

All MBOs within the cache group partitions data by requester and device identity.

Partitioning Data by User and Device Identity Guidelines

Follow these guidelines when configuring MBOs that partition MBO data by a requester’s identity.

Table 50. Guidelines for defining an MBO that partitions data by user identity

Guideline	Description
Setting Partition by Requester and Device Identity	You can select this option when defining a new cache group or modifying the properties of an existing cache group. The default value is unselected.
Local MBOs and structure MBOs (complex data types)	Local and structure MBOs do not support this option.

Guideline	Description
Relationship	<p>If an MBO references a user-partitioned MBO then both MBOs must be user-partitioned. User-partitioned MBOs can reference non-user-partitioned MBOs. For example, there are three MBOS:</p> <ul style="list-style-type: none"> • SalesOrderMBO • LineItemMBO • ProductMBO <p>The LineItemMBO has a foreign key reference to the SalesOrderMBO and a foreign key reference to the ProductMBO.</p> <p>In this case the modeler/developer specifies the SalesOrderMBO is requestor-partitioned, meaning that the LineItemMBO must also be requestor-partitioned (since it contains a foreign key reference to another requestor-partitioned MBO).</p> <p>The modeler does not specify the ProductMBO as requestor-partitioned since doing so needlessly duplicates reference data in the cache.</p>
Cache group	The MBO must be in a cache group that uses the On demand policy.
MBO cache affecting operations	Any cache affecting operation is localized to the partition of the requester.
Data change notification (DCN)	DCN does not support modifying data at the partition level.

Assigning Mobile Business Objects to a Cache Group

Assign mobile business objects (MBOs) to a cache group based on the group's data refresh needs. Once deployed to Unwired Server, the cache policy determines the data refresh policy for each cache group.

All MBOs, except for local business objects, belong to a cache group. By default, new MBOs are assigned to the designated default cache group, which is initially defined to use an On demand policy with a zero cache interval.

1. To move an MBO from one cache group to another from the Workspace Navigator, drag and drop the MBO from the old cache group to the new.
2. To change the default cache group, right-click the cache group to designate as the new default and select **Set as Default**. Alternatively, select **Use as default cache group** in the Properties dialog.

Online Cache Group Policy

Adding MBOs to a cache group that uses the Online policy indicates that the MBOs are to be used only in Workflow forms where access to real-time enterprise information system (EIS) data is required. Data is delivered to the Workflow client based on the `findByParameter` object query definition.

Unwired Platform behavior and usage for MBOs using the Online cache group policy includes:

Unwired WorkSpace

- **Synchronization** – MBO defined synchronization is disabled:
 - The Properties view excludes the Synchronization tab.
 - The Load Parameters tab in the Properties view excludes the Synchronization Parameter column.
 - You cannot add MBOs to any synchronization groups.
- **Cache Update Policies** – Cache update policies are disabled for all operations. All operations use a **No cache update policy**.
- **Load Parameters** – Map **Propagate to attributes** to **Load parameters** from the Load Parameters tab. The generated `findByParameter` object query parameters used by the Workflow form are automatically generated based on these mappings. You cannot map load parameters to synchronization parameters.
- **Object Queries** – one read-only object query named `findByParameter` is automatically generated by Unwired WorkSpace. By default, the return type is "Return multiple objects" and "Create an index" is true. These are the only values you can modify. `findByParameter` query parameters are generated for every load parameter that has a "Propagate-to attribute".

Note: The `findByPrimaryKey` object query and any other user-defined object queries are removed, and all buttons are disabled. An MBO using the Online cache policy must have at least one load parameter-to-Propagate to attribute mapping.

If you modify a "Propagate To attribute" of a load parameter belonging to an MBO using an Online cache group policy, the object query is automatically updated.

- **Deployment** – if you select replication-based synchronization from the Deployment wizard, any MBO using the Online Cache Group policy is unavailable for deployment. Mobile workflow forms support only message-based synchronization.

Mobile Workflow Forms Editor

Specify and bind keys as usual; the input data binding source (`findByParameter`) determines the operation load parameters.

Limitations

Understand the limitations and solutions when implementing Online cache group policy.

- Relationships between MBOs are not supported, unless they are between MBOs generated from the same load operation (also referred to as multiple MBOs from a single load operation or commingled MBOs).
- The requirement of propagating the value of the load parameter to an attribute can be bypassed, and is illustrated with this example:

1. `SELECT id, empName, empDeptId FROM sampledb.dba.emp` (no load parameter)
2. Modify the load definition to be:

```
SELECT id, empName, empDeptId FROM sampledb.dba.emp
WHERE :fakeLP != null
```

3. Create a new int attribute (fakeAttrib) on the MBO and on the Load Parameters tab and propagate fakeLP to the newly added attribute (fakeAttrib).

The object query now takes a parameter even though it is not used. In effect, the load operation is used to select the data and the generated object query returns all the rows returned from the load operation.

Cache Update Policy

Fine-tune device application and Unwired Server performance by defining a cache update policy for mobile business object operations.

Setting a cache update policy for mobile business object (MBO) operations gives you more control of both Unwired Server interactions with the enterprise information system (EIS) to which the MBO is bound, and Unwired Server cache database (CDB) updates. Fine-tuning these interactions and updates improves both Unwired Server and device application performance.

- MBO operations perform specific functions based on their definition:
 - Read operation – the EIS operation used to define and initially populate the CDB (from the EIS) for the MBO.
 - Create, update, delete (CRUD operations) – modify EIS data depending on the definition of the operation. Unwired Server maintains a cache (CDB) of back-end EIS data to provide differential synchronization and to minimize EIS interaction. When an operation is submitted from a device application to the EIS, the cache must be refreshed.

While this type of bulk-fetch and CDB caching are effective in reducing the number of interactions required with the back-end EIS, and work well in some other cases (where MBO data is occasionally updated in the back-end), performance suffers if changes are initiated from Unwired Server (by way of MBO operations), or if changes are frequent. A cache update policy provides alternative methods of updating the cache at finer granularity, which improves performance.

- Cache update policy – determines how the CDB is updated after an operation. You can set the cache update policy for operations, with these exceptions:

Develop

- "Other" operations do not support a cache update policy.
- Delete operations always use the Apply results to cache policy, which cannot be unselected.

The cache update policies from which to choose to associate with MBO CUD operations:

- Invalidate the cache
- Apply results to the cache
- No effect – if a cache update policy is not selected, the operation results are not applied to the cache.

When an MBO operation is called, its cache update policy determines how operation results are applied to the CDB.

Note: Other methods used to update the CDB that are external to MBO operations, and not associated with cache update policies include:

1. EIS-initiated DCN – an HTTP request to Unwired Server, in which the DCN request contains information about the changed data, or the changed data itself.
2. Scheduled data refresh – Unwired Server polls the EIS for changes at specified intervals.
3. MBO cache group – every MBO belongs to a cache group that specifies a cache refresh policy for every MBO in that group. Plan carefully to maximize cache group and cache update policy efficiency. Examples include:
 - A poorly designed MBO might have an operation with a cache update policy that updates only the operation results to the CDB, but the MBO belongs to a cache group with an interval that refreshes the entire MBO on too short a schedule, minimizing the value of the cache update policy.
 - This same MBO properly designed might have a cache group that refreshes the MBO nightly, increasing Unwired Server performance by differing load from peak usage hours.

Setting a Cache Update Policy

Set a cache update policy from the Properties view.

1. From the Mobile Application Diagram, click an operation to access it from the Properties view.
2. Select the **Cache Update Policy** tab.
3. Select the cache update policy. You can select multiple cache update policies for a given operation, depending on the results you want to achieve:

Cache Update Policy	Description
Invalidate the cache	<p>restricts invalidation to only those cache partitions affected by the create or update operation.</p> <p>An operation that uses the invalidate cache policy:</p> <ol style="list-style-type: none"> 1. Performs the create or update operation. For example, insert a new record or update an existing record in the enterprise information system (EIS). 2. Invalidates and refreshes the cache (CDB) for those mobile business object (MBO) partitions affected by the operation, not the entire MBO instance. <p>When defining MBO CUD operations, use cache invalidation sparingly, since it invalidates the entire MBO cache table. Consider using apply results instead, since this updates a single row in the cache rather than invalidating all rows in the cache for an MBO. When the client invokes a CUD operation on an MBO it can be modeled to “invalidate” the cache for that MBO. When an MBO cache is marked as “invalid” the next read request for that MBO refreshes the cache. This places load on the Enterprise Resource as well as the SUP server.</p> <p>Do not use the Invalidate the cache policy for an operation when using the data change notification interface (DCN) to populate the MBO.</p> <hr/> <p>Note: To achieve optimum performance when using cache invalidating operations, result-affecting load parameters should be propagated into the MBO attributes, which are defined in the Attributes > Load Parameters tab.</p>

Cache Update Policy	Description
<p>Apply results to the cache</p>	<p>Modifies the CDB based on the returned result set of the called MBO operation:</p> <ul style="list-style-type: none"> • Create and update operation use this policy by default, which can be modified. • Delete operations use this policy which is always set and cannot be modified. <p>Validation for this cache policy for Web service and SAP MBOs includes:</p> <ul style="list-style-type: none"> • A warning message if you model a create MBO operation that does not have a Primary key definition. • A warning message if you model a create MBO operation if the output record does not contain the "primary-key columns" <p>Operation 'Entity->create()' has 'Apply results to the cache' option set without returning primary key fields.</p> <p>For database MBOs:</p> <ul style="list-style-type: none"> • Unwired WorkSpace must execute a database MBO operation to retrieve the output records. • The output record of the "CREATE" operation "{0}" with "Apply results to the cache" option set does not contain the primary key columns. <p>For JDBC data sources, Unwired WorkSpace does not validate whether an operation with Apply result to the cache policy returns primary key fields as part of the output record. If user want to use this policy option, you need to know whether the definition of the operation can return the primary key column or not.</p> <ul style="list-style-type: none"> • "CREATE" operation "{0}" has "Apply results to the cache" option set without returning primary key fields.
<p>No cache update policy</p>	<p>If you unselect all check boxes for create or update operations, the operation has no effect on the CDB. Data refresh depends on other mechanisms to update the CDB. For example, the cache group to which the MBO belongs.</p>

Cache Update Policy Requirements

Mobile business objects and operations must meet certain requirements to support a cache update policy.

Before you can define a cache update policy for a mobile business object (MBO) operation, the MBO must meet these requirements:

Table 51. Cache update policy requirements

Policy	Requirements
Apply results to the cache	<ul style="list-style-type: none"> • The MBO must be bound to a data source that has one or more Primary key attributes set. • All columns in the record set returned by the operation contains all key attributes.
Invalidate the cache	<p>A partition key identifies the cache partitions affected by the operation. Partition key definition guidelines require that:</p> <ul style="list-style-type: none"> • An input parameter must be mapped to a synchronization key or to an attribute of another MBO to be considered a partition key. • Input parameters that are only mapped to personalization keys are excluded as partition keys, since personalization keys are available only within a client context and not suitable for this cache update policy. If you want to use personalization keys as partition keys, you must map the personalization keys to a synchronization parameter. • A synchronization key must be mapped to an MBO attribute.

Cache Update Policy Examples

These scenarios illustrate how to use cache update policies and mobile business object (MBO) operations to combine and merge in-memory MBO attributes and enterprise information system (EIS) results and apply them to the Unwired Server cache (CDB).

Insert a row into the cache

The merged results represent a new row that is inserted into the cache. Multiple row insertions are not supported.

1. Identify an EIS operation from which you define the MBO create operation that returns all MBO attributes associated with that MBO.
2. Associate the MBO with a Cache Group with an On demand policy and a very large cache interval. This ensures that the operation results and not the cache policy refreshes the cache.
3. Define a create operation and select **Apply results to the cache** as the cache update policy associated with the EIS operation.
4. Deploy the package.
5. Synchronize the client to populate the cache.
6. Create an instance of the MBO on the client device and note the surrogate primary key.
7. Invoke the create operation on the MBO and synchronize. A row with the surrogate primary key that contains all the MBO attributes is created in the cache.
8. Inspect the results on the client device and verify that all the attributes are populated.

Update the cache

The merged results represent a change to an existing cached row. Multiple row updates are not supported.

1. Identify an EIS operation from which you define the MBO update operation. The operation may or may not return attribute values.
2. Associate the MBO with a Cache Group with an On demand policy and a very large cache interval. This ensures that the operation results and not the cache policy refreshes the cache.
3. Define an update operation and select **Apply results to the cache** as the cache update policy associated with the EIS operation.
4. Deploy the package.
5. Populate the EIS (by whatever means) with a single instance of the MBO and note the attribute values of that MBO.
6. Synchronize the client to populate the cache.
7. Update some of the MBO attributes from the client.
8. Invoke the update operation on the MBO and synchronize.
9. Inspect the results on the client device and verify that the attributes have been updated as expected.

Delete a row from the cache

The in-memory attributes provide the surrogate keys for the row(s) that are marked for deletion from the cache. To support relationship composite deletes, multiple rows can be deleted. The EIS results are not required since the surrogate key provided by the in-memory attributes is sufficient to identify the rows.

1. Construct a composite relationship between two MBOs.
2. Identify/construct an EIS operation which deletes the composite (cascading) relationship from the EIS.
3. Associate the MBO with a Cache Group with an On demand policy and a very large cache interval. This ensures that the operation results and not the cache policy refreshes the cache.
4. Define a delete operation and select **Apply results to the cache** as the cache update policy associated with the EIS operation.
5. Deploy the package.
6. Populate the EIS (by whatever means) with a single instance of the MBO hierarchy and note the attributes of the MBOs in the hierarchy.
7. Synchronize the client to populate the cache and provide client access to the MBO hierarchy.
8. Invoke the delete operation on the root MBO instance and synchronize.
9. Inspect and confirm that all the MBO instances associated with the hierarchy have been deleted on the client device.

Notify when the cache changes with a primary key (no cache update policy)

The in-memory attributes provide sufficient information to establish the primaryKey-to-surrogatePrimaryKey linking, creating a row in the cache and marking it as "create pending" until an On demand or Scheduled refresh occurs.

1. Create an MBO and associate it with a cache group that has a default cache policy set to a zero On demand cache interval. When set to zero, each device-initiated synchronization refreshes the entire cache prior to synchronization.
2. Identify/construct an EIS operation which creates an MBO instance and returns nothing.
3. Define a create operation and unselect all cache update policies associated with the EIS operation.
4. Deploy the package.
5. Synchronize the client to populate the cache.
6. Create an instance of the MBO on the client device and take note of the surrogate primary key. Populate the primary key attributes of the MBO.
7. Invoke the create operation on the MBO and synchronize.
8. Inspect the results on the client device and verify that all the attributes are populated.

Notify when the cache changes without a primary key (no cache update policy)

Since the in-memory attributes are not sufficient to establish a primaryKey-to-surrogatePrimaryKey linking, a row is created in the cache and marked as logically deleted.

1. Create an MBO and associate it with a cache group that has a default cache policy set to a zero On demand cache interval. When set to zero, each device-initiated synchronization refreshes the entire cache prior to synchronization.
2. Identify/construct an EIS operation which creates an MBO instance and returns no results.
3. Define a create operation and unselect all cache update policies associated with the EIS operation.
4. Deploy the package.
5. Synchronize the client. An initial synchronization populates the cache.
6. Create an instance of the MBO on the client device and take note of the surrogate primary key. Do not populate the primary key attributes of the MBO.
7. Invoke the create operation on the MBO and synchronize.
8. Inspect the results on the client device and verify that the original instance has been deleted and has been replaced with a new instance with a new surrogate primary key.

Combining cache update policies

In some cases the results returned from the EIS operation are sufficient to establish the surrogatePrimaryKey-to-primaryKey linking, but not sufficient to fully populate the MBO in the cache. In these cases combining the apply results and the invalidate the cache cache update policy settings in a single operation can achieve the desired results.

Develop

1. Identify/construct an EIS operation which creates an MBO instance and returns all the MBO attributes (not including surrogate key fields) associated with that instance, except for one non-primary-key attribute.
2. Associate the MBO with a Cache Group with a Never On demand cache interval.
3. Define a create operation that uses both the **Apply results to the cache** and **Invalidate the cache** cache update policies and associate it with the EIS operation identified above.
4. Deploy the package.
5. Synchronize the client to populate the cache.
6. Create an instance of the MBO on the client device and note the surrogate primary key.
7. Invoke the create operation on the MBO and synchronize. A row with the surrogate primary key noted previously is created in the CDB that contains all the MBO attributes including the one which was not returned from the create operation.
8. Inspect the results on the client device and verify that all the attributes are populated on the client.

Object Queries

Object queries are SQL statements associated with a mobile business object (MBO), that returns a subset of a result set. For example, an object query is used to filter downloaded data to display a single row of a table when triggered.

Define object queries to return a subset of MBO results, either from an MBO deployed to Unwired Server or a local business object.

Table 52. Object query usage

MBO type	Usage
Not bound to a data source (local business object)	The requested data must be available on the mobile device. If not, another (syncable) MBO must trigger a synchronization to download the requested data. The query can then continue to return data from the client's local database.
Bound to a data source	<ol style="list-style-type: none">1. Create the query.2. Call the query from the device application at runtime to display a subset of the results on the device.

MBO type	Usage
Contained in a cache group that uses an Online policy	<p>MBOs that use an Online cache group policy generate one, read-only object query named <code>findByParameter</code>, which is automatically generated by Unwired WorkSpace. <code>findByParameter</code> query parameter(s) are generated for every load parameter that has a Propagate to attribute. The <code>findByPrimaryKey</code> object query and any other user defined object queries are removed for MBOs that use an Online cache policy.</p> <p>By default the return type is Return multiple objects and Create an index is true, and these are the only values you can modify.</p> <p>If you modify a Propagate To attribute of a load parameter belonging to an MBO using an Online cache group policy, the object query is automatically updated.</p>

Generating Object Queries from Primary Key Attributes

An object query is automatically generated for each mobile business object (MBO) attribute identified as a **Primary key**, as well as an additional "composite" object query if there are multiple primary keys.

If an MBO uses an Online cache group policy, these options are disabled. See *Online Cache Group Policy*. Object queries generated from primary key attributes return a single instance (row) of data, which is useful in many applications. For example, you could create a new record and use the object query to return that record to the device application. Depending on the data source to which the MBO is bound, it may require multiple primary keys to uniquely identify an instance of the data. Unwired WorkSpace:

- Generates an object query named `findByPrimaryKey`. This query could be generated from a single primary key attribute, or a composite, if multiple attributes are identified as primary keys:
 - Generates an object query named `findByPrimaryKey` if there is only one primary key attribute.
 - If there are multiple primary keys, generates an object query for every attribute identified as a primary key . For example, primary keys `attA`, `attB`, and `attC` generates queries `findByattA`, `findByattB`, `findByattC`, and `findByPrimaryKey` (which is a composite of the three primary keys).
 - For auto-generated queries, the return type for `findByPrimaryKey` is a single object, with a setting of `isMany = false`, for other auto-generated queries, the return type is a multiple object, with a setting of `isMany = true`.
- Allows you to delete any of the auto-generated object queries. A warning message appears, if you delete the `findByPrimaryKey` object query. Deleting other queries does not generate a warning message. For example, if you:
 - Remove query `findByA`, uncheck **Primary key** for attribute A, then check **Primary key** for attribute A again, the query `findByA` is generated again.

- Remove query `findByPrimaryKey`, then check or uncheck **Primary key** for any attribute, `findByPrimaryKey` is generated again.

The steps for manually generating the `findByPrimaryKey` object query are:

1. Create the MBO from an existing data source. For example, drag-and-drop a table on to the Mobile Application Diagram.
2. From the Properties View Attributes tab, select the **Primary key** attribute(s) that uniquely identify a row.
3. An informational prompt informs you that the name query will be generated. Click **Ok**. Corresponding object queries are generated.
4. Select the Object Queries tab to view the query definition.

The **Primary key** query follows these guidelines:

- The name of the query is `findBy $name$` , where $name$ is the name of the attribute. If the primary consists of only one attribute, there is only a `findByPrimaryKey` method and it returns a single object.

If a primary key consists of multiple attributes:

- `findByPrimaryKey` is still generated, and
- `findByAttribute1`, `findByAttribute2` are generated, but can be deleted. In this case, `Attribute1` and `attribute2` are also primary keys.
- If you change the primary key attribute name, the name of the query automatically changes as well.
- If you change the MBO name, the query definition is automatically updated to reference the newly named MBO.

Manually Defining and Editing Object Queries

Add, edit, and delete object queries, including the `FindAll` object query for an existing mobile business object (MBO) from the Properties view.

Define, or modify, object query properties used to implement the object query.

1. From the Mobile Application Diagram, right-click the MBO for which you are configuring an object query and select **Show Properties View**. Or double-click the MBO title compartment (not on the title) to open the Properties view.
2. In the Properties view, select the **Attributes** tab located on the left side, then select the **Object Queries** tab located on the top.
3. To create a new object query, select **Add**. Follow the wizard instructions to create and add the object query. These properties can be modified later by selecting **Edit**, or by selecting their corresponding check boxes from the Properties view:
 - Name – identifies the query.
 - Comment – an optional description of the object query.

- Parameters – any additional parameters used to modify the object query. Use the **Add**, **Delete/Delete All**, **Up**, and **Down** buttons to create, remove, or rearrange the parameters.
 - Generate – generates a query based on the defined parameters, which can be edited as required. A parameter must map to an available attribute to be generated, but the Generate button is enabled even if there are no parameters. In this case clicking Generate creates a template from which you can fill in your own definition.
 - Query definition – the SQL statement that defines the query.
 - Create an index – generates a composite non-unique index for all mapped attributes.
 - Return type – select either:
 - **Return a single object** – to return a single object (one row).
 - **Return multiple objects** – the default option unless a result set is detected. Indicates the object query can return more than one object.
 - **Return a result set** – Unwired WorkSpace parses the query definition. If a JOIN operation is specified, no matter which return type option is selected, the return type is changed to **Return a result set** and an information dialog notifies you of this change when you click **OK**. You can then select any of these three options.
4. (Optional) Select **Generate FindAll query** – returns a complete list of MBO data values. For example, use the FindAll query in your device application to get all customers from the Customers MBO. The client code could be:

```
CustomerList customers = Customer.FindAll();
```

or

```
List<Customer> customers = Customer.FindAll();
```

This option is selected by default, and also generates the `isGenerateFindAllQuery()` method which returns a boolean, indicating that the FindAll query was generated for normal and local MBOs, and always returns true for structure MBOs. Unselect this option for child MBOs in composite relationships.

5. Click **Finish**.

Object Query Definition Guidelines

Understand how to define object queries.

Support for various compact databases

Since object queries can run on multiple mobile devices that may run different compact databases (UltraLiteJ or SQLite for example), object queries support a subset of UltraLiteJ SQL statements.

Table 53. Object query restrictions

SQL	Restrictions
Select statement	Supported – Order by Unsupported: <ul style="list-style-type: none"> • Bulk and Math functions • Group by • For • Option • Row limitation • As
Input-parameter	Supported – :name
Comparison operators	Supported: <ul style="list-style-type: none"> • Date format is <i>YYYY-MM-DD</i> and contained in single quotes. • Literal strings must be contained in single quotes.
From clause	Supports multiple parts but from only one MBO.

See the UltraLiteJ documentation for more information about the *UltraLite SELECT statement clauses*.

Object queries must use aliases

Define the object query using an alias that references the MBO and attribute names (not table and column names from which they are derived). For example, if you have an MBO named Cust with a cust_id attribute (which is a primary key), defining this object query:

```
SELECT c.* from Cust c WHERE c.cust_id = :cust_id
```

and this parameter:

- Name – cust_id
- Datatype – INT

results in an object query that returns a single row from the Customer table. You must use an alias (c and c.attribute_name) in the query definition or an error occurs during code generation.

General behavior

General object query behavior, including assigning parameters a default value/primary key includes:

- For automatically generated object queries derived from "Primary key" settings, Unwired WorkSpace returns the attributes marked as primary key parameters.
- For manually created object queries, the parameter field allows you to select any of the available attributes, and, when selected, matches the datatype accordingly. You can still manually type in the attribute name.

Additional implied attributes

When manually defining an object query, you can include additional implied attribute columns that allow you to filter a selection based on the implied attributes. Implied attributes and corresponding datatypes, include:

- surrogateKey (long)
- foreign keys (long, if the object has one or more parents). Use the actual name of the foreign key.
- pending (boolean)
- pendingChange (char) – if pending is true, then 'C' (create), 'U' (update), 'D' (delete), 'P' (indicates this row is a parent (source) in a composite relationship for one or more pending child (target) objects, but this row itself has no pending create, update or delete operations). If pending is false, then 'N'.
- disableSubmit(boolean)
- replayCounter

As an example, specifying the foreign key enables selection of children objects of the specified object. The pending flag locates only those objects that have pending changes, and so on.

This example finds the employee's first name, last name, and pending state, identified by the employee's social security number (ssn):

```
select x.emp_fname, x.emp_lname, x.pending from Employee x where
x.ssn_number = :ssn
```

Validation rules

Follow these rules when defining a object query:

- "findBy" is a reserved word for an object query. If you create an operation starting with "findBy", you receive the warning message: Name of operation "{0}" start with 'findBy'. This could cause a name conflict when generating client code.
- Do not duplicate query or operation names.
- Do not use these reserved names as the query name: "pull", "downloadData."
- Do not use an operation name, query name, or attribute name that is the same as MBO name to which they belong.
- While clicking **Generate** generates a valid query, Unwired WorkSpace does not parse or validate the generated query. If you modify the query, be mindful that parameters are not validated until code is generated. For example, this error (mixed case between parameter name and query definition) is not detected until code generation or deployment:
Parameter: Param1 (INT)

Query definition: `SELECT x.* FROM TravelRequest x WHERE x.trvl_id = :param1`

Object Query Indexes

Indexes improve the performance of searches on the indexed attributes (database columns to which the MBO attributes map), by ordering a table's rows based on the values in some or all the attributes. An index locates rows quickly, and permits greater concurrency by limiting the number of database pages accessed. An index also provides a convenient means of enforcing a uniqueness constraint on the rows in a table.

Object query examples that could serve as indexes.

Object query definition	Description
<code>select x.fname, x.lname from Customer x where x.state := :state and x.city := :city</code>	Create one index on the attributes "state" and "city" of the "Customer" MBO.
<code>select x.fname, x.lname from Customer x where x.state := :state or x.city := :city</code>	One query can only generate one index, all the attributes referenced in the query construct a composite index.
<code>select x.fname, x.lname, y.prod_id, y.quantity from Customer x, Sales_order y where x.id := y.id and y.prod_id := :prod_id</code>	No index is generated for join queries.

Creating Object Query Indexes

Add indexes to object queries from the Properties view.

When creating indexes, the order in which you specify the attributes becomes the order in which the attributes appear in the index. Duplicate references to column names in the index definition are not allowed.

1. In the Mobile Application Diagram, right-click the MBO, and select **Show Properties View**. Alternatively, click in the MBO title header to open the associated MBO properties view.
2. In the Properties view, select the **Attributes** tab, then select the **Object Queries** tab.
3. You can either:
 - Add an index to an existing object query – highlight the object query and select **Edit**. You cannot edit the `findByPrimaryKey` object query.
 - Create a new object query and add an index – select **Add**. This requires that a parameter be mapped to an attribute, which allows the index in the query to be generated.
4. Define the query in the **Query Definition** window.
5. Select **Create an index**. Unwired Workspace generates an index based on the query definition.

When to Create an Object Query Index

There is no simple formula to determine whether an index should be created. You must consider the trade-off of the benefits of indexed retrieval versus the maintenance overhead of that index.

Consider these factors in determining if you should create an index:

- **Keys and unique columns** – Unwired WorkSpace automatically creates indexes for `findByPrimaryKey` object queries. You should not create additional indexes on these columns. The exception is composite keys, which can sometimes be enhanced with additional indexes.
- **Frequency of search** – if a particular column is searched frequently, you can achieve performance benefits by creating an index on that column. Creating an index on a column that is rarely searched may not be worthwhile.
- **Size of table** – indexes on relatively large tables with many rows provide greater benefits than indexes on relatively small tables. For example, a table with only 20 rows is unlikely to benefit from an index, since a sequential scan would not take any longer than an index lookup.
- **Number of updates** – an index is updated every time a row is inserted or deleted from the table and every time an indexed column is updated. An index on a column slows the performance of inserts, updates and deletes. A database that is frequently updated should have fewer indexes than one that is read-only.
- **Space considerations** – indexes take up space within the database. If database size is a primary concern, you should create indexes sparingly.
- **Data distribution** – if an index lookup returns too many values, it is more costly than a sequential scan. Also, you should not create an index on a column that has only a few distinct values.
- **Order by** – if you use object queries (also called dynamic queries) with "order by," then you might require indexes for ordering columns to ensure that the database can use an index for ordering, rather than creating a temporary table which can be slow on a mobile device.

FindAll Object Query Guidelines

Understand FindAll query definition guidelines.

By default, a FindAll object query is generated for every MBO and uses these values:

- Name – FindAll
- Parameters – none (A FindAll query without parameters generates a query such as

```
Select * from ...
```

)
- Query Definition – `SELECT x.* FROM {Entity} x`
- Create an index – false

Develop

- Return Type – Multiple Objects (accepts {Single Object, Multiple Objects, Result Set})

Unwired WorkSpace validates the query name and disallows it if it is a reserved word or restricted in some way, including:

- All MBO operation names
- Standard generated getter/setter methods (Get{Attribute}/Set{Attribute})
- Standard generated relationships (Get{Relationship}/Set{Relationship}/Get{Relationship}Size)
- Standard methods (find, create, delete, update, save, refresh)
- GetMetaData
- GetClassMetaData
- Anything that starts with a underscore (e.g. _init)
- IsDeleted, IsDirty, and so on
- KeyToString
- Equals
- GetHashCode
- xxxFilterBy(...)
- Bind
- Load
- Find
- Find_os
- Merge
- CopyAll
- CreateBySQL
- GetDownloadState
- Set/GetOriginalState
- CancelPending
- CancelPendingOperations
- SubmitPendingOperations
- Internal_{xxx}
- SubmitPending
- FromJSON/ToJSON{List}
- GetSize
- FindWithQuery
- Subscribe_{xxxx}/Unsubscribe_{xxx}
- GetPendingObjects
- GetSynchronizationParameters
- GetLogRecords
- LastOperation
- GetCallbackHandler

Result Set Filters

A result set filter is a custom Java class an experienced developer writes in order to specifically manipulate the rows or columns of data returned from a read operation for an MBO.

To write a filter, developers must have previous experience with Java programming — particularly with the reference implementations for `javax.sql.RowSet`, which is used to implement the filter interface and described in the *JDBC RowSet Implementations Tutorial*.

Note: Result set filters depend on the `sup-ds.jar` file, located in the `com.sybase.uep.tooling.api/lib` subdirectory. For example, `C:\Sybase\UnwiredPlatform\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\eclipse\plugins\com.sybase.uep.tooling.api_<version_and_timestamp>\lib`.

When a read operation returns data that does not completely suit the business requirements for your MBO, you can write and add a filter to the MBO to customize the data into the form you need. Developers can write a filter to add, delete, or change columns as well as to add and delete rows.

You can chain multiple filters together. Multiple filters are processed in the order they are added, each applying an incremental change to the data. Consequently, Sybase recommends that you always preview the results, taking note that the MBO has a different set of attributes than it would have had directly from the read operation. You can map and use the altered attributes in the same way you would a regular attribute from an unfiltered read operation.

Note: The filter interfaces are defined in terms of `java.sql.ResultSet` and `java.sql.ResultSetMetaData`, but these standard JDBC interfaces tend to be read-only implementations. To change data, use a `CachedRowSetImpl` object instead. This object implements `ResultSet` but also allows you to modify row data.

Example: a simple SELECT statement filter

Suppose you have an MBO based on this query that returns customer information, and you do not want first name and last name divided between two columns (`fname` and `lname`) :

```
SELECT * FROM sampledb.customer
```

Instead, write a filter that replaces these columns with a single concatenated "commonName" column.

Note: You could also implement the above example with a more advanced SQL statement with additional computation in the MBO definition:

```
SELECT id, commonName=fname+' '+lname, address, city, state, zip, phone, company_name FROM customer
```

Example: two separate data sources filter

Suppose you have customer data in two data sources: basic customer information is in an SAP® repository, and more complete details are contained in another database on your network, for example, SQL Anywhere®. You can use a result set filter to combine the SAP customer data with detailed customer data from the database, so that the MBO displays a complete set of information in a single view. You can accomplish this by:

1. Creating a filter for the SAP backend and add it to an SAP MBO.
2. Add a JDBC connection for the SQL Anywhere backend in the filter, then use the SQL Anywhere data to filter the SAP result.
3. Validate the results are what you expect upon completion. When you synchronize the SAP MBO, you should see data from both SAP backend and SQL Anywhere backend.

See also

- *Filtering Result Sets Returned by Attributes* on page 212
- *Writing a Custom Result Set Filter* on page 214
- *Adding Result Set Filters* on page 212
- *Validating Result Set Filter Performance* on page 216
- *Previewing Mobile Business Object Attributes* on page 155
- *Binding an SAP Data Source to a Mobile Business Object* on page 96
- *Binding a Web Service Data Source to a Mobile Business Object* on page 100
- *Binding a Database Data Source to a Mobile Business Object* on page 87

Filtering Result Sets Returned by Attributes

Configure a result set filter when you define the attributes used to read data from your data source and create a result set (that is, a set of rows and meta-information). You can configure attributes when you create a mobile business object or when you edit the object's properties.

Before adding a result set filter to an MBO from Unwired WorkSpace, perform these tasks:

See also

- *Result Set Filters* on page 211

Adding Result Set Filters

Choose and add a result set filter when you edit Attribute properties for a mobile business object from the **Definitions** tab. You can also configure result set filters when you create an object. You can choose a predefined or a custom filter, if you have created one.

1. (Optional) If you have not yet created the classes, then in the **Resultset Filters** area of the **New Attributes** dialog, click **Create** to run the New Java Class wizard.
2. If prompted, add a Java nature.
 - (Recommended) Click **Yes** to add a Java nature. In Eclipse, a Java nature adds Java-specific behavior to projects. A Java nature is recommended because you are creating a

new Java project and adding a Java class to it. By default an Unwired Platform project does not include all the required behaviors for Java development. Clicking **Yes** automates this process.

If you clicked **Yes**, a wizard appears allowing you to enter the java package name and java class name. Click **Finish** in the wizard to compile the java skeleton source file and add the skeleton java filter class to the MBO. If you are adding the filter from the **Definition** tab of Attribute Properties view, you are prompted to refresh the data source definition. Choose **Yes** in response to this prompt. Other wise, choose **No** and only refresh after the logic has been put implemented and the java class has been built.

Note: Only the wizard generates a skeleton java source file.

- Click **No** if you do not want to add the Java nature to the selected Mobile Application Project.

Note: When a mobile application project is exported as an archive file, the `Filters` folder is not archived in the zip file if the `Filters` folder is empty. In these cases, you must manually create the `Filters` folder after you import the project into Unwired WorkSpace.

3. Implement the new class by writing the real implementation on top of the skeleton created as documented in the next topic, *Writing a Custom Result Set Filter*.
4. Add the filter you require to your mobile business object. In the **ResultSet Filters** area of the **New Attributes** dialog, click **Add**. If you are creating an MBO, you can also perform similar action with the corresponding wizard.
5. Select an existing filter class that you imported into Unwired WorkSpace.
Only valid filters (that is, filters implemented with `com.sybase.uep.eis.ResultFilter`) appear in the filters table.
6. (Optional) Repeat steps 3 and 4 to add more filters to the mobile object.
7. (Optional) If the filters are not in the order you require, reorder them with either the **Up** or **Down** button. The order of multiple filters affects the actual Result set and metadata.
8. Test and preview the Result of your filter settings:
 - a) To reuse input values you have already saved for previous previews, select **Existing Configuration**. Otherwise, load defaults, or create a new set of input values expressly for this preview instance.
 - b) To run the preview, click **Preview**.

If the data filters successfully, `Execution Succeeded` appears at the top of the Preview dialog and data appears in the **Preview Result** window.

See also

- *Result Set Filters* on page 211

Writing a Custom Result Set Filter

Write a custom result set filter to define specific application processing logic. Save the compiled Java class file to location that is accessible from Unwired Workspace.

In the custom filter, configure attribute properties so that the returned record set can be better consumed by the device client application. Sometimes, a result set returned from a data source requires unique processing; a custom filter can perform that function before the information is downloaded to the client.

Data in the cache is shared by all clients. If you need to identify data in the cache to a specific client, you must define a primary key attribute that identifies the client (such as `remote_id` or `username`).

1. (Required) Create a record set filter class that implements the `com.sybase.uep.eis.ResultSetFilter` interface.

This interface defines how a custom filter for the data is called.

For example, this code fragment sets the package name and imports the required classes:

```
package com.mycompany.myname;
import java.sql.ResultSet;
import java.util.Map;
```

2. (Recommended) Implement the `com.sybase.uep.eis.ResultSetFilterMetaData` interface as well as the `com.sybase.uep.eis.ResultSetFilter` interface on your filter class.

If you choose not to implement this interface, Unwired Workspace will have to execute a chain of mobile business object operations and filters and fetch real data before you can see the actual output column names and their datatypes. By first implementing these interfaces, the operation does not need to be executed first. Instead, the `getMetaData` obtains the necessary column or data type information.

This example sets the package name but uses a different combination of classes than in the example for step 1:

```
package com.mycompany.myname;
import java.sql.ResultSetMetaData;
import java.util.Map;
```

3. Call the appropriate method, which depends on the interfaces you implement.

`ResultSetFilter` filters the data in the first option documented in step 1. Each filter defines a distinct set of arguments. Therefore, use only the arguments with the appropriate filter that defines these arguments in `getArguments()`, rather than use all filters and data source operations.

The result set passed in contains the grid data, which should be considered read-only—do not use operations that change or transform data.

```
public interface ResultSetFilter {
    ResultSet filter(ResultSet in, Map<String, Object> arguments)
    throws
        Exception;
    Map <String, Class> getArguments();
}
```

Next, use `ResultSetFilterMetaData` to format the data from step 1. Use this interface to avoid executing an extraneous data source operation to generate a sample data set.

```
public interface ResultSetFilterMetaData {
    ResultSetMetaData getMetaData(ResultSetMetaData in, Map<String,
    Object> arguments) throws Exception;
}
```

Note: If the filter returns different columns depending on the argument values supplied, the filter may not work reliably. Ensure that any arguments that affect metadata have constant values in the final mobile business object definition, so the schema does not dynamically change.

4. Implement the class you have created, defining any custom processing logic.
5. Save the classes to an accessible Unwired WorkSpace location. This allows you to select the class, when you configure result set filters for your mobile business object.
6. In Unwired WorkSpace, refresh configured MBO attributes, to see the result.

MBO load operations can take parameters on the enterprise information system (EIS) side. These load parameters are defined from Unwired WorkSpace as you create the MBO. For example, defining an MBO as:

```
SELECT * FROM customer WHERE region = :region
```

results in a load parameter named "region".

As an example, if you want a filter that combines `fname` and `lname` into `commonName`, add `MyCommonNameFilter` to the MBO. When `MyCommonNameFilter.filter()` is called, the "arguments" input value to this method is a `Map<String, Object>` that has an entry with the key "region". Your filter may or may not care about this parameter (it is the backed database that requires the value of `region` to execute the query). But your filter may need some other information to work properly, for example the remote user's zipcode. The `ResultSetFilter` interface includes

```
java.util.Map<java.lang.String, java.lang.Class>
```

`getArguments()` that you must implement. In order to arrange for the remote user's zipcode (as a `String`) to be provided to the filter, write some custom code in the body of the `getArguments` method, for example:

```
public Map<String, Object> getArguments {
    HashMap<String, Class> myArgs = new HashMap<String, Class>();
    myArgs.put("zipcode", java.lang.String.class);
    return myArgs;
}
```

This informs Unwired WorkSpace that the "zipcode" parameter is required, and is of type String. Unwired WorkSpace automatically adds the parameter for the load operation, so this MBO now has two (region and zipcode). Your filter gets them both when its `filter()` method is called, but can ignore region if it wants.

See also

- *Result Set Filters* on page 211

Viewing the Filter Class Output Stream

You can set debugging options to view the output stream when using `System.out.println` in filter classes to help you debug your filter classes.

1. Go to `<SUP Installation Root>\Eclipse` and open the `UnwiredWorkSpace.bat` file with a text editor.
2. If the `-vm` options is specified, replace `javaw.exe` with `java.exe`.

Note: The **javaw.exe** command is the same as the **java.exe** command except that with **javaw.exe**, there is no associated console window.

3. After the line `%ECLIPSE_ROOT%\eclipse.exe" %ADDITIONAL_ARGS%` add either `-debug` or `-consoleLog`.
4. Start Eclipse.
A Java console window appears with the output.
5. View the debugging statement through `System.out.println` on the server side.
On the Unwired Server side, all debug statements are saved to the `ml.log` file.

Validating Result Set Filter Performance

After you deploy the filters to Unwired Server, synchronize data and ensure that filters are performing as you expect.

1. Confirm that the columns appear correctly after the filter has been added to the mobile business object.
 - a) Refresh the object.
 - b) In the Properties view, select the **Attribute Mapping** tab.
 - c) Verify that columns are correctly listed in the **Map to** column.
2. From the device client or the device simulator, open the mobile object, and check that the new column appears.
3. Synchronize the object from the device client or simulator.
4. Troubleshoot filters if issues arise:
 - During synchronization, all `System.out` statements are printed to the Unwired Server log.

- If you started Unwired WorkSpace with the `-consoleLog` in `java.exe`, `System.out` statements are also printed to the console window.

See also

- *Result Set Filters* on page 211

Packaging and Deploying Mobile Business Objects

Create and deploy a deployment package to the Unwired Server that contains your mobile business objects (including role mappings, server connection mappings, and other mobile business object related artifacts). Optionally create a deployment profile that allows you to manage multiple deployment packages.

You can either:

- Deploy a Mobile Application project – only the selected project is deployed. This method allows you to save the project in a deployment profile so that it can be reused.
- Create the mobile deployment package from the Deployment Package wizard – allows you to select any number of projects to add to the mobile deployment package. This method provides maximum flexibility allowing you to reuse the deployment profile, bundle multiple deployment packages, deploy to multiple Unwired Servers, and so on.

See also

- *Developing a Mobile Business Object* on page 69
- *Working with Mobile Business Objects* on page 118
- *Mobile Business Objects* on page 70

Deploying a Mobile Application Project

Deploy a Mobile Application project directly to an Unwired Server, and optionally create a reusable deployment profile.

To avoid errors or inconsistent behavior, client applications must be regenerated whenever a package has been redeployed. Restarting the client application is not sufficient to reset the client for a package that has been redeployed.

1. Right-click the Mobile Application project and select **Deploy Project**.

Alternatively, you can launch the deployment wizard, which automatically sets the Unwired Server portion of the wizard, by dragging a Mobile Application project folder from Workspace Navigator and dropping it on the Unwired Server in Enterprise Explorer to which you are deploying.

Note: As an option, you can press F9 when your cursor is in the Mobile Application Diagram to launch the Deployment wizard for the corresponding project. If a deployment

profile exists for the project, F9 performs quick deployment of the project according to the profile.

2. Enter the deployment mode, target version, Package namespace, and click **Next**.
3. Select the contents to be deployed and click **Next**.
4. Create or add required JAR files for MBOs that use Resultset Filters or Custom Result Checkers and click **Next**.
5. Select a target server for the project and click **Next**. (Optional) If no Unwired Server connection exists, click **Create** and define a connection profile for one to which you can connect and deploy the project.

If a server connection has been open a long time without activity (a day or two for example), the connection may time out without your knowledge. While you may still be able to select the target server, deployment fails. Verify you have a working connection by pinging the server. If necessary, reconnect to the Unwired Server to which you are deploying from Enterprise Explorer.

6. Map connection profile to server connections – You must map design-time connection profiles to server-side (run-time) enterprise information system (EIS) data sources referenced by the MBOs in the project. Deployment fails if the EIS data sources are not running and available to connect to. To map the connection profile to a server connection, select the connection profile from the list of available connection profiles then select the corresponding server connection to which it maps.

Contact the system administrator in cases where your development environment permits access to systems that the Unwired Server prohibits.

Note: You can also modify server connection properties (Web service connections only).

7. If a logical role is defined in your MBO, map logical roles to physical roles. If there are no logical roles defined, this page is skipped. Click **Next**.
8. (Optional) Specify the name and location for the new deployment profile:
 - Save the deployment settings as a deployment profile – if you do not save your settings to a deployment profile, they are lost when you exit the Deploy wizard.
 - Enter or select the parent folder – by default, Deployment is the folder in which the deployment profile is saved.
 - File name – the name of this deployment profile. The deployment profile is assigned a `.deploy` extension.
9. Click **Finish** to deploy the project to the Unwired Server's Packages folder.

Creating a Mobile Deployment Package

Create a mobile deployment package that contains the mobile business objects (MBOs) to be deployed to an Unwired Server.

1. Launch the New Mobile Deployment Package wizard.

From	Action
WorkSpace Navigator	Right-click the Mobile Application project and select Create Mobile Deployment Package .
File menu	Select File > New > Mobile Deployment Package .

2. Enter the project folder and file name for the mobile deployment package and click **Next** (the file name is used as the default 'package name' in the next step).

By default, when the mobile deployment package is created, it is given a .pkg extension. For example, if the file name is test_package, the mobile deployment package file name is test_package.pkg.

3. Enter the name of the mobile deployment package (maximum 64 characters) that is deployed to the Packages folder of the target Unwired Server, and an optional description and click **Next**.

The value of the Package name field entered here is not the package name on the Unwired Server.

4. Select the MBOs to be included in the package and click **Finish** or **Next**:

- Select the project level to select all of the MBOs from the project.
- Select one or more individual MBOs.

Note: If any MBO contains an error, an error icon displays next to that MBO. You can still include the MBO in the package.

If any MBOs in the package have dependencies, those dependencies are included, and display in the **Dependencies** section.

5. If any of the MBOs include JAR files, for example, if you have created custom result checkers or result set filter classes, the Package User-defined Classes page prompts you to deploy them to Unwired Server. You must deploy your custom classes to the server to use them.
6. Review the summary and click **Finish**.

The mobile deployment package (<file name>.pkg) appears in the WorkSpace Navigator and the Mobile Deployment Package editor opens.

Configuring a Mobile Deployment Package

Use the Mobile Deployment Package editor to configure or modify the contents of the mobile deployment package.

1. Open the Mobile Deployment Package editor:

From	Action
A new mobile deployment package	The editor automatically opens after creating a mobile deployment package.

From	Action
An existing mobile deployment package	Right-click the mobile deployment package and select Open .

2. Select the **Configuration** tab to configure or modify these mobile deployment package settings:

Table 54. Mobile deployment package configuration

Screen	Description
General Information	<ul style="list-style-type: none"> • Package name – identifies the package to be deployed to the Unwired Server. • Description – (optional) a description of the package.
Contents	Lists MBOs included in the package. You can add or remove MBOs. Any dependencies are automatically updated.
Package User-defined classes	If any of the MBOs include custom result checkers or result set filter classes, the Package User-defined Classes screen prompts you to deploy them to Unwired Server.
Roles	A read-only field that displays the roles assigned to the MBOs or operations. You must modify roles at the MBO or operation level. You can map logical roles to physical roles when you deploy the package or create a deployment profile.
Dependencies	A read-only field that displays any MBO dependencies.
Data Sources	A read-only field that displays the data source to which the MBOs are bound. You can bind or rebind to a server connection when you deploy the package or create a deployment profile.

3. Select **File > Save**.

Building a Mobile Deployment Package

Construct a JAR file that contains XML files that contain the metadata of the mobile deployment package.

Prerequisites

A mobile deployment package must already exist before you can build it.

Task

1. Expand the Deployment subfolder of the Mobile Application project.
2. Right-click the deployment package and select either **Build Package (full)** or **Build Package (incrementally)** to build the JAR file.

Deploying Mobile Deployment Packages while Creating a Deployment Profile

Deploy a mobile deployment package and optionally create a deployment profile to store a deployment scenario that can be executed multiple times.

Prerequisites

A connection profile to at least one Unwired Server must be available before you deploy mobile deployment packages to a server.

Task

Following these instructions you can:

- Deploy a mobile deployment package without creating a deployment profile, Or
 - Deploy a mobile deployment package and create a reusable deployment profile.
- If you deploy a mobile deployment package without creating a deployment profile, the settings are retained and used during subsequent deployments. For example, deployment mode (Replication based or Message based).
1. Launch the Mobile Deployment Package wizard by Right-clicking the mobile deployment package (the file with the .pkg extension) and selecting **Deploy Package**.
 2. Enter the deployment mode for the package, a target version (including the package namespace), select whether the package uses message-based or replication-based synchronization, and click **Next**.
 3. Select a target Unwired server for the package, connect to it, select the domain and security configuration (if not using the default), and click **Next**.
 4. (Optional) Map connection profile to server connection. Allows you to map connection profiles used for development to an appropriate server-side connection. For example, your development environment might permit access to certain systems that the Unwired Server prohibits. To map the connection profile to a server connection select the connection profile from the list of available connection profiles then select the corresponding server connection to which it maps.
 5. Map logical roles to physical roles. Click **Next**.
 6. (Optional) Specify the name and location for the new deployment profile:
 - Select **Save the deployment settings as a deployment profile** – if you do not save your settings to a deployment profile, they are lost when you exit the Deploy Package wizard.
 - Enter or select the parent folder – by default, Deployment is the folder in which the deployment profile is saved.

- File name – the name of this deployment profile. The deployment profile is assigned a `.deploy` extension.
7. Click **Finish** to deploy the mobile deployment package to the Unwired server's Packages folder, and save the information in a deployment profile.

Configuring a Deployment Profile

Update existing deployment profiles, bundle multiple mobile deployment packages, and deploy contents to multiple Unwired servers.

1. Expand the Mobile Application project of interest.
2. Right-click the mobile deployment profile identified by the `.profile` extension, and click **Open**.
3. Use the Target Mapping section of the Configuration page to edit the configuration and target Unwired servers for a package.
4. Use the Package Description section of the Configuration page to edit the package description.
5. Use the Servers section of the Configuration page to edit the servers for a package.

Editing General Deployment Profile Information

Review and edit general deployment profile information from the Overview page of the Deployment Profile editor.

1. Select the **Overview** tab from the Deployment Profile editor.
2. Review or edit the information on this page.

Table 55. Overview page

Option	Description
General Information	Basic information about the deployment profile, including the name, and description. You can edit both the name and description.
Servers	(Read only) The Unwired servers that are targeted for deployment using this deployment profile.
Packages	(Read only) The packages included in this deployment profile.

Adding a Package to a Deployment Profile

The deployment profile can contain multiple packages, each with multiple target servers and settings.

1. In the Deployment Profile editor, select the Configuration tab.
2. In the Target Mapping section, click **Add Package**.
3. Select one or more packages to include in this deployment profile.

4. Click **OK**.

Removing a Package from a Deployment Profile

You can remove a package from a deployment profile when it is no longer needed.

1. In the Deployment Profile editor, select the Configuration tab.
2. In the Target Mapping section, select the package you want to remove from the deployment profile.
3. Click **Remove Package**.

All custom configurations for the specific servers are also removed.

Packaging Jars for Deployment

If you have created JAR files for custom result checkers or result set filters during mobile application development, include them when deploying the mobile application project to Unwired Server.

1. During deployment of a Mobile Application project or Mobile Deployment package, the **Package Jars** dialog allows you to add JAR files.
2. Select the **Add JAR** to add a JAR file from a Mobile Application Project or **Add external JAR** to add a JAR file from the file system. Select **Delete** or **Delete all** to remove JAR files from this deployment.

Note: By default, all result checker or result set filter classes used by selected MBOs to be deployed are checked in the wizard and the default JAR location is the root folder of the current SUP project. Class files with compile errors are excluded.

3. Click **Next** when you have included all JAR files.

Modifying Target Servers

Modify target Unwired Servers to which mobile deployment packages are deployed. A deployment profile can contain multiple mobile deployment packages, each with multiple target Unwired Servers as their destination.

Adding a Target Server to a Deployment Package

You can select multiple target Unwired Servers to which a package within a deployment profile is deployed.

1. In the Deployment Profile editor, select the Configuration tab.
2. Select a package in the Target Mapping section.
3. Click **Add Target**.
4. Select a server to add as a target for this package.
5. Click **OK**.

Changing a Target Server for a Deployment Package

You can change the target Unwired Server for a deployment package contained in a deployment profile.

1. In the Deployment Profile editor, select the Configuration tab.
2. Expand the deployment package in the Target Mapping section and select the server you want to change.
3. Click **Change Target**, located on the right.
4. Select a different server to use as a target for this package.
5. Click **OK**.

The existing configuration settings used for the original target server are used for the new target server.

Removing a Target Server from a Deployment Package

You can remove a target Unwired Server from a deployment package contained in a deployment profile.

1. In the Deployment Profile editor, select the Configuration tab.
2. Expand the package in the Target Mapping section and select the server you want to remove
3. Click **Remove Target**.
Any custom configuration settings for the target are also removed.

Configuring a Mobile Deployment Package for the Target Unwired Server

Define the deployment mode, server connection mappings, and role mappings for the mobile deployment package based on the target server's environment.

For each Unwired Server to which you deploy a mobile deployment package, you can modify package settings such as server connection mappings (data source to server connection), role mappings(logical to physical), and deployment modes specific for that server's environment.

Modify settings for:

- A mobile deployment package
- A Mobile Application project
- A deployment profile – from the Deployment Profile editor's Configuration tab, expand the package and select the server you are configuring the package for and select **Configure Package**.

Deployment Mode and Target Version

You can set the version and modes in which a Mobile Application project or mobile deployment package are deployed to the target Unwired server.

Table 56. Deployment modes and target version

Option	Description
Update	Updates the target package with an updated version.
No Overwrite	Deploys the source package only if there are no objects in the target package that have the same name as any of the objects being deployed
Replace	Replaces any of the target objects with those in the package.
Verify	Does not deploy the package but reports what, if any, errors would occur if you were to deploy the package using Update mode.
Target Version	Determines the version of the target package to which the package is to be deployed. By default, the current project version is used. You can enter a different version, if appropriate. The version consists of two numbers. For example, 1.0.
Package namespace	The location in which the deployed unit resides. The default value is the Mobile Application project name.
Replication based or Message based	<ul style="list-style-type: none"> • Replication-based – all data within the package is replicated in the Unwired Server cache (CDB). • Message-based – used within a Mobile Workflow package, which uses push synchronization through a dedicated channel. <hr/> <p>Note: This setting is now ignored:</p> <ul style="list-style-type: none"> • Unwired WorkSpace – the package type (MBS and RBS) in the deployment wizard is no longer relevant. • Sybase Control Center – all packages display as Unified, and there is no package option in the deployment page.

Target Server Properties

Select Unwired Server specific domain properties when deploying a project to the server to which you are connected.

Table 57. Target server properties

Property	Description
Domain	The domain to which deployment occurs. Separating projects into domains allows you to share various Unwired Server resources between customers while keeping their data separate.

Property	Description
Security configuration	Each domain supports a variety of security configurations, which allows each customer to have a security configuration that meets their needs. Once configured on Unwired Server, select the security configuration from the drop-down list.

Configuring Server Connection Mappings

Map design-time data source connection profiles to server connections supported by the Unwired Server.

When developing mobile applications, you bind the mobile business objects to the data sources available to the Unwired WorkSpace and available from the Enterprise Explorer. When you deploy Mobile Application projects or Mobile Deployment packages to the Unwired Server, you must change data sources from connection profiles to server connections available to the Unwired Server .

1. During deployment of a Mobile Application project or Mobile Deployment package, the **Server Connection Mapping** dialog allows you to change data sources.

If Unwired Server has a server connection name that matches that of the connection profile, it is selected by default. Otherwise, the server connection mapping is left empty.

2. Map each design-time connection profile to a corresponding server connection by selecting a server connection from the drop-down list, or select **New Server Connection** to create a new server connection.

The **Server connection properties** field displays information about the selected server connection.

For Web service MBOs the mapping dialog automatically loads the design-time SOAP address in the address field, which you can change. Be sure the address is a SOAP address location in the published WSDL, not the WSDL address itself.

3. Click **Next/Finish** when you have mapped all connection profiles to server connections.

Configuring Role Mappings

Map design-time logical roles to runtime physical roles.

When developing mobile business objects you can create and assign logical roles to mobile business objects and operations. When you deploy mobile business objects to Unwired Server, you can map these logical roles to physical roles that are valid on Unwired Server.

When mapping logical roles to physical roles, the wizard:

- The wizard displays roles and mappings of the server to which you are deploying. When creating a deployment profile, only user configured role mappings display.

- Allows you to map a logical role to None (authorization always fails if a role is mapped to None, which disables access to the operations protected by this role).
 - Allows you to map a logical role to Auto (automatically passes through the mapping if the role exists on Unwired Server and the logical and physical role names match).
1. During deployment of a Mobile Application project or mobile deployment package, follow wizard instructions to map logical roles to physical roles. Change the mapping for a logical role, if required:
 - To change the state to either None or Auto, click the cell adjacent to the logical role and click one of these options.
 - To change the role mapping itself, click the cell adjacent to the logical role and choose **(Map Role)**. This command displays the **Role Mappings** dialog that allows you to manually set the logical and physical role mappings you require.
 2. Selecting **(Map Role)** displays the Role Mappings dialog with the name of the physical roles to which you map in the text area of the dialog. When saved, the mapped physical roles display as a comma separated list.
Click **OK** after you map the role. You must map at least one physical role to enable **OK**.

Table 58. Role mappings dialog

Option	Description
Role mappings	For this deployment. For example, if a mobile business object operation has the role design_role assigned to it, and the Unwired Server to which you are deploying has a physical role named runtime_role, if you map design_role to runtime_role, any mobile business objects and operations assigned the logical role design_role are assigned runtime_role upon deployment and assume the characteristics of runtime_role.
Available physical roles	Displays the physical roles that are available on the Unwired Server to which you are deploying.
Assigned physical roles	Displays the physical roles that are assigned to the package that you are deploying.

Option	Description
Physical role availability	<p>Modify physical role availability:</p> <ul style="list-style-type: none"> • Add – add a physical role to the list of assigned physical roles, allowing that physical role to be mapped to a logical role. • Input – allows you to enter a known physical role even if it is not listed. • Remove – makes the selected physical role unavailable for mapping. • Add All – makes all physical roles located on the Unwired Server available for mapping. • Remove All – makes all physical roles located on the Unwired Server unavailable for mapping. <hr/> <p>Note: Removing physical roles only removes their availability to be mapped to logical roles. It does not remove them from the Unwired Server.</p>

Deleting a Mobile Deployment Package

Delete a mobile deployment package from the Deployment folder.

1. Open the Mobile Application project folder that contains the package you want to delete.
2. Right-click the package (a package icon at the same level as the other folders identified with a `.pkg` extension) and click **Delete**.

Deleting a Deployment Profile

Delete a deployment profile from the Deployment folder.

1. Open the Mobile Application project folder that contains the deployment profile you want to delete.
2. Right-click the deployment profile (a profile icon at the same level as the folders identified with a `.profile` extension) and click **Delete**.

Deleting a deployment profile does not delete the deployment packages that it contains.

Viewing Deployment Errors

View deployment errors by selecting the Show Deployment Status option from the deployment error dialog.

1. Deploy a project or deployment package.
2. If an error occurs during deployment, select the **Show Deployment Status** button on the Problem dialog to display additional error information.

You can also view errors from the Error view, by selecting **Window > ShowView > Other > Error Log**.

Managing Deployed Packages and Mobile Business Objects

Once deployed, manage deployment packages and mobile applications from the administration console. However, you can perform limited administration from the Mobile Development perspective.

Managing a Deployed Package

From the Enterprise Explorer you can refresh or remove deployment packages from an Unwired Server.

1. From the Enterprise Explorer, expand the Unwired Servers folder, Unwired Server, Domain folder, specific domain, then the Packages folder.
2. Right-click the package and select:
 - Enable – allows client access to the mobile business objects contained in the package.
 - Disable – denies client access to the mobile business objects within the package.
 - Delete – deletes the package and its contents from the Unwired Server.
 - Refresh – reflects any changes made to the package and its contents.

Managing a Deployed Mobile Business Object

From the Enterprise Explorer you can refresh, enable, disable, and view the properties of a mobile application's mobile business object deployed to an Unwired Server.

1. From the Enterprise Explorer, expand the Unwired Servers folder, Domains, domain of interest, and expand the package.
2. Right-click the Mobile Business objects folder and select **Refresh** – reflects any changes made to the mobile business objects in the folder.
3. Expand the Mobile Business objects folder, right-click the MBO of interest and select **Properties**. You can view (but not modify) these properties:
 - Common properties
 - Data Sources
 - mobile business object operations and their properties
 - Role mappings

Note: You cannot delete individual mobile business objects from the Unwired Server. Instead you must either delete the package and redeploy an updated package, or deploy a new package.

Managing Deployed Personalization Keys

From the Enterprise Explorer you can refresh and view the properties of a personalization key deployed to Unwired Server.

1. From the Enterprise Explorer, expand the Unwired Servers folder, Domains, domain of interest, and expand the package.

2. Right-click the Personalization keys folder and select **Refresh** – reflects any changes made to the personalization keys contained in the folder.
3. Expand the Personalization keys folder, right-click the personalization key of interest and select **Properties**. You can view (but not modify) these common properties:
 - Name
 - Type
 - Protected
 - Default value(s)
 - Description

Note: You cannot delete individual personalization keys from Unwired Server. Instead you must either delete the package and redeploy an updated package, or deploy a new package.

Develop a Device Application

Develop custom device applications from your data source for one or more device platforms.

Once you have developed your mobile business objects (MBOs), you have these options for creating custom device applications:

- In the Mobile Application Diagram, invoke the Generate Code wizard to generate code in Java (BlackBerry devices), C# (Windows Mobile devices), or Objective C (iOS devices), which can be used to call the mobile business object operations. This code can then be imported into an integrated development environment (IDE) of your choice to create the device application. You can also customize the generated object API code in the applicable development environment, where you can then test by deploying to an emulator or device. For example, if you are using Visual Studio to develop custom applications for Windows Mobile devices, generate the C# code from the data source in Eclipse, then customize the client object API code for the device application in Visual Studio. See the *Developer Reference Guide* for your platform (BlackBerry, Windows Mobile, or iOS) for more information.
- Use the Mobile Workflow Forms Editor in Eclipse to generate the code for a mobile workflow package. See *Sybase Unwired WorkSpace – Developing a Mobile Workflow Package* for more information.

Generating Object API Code

Generate object API code containing mobile business object (MBO) references, which allows you to use APIs to develop device applications for various mobile devices.

Prerequisites

Before generating device client code, develop the MBOs that will be referenced in the device applications you are developing. A mobile application project must contain at least one non-

online MBO. You must have an active connection to the data sources to which the MBOs are bound.

Task

1. Launch the **Code Generation** wizard.

From	Action
The Mobile Application Diagram	Right-click within the Mobile Application Diagram and select Generate Code .
Workspace Navigator	Right-click the Mobile Application project folder that contains the mobile objects for which you are generating API code, and select Generate Code .

2. (Optional) Enter the information for these options:

Note: This page of the code generation wizard is seen only if you are using the Advanced developer profile. See *Switching Between Developer Profiles*.

Option	Description
Select code generation configuration	<p>Select either an existing configuration that contains code generation settings, or generate device client code without using a configuration:</p> <ul style="list-style-type: none"> • Continue without a configuration – select this option to generate device code without using a configuration. • Select an existing configuration – select this option to either select an existing configuration from which you generate device client code, or create a new configuration. Selecting this option enables: <ul style="list-style-type: none"> • Select code generation configuration – lists any existing configurations, from which you can select and use for this session. You can also delete any and all existing saved configurations. • Create new configuration – enter the Name of the new configuration and click Create to save the configuration for future sessions. Select an existing configuration as a starting point for this session and click Clone to modify the configuration.

3. Click **Next**.
4. In **Select Mobile Objects**, select all the MBOs in the mobile application project or select MBOs under a specific synchronization group, whose references, metadata, and dependencies (referenced MBOs) are included in the generated device code.

Dependent MBOs are automatically added (or removed) from the **Dependencies** section depending on your selections.

Unwired Server automatically computes the page size after you choose the MBOs based on total attribute size. If an MBO's accumulated attribute size is larger than the page size setting, a warning displays.

Note: Code generation fails if the server-side (run-time) enterprise information system (EIS) data sources referenced by the MBOs in the project are not running and available to connect to when you generate object API code. You can avoid this failure by installing the `net_rim_api.jar` file.

5. Click **Next**.
6. Enter the information for these configuration options:

Option	Description
Language	Choose the language used for developing the client applications: <ul style="list-style-type: none"> • Java • C# • Objective-C
Platform	Select the platform (target device) from the drop-down list for which the device client code is intended. The platform is dependent on the language selected. <ul style="list-style-type: none"> • Java <ul style="list-style-type: none"> • Java ME for BlackBerry • C# <ul style="list-style-type: none"> • NET Framework for Windows • NET Compact Framework 3.5 for Windows Mobile • Objective C <ul style="list-style-type: none"> • iOS
Unwired Server	Specify a default Unwired Server connection profile to which the generated code connects at runtime.

Option	Description
Server domain	<p>Choose the domain to which the generated code will connect. If you specified an Unwired Server to which you previously connected successfully, the first domain in the list is chosen by default. You can enter a different domain manually.</p> <hr/> <p>Note: This field is only enabled when an Unwired Server is selected.</p>
Page size	<p>Optionally, select the page size for the generated client code. If the page size is not set, the default page size is 4KB at runtime. The default is a proposed page size based on the selected MBO's attributes.</p> <p>The page size should be larger than the sum of all attribute lengths for any MBO that is included with all the MBOs selected, and must be valid for the database. If the page size is changed, but does not meet these guidelines, object queries that use string or binary attributes with a WHERE clause may fail.</p> <hr/> <p>Note: This field is only enabled when an Unwired Server is selected. The page size option is not enabled for message-based applications.</p>
Package, Namespace, or Name Prefix	<ul style="list-style-type: none"> • Package – enter a package name for Java. • Namespace – enter a namespace for C#. • Name Prefix – enter a name prefix for Objective C.

Option	Description
Destination	<p>Specify the destination of the generated device client files. Enter (or Browse) to either a Project path (Mobile Application project) location or File system path location. Select Clean up destination before code generation to clean up the destination folder before generating the device client files.</p> <hr/> <p>Note: If you select Java as the language, enter a project path, specify a mobile application project folder, and select Generated Code as the destination. JAR files are automatically added to the destination for the platform that supports compiling of the generated client code.</p>
Replication-based	<p>Select to use replication-based synchronization.</p> <hr/> <p>Note: This option is not available for mobile applications that will run on iOS devices.</p>
Message-based	<p>Select to use message-based synchronization.</p> <hr/> <p>Note: Use message based synchronization for iOS mobile applications. This option is not available for Java or C# applications.</p>

7. Select **Generate metadata classes** to generate metadata for the attributes and operations of each generated client object.
8. Select **Generate metadata and object manager classes** to generate both the metadata for the attributes and operations of each generated client object and an object manager for the generated metadata.
 The object manager allows you to retrieve the metadata of packages, MBOs, attributes, operations, and parameters during runtime using the name instead of the object instance.
9. If you selected Java as the language, you can select **Generate JavaDoc** to include the Object API JavaDoc documentation in the output directory.
10. Click **Finish**.

Installing the net_rim_api.jar File

Install the net_rim_api.jar file to avoid code generation errors when generating Java code for BlackBerry native application development.

After generating Java code for BlackBerry, error icons appear next to the project for which you generated the code. These errors can be viewed in the Problems view and usually in the project's DB.java file, and appear because of dependencies on the net_rim_api.jar file if it is not in the project build path. Because you typically build the application using either the BlackBerry Java Plug-in for Eclipse (eJDE) or BlackBerry JDE you can ignore these errors, however, you can avoid them by following this procedure for any projects for which you generate Java code for BlackBerry.

1. Install the BlackBerry development environment.
See Installing the BlackBerry Development Environment in the Developer Guide for BlackBerry
2. Right-click the Mobile Application Project, and select **Properties**.
3. Select **Java Build Path**.
4. Select the **Libraries** tab, and click **Add External Jars**.
5. locate and select the net_rim_api.jar file, which depends on where the JDE is installed. For example, C:\Program Files\Research In Motion\BlackBerry JDE 5.0.0\lib.
6. Click **OK** until you exit the dialog.

Develop

Eclipse Basics

Basics topics describe user interface functionality. Topics describe perspectives, views, editors, resources, and other general user interface features and tasks.

Some features and tasks are provided by Eclipse functionality.

For additional information, see the Eclipse *Workbench User Guide* located at the main level of the online help bookshelf. This documentation is also available on the Eclipse Web site at <http://www.eclipse.org/documentation/>.

Opening a Perspective

Open a perspective to work with its resources to perform a task.

1. Choose one of:

- Click **Open Perspective**  on the main toolbar.

- Select **Window > Open Perspective** from the main menu bar.

2. If the perspective that you want to open is not in the list, select **Other**.

3. Select the perspective that you want to open and click **OK**.

Perspectives

A perspective provides a set of capabilities that enable you to work with resources to perform a task.

A perspective is the arrangement of views and editors in the *Workbench*.

- Views provide ways to navigate and work with *resources*. Each view has associated menus and may have its own toolbar.
- Editors provide tools to create and modify resources.
- Menu bars and context menus provide the items you need to create and manipulate resources.
- Creation wizards, which are associated with the resources in a view, guide you through the process of creating resources, for example, a project.
- Cheat sheets guide you through complex tasks by either showing you how to perform the task or performing some of the task for you. A cheat sheet opens as a view in a perspective.


More than one perspective can be open, but only one perspective at a time can be displayed in the Workbench. When more than one perspective is open, you can select the perspective that you want to display from the Perspective shortcut bar.

See also

- *Perspective Shortcut Bar* on page 238
- *Rearranging Views in a Perspective* on page 238
- *Moving the Perspective Shortcut Bar* on page 239
- *Resetting an Active Perspective to its Default Appearance* on page 240
- *Resetting an Inactive Perspective to its Default Appearance* on page 240

Perspective Shortcut Bar

Use the Perspective shortcut bar to quickly access open *perspectives*.

You can also open a perspective using the Open Perspective button  on the shortcut bar. Multiple perspectives can be open, but only one perspective can be active at a time. When you select a perspective from the Perspective shortcut bar, it becomes the active perspective. Any perspective that you open, but do not close, appears on the Perspective shortcut bar.

You can rearrange the order of the perspectives by dragging and dropping the perspective tabs in the shortcut bar.

By default, the Perspective shortcut bar is docked horizontally in the top-right corner. It can also be docked under the main toolbar or vertically to the left of a perspective. Right-click the shortcut bar to change the docking location.

See also

- *Perspectives* on page 237
- *Rearranging Views in a Perspective* on page 238
- *Moving the Perspective Shortcut Bar* on page 239
- *Resetting an Active Perspective to its Default Appearance* on page 240
- *Resetting an Inactive Perspective to its Default Appearance* on page 240







Rearranging Views in a Perspective

Rearrange the views in a perspective by moving a view to a new docking location in the perspective.

1. Click in the title bar of the view that you want to move.
2. Hold down the left mouse button and drag the view to the new area.

As you move the view, the drop cursor icon changes appearance to help you determine where the view can be docked.

Table 59. Drop cursors

Drop cursor	Cursor name	Description
	Dock Above	Dock above the view that is under the cursor.
	Dock Below	Dock below the view that is under the cursor.
	Dock to the Right	Dock to the right of the view under the cursor.
	Dock to the Left	Dock to the left of the view under the cursor.
	Stack	The view appears as a tab in the view under the cursor.
	Restricted	The view cannot be docked. For example, a view cannot be docked in an editor.

- When the view is in position, release the left mouse button to drop the view onto the new location.

When you close the application, the new configuration is saved.

See also

- *Perspectives* on page 237
- *Perspective Shortcut Bar* on page 238
- *Moving the Perspective Shortcut Bar* on page 239
- *Resetting an Active Perspective to its Default Appearance* on page 240
- *Resetting an Inactive Perspective to its Default Appearance* on page 240

Moving the Perspective Shortcut Bar

The Perspective shortcut bar runs horizontally in the upper left corner of a perspective by default.

The Perspective shortcut bar can be docked horizontally at the top right, or vertically to the left of a perspective.

- Right-click in the Perspective shortcut bar to open its context menu.
- Do one of the following:

Select	To dock the shortcut bar
Dock on > Top Right	At the top right, horizontally adjacent to the main toolbar.

Select	To dock the shortcut bar
Dock on > Top Left	At the top left, horizontally below the main toolbar. This is the default.
Dock on > Left	At the top right, vertically on the side of a perspective.

See also

- *Perspectives* on page 237
- *Perspective Shortcut Bar* on page 238
- *Rearranging Views in a Perspective* on page 238
- *Resetting an Active Perspective to its Default Appearance* on page 240
- *Resetting an Inactive Perspective to its Default Appearance* on page 240

Resetting an Active Perspective to its Default Appearance

After you customize a perspective, you can return to its default layout.

1. Select **Window > Reset Perspective** from the main menu bar.
2. Click **OK** to reset the perspective to its original layout.

See also

- *Perspectives* on page 237
- *Perspective Shortcut Bar* on page 238
- *Rearranging Views in a Perspective* on page 238
- *Moving the Perspective Shortcut Bar* on page 239
- *Resetting an Inactive Perspective to its Default Appearance* on page 240

Resetting an Inactive Perspective to its Default Appearance

After you customize a perspective, you can return to its default layout.

1. Select **Window > Preferences**.
2. Expand **General** and select **Perspectives**.

The right pane is titled Perspectives.

3. From the **Available Perspectives** list, select the perspective that you want to reset.
4. Click **Restore Defaults**.
5. Click **OK** to reset the perspective to its original layout.

See also

- *Perspectives* on page 237
- *Perspective Shortcut Bar* on page 238
- *Rearranging Views in a Perspective* on page 238

- *Moving the Perspective Shortcut Bar* on page 239
- *Resetting an Active Perspective to its Default Appearance* on page 240

Opening a View

You can open a view in a perspective.

1. Select **Window > Show View** from the main menu bar.

A list of available views displays.

Note: If the view you want does not display, select **Window > Show View > Other**.

2. Select the view that you want to open.

See also

- *Creating a Fast View* on page 243
- *Opening a Perspective* on page 237
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247

Views

A view is similar to a pane. Views are based on perspectives and provide different ways to navigate and work with *resources*.

Views can be:

- Moved from area to area with simple drag and drop to customize your perspective.
- Stacked on top of one another in a tabular form to reduce clutter and increase the area for competing views.
- Detached from a perspective and displayed as a standalone window. When you maximize pages in the editor view or design canvas, all detached views remain open and visible. However, detached views do not remain visible when you change perspectives.

Each view has associated menus and may have its own toolbar. The menu at the top left of the view's title bar provides common functions to manipulate the view, such as moving, resizing, and maximizing the view. The pull-down menu at the right end of the view's title bar contains functions that apply to all of the resources in a view, not to just one particular resource. These functions may include sorting or filtering. The menus and toolbar associated with a view only affect the resources in that view. Modifications made in a view are saved immediately.

See also

- *Detaching a View* on page 242
- *Floating a View* on page 242
- *Creating a Fast View* on page 243

- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Editors* on page 247
- *Resources* on page 248

Detaching a View

You can detach any number of views from a perspective and display them as separate windows that float on the perspective.

When you maximize pages in the editor view or design canvas, all detached views remain open and visible. However, detached views do not remain visible when you change perspectives.

1. Right-click the tab of the view you want to detach and select **Detached**.

The view reappears as a standalone window.

2. Drag the view to the desired location.
3. To reanchor the view, right-click the tab of the view and unselect **Detached**.

The view is reattached to its default location.

See also

- *Views* on page 241
- *Floating a View* on page 242
- *Creating a Fast View* on page 243
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Editors* on page 247
- *Resources* on page 248

Floating a View

You can detach a view and float it on top of a perspective.

1. Click in the title bar of the view that you want to float.
2. Hold down the left mouse button and drag the view to an area on your desktop outside the perspective.
3. When the view is in position, release the left mouse button to drop the view onto the new location.

When you close the application, the new configuration is saved.

See also

- *Views* on page 241
- *Detaching a View* on page 242

- *Creating a Fast View* on page 243
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Editors* on page 247
- *Resources* on page 248

Creating a Fast View

A Fast View lets you manage the use of perspective space.

1. Click in the title bar of the *view* that you want to move.
2. Drag and drop the view onto the **Fast View** shortcut bar, which, by default, runs vertically to the left of a perspective.

When you release the mouse button and drop the view, its icon appears on the Fast View shortcut bar, and the view no longer appears in the perspective.

See also

- *Views* on page 241
- *Detaching a View* on page 242
- *Floating a View* on page 242
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Editors* on page 247
- *Resources* on page 248
- *Opening a Perspective* on page 237

Fast Views

A Fast View is an effective way to store and quickly access views in a perspective.

Fast Views:

- Help you manage the use of your perspective space.
- Give you quick and easy access to views that are open, but are not currently displayed in your perspective.
- Display in the perspective with a single click from the Fast View shortcut bar.
- Move back to the Fast View shortcut bar when you click outside the view.

Fast View Shortcut Bar

Use the Fast View shortcut bar to quickly access an open view.

The Fast View shortcut bar may, by default, either run vertically down the left side of a perspective or horizontally at the bottom left of the perspective. You can also dock the shortcut bar vertically to the right of a perspective. When a view is dropped onto the Fast View shortcut bar, the view appears as an icon.

Converting a Fast View to a View

You can convert a Fast View to a view.

1. Click the icon of the view on the **Fast View** shortcut bar that you want to convert to a view.

The Fast View shortcut bar, by default, runs vertically on the left of a perspective.

2. Drag the view and drop it in the perspective.

Moving the Fast View Shortcut Bar

Move the Fast View shortcut bar to dock it horizontally at the bottom of a perspective or vertically to the right of a perspective.

The Fast View shortcut bar, by default, runs vertically down the left side of a perspective.

1. Right-click in the Fast View shortcut bar.
2. Do one of the following:

Select	To dock the shortcut bar
Dock On > Left	Vertically on the left side of a perspective. This is the default.
Dock On > Right	Vertically on the right side of a perspective.
Dock On > Bottom	Horizontally at the bottom of a perspective.

WorkSpace Navigator

The WorkSpace Navigator view displays *resources* in a hierarchy.

The top-level resource, or parent, is always a project, which is a container for all other resources.

Use tools in the WorkSpace Navigator view to:

- Navigate around your resources using the WorkSpace Navigator toolbar
- Show or hide file extensions
- Show or hide WorkSpace Navigator extensions
- Filter
- Sort
- Link a resource with an editor
- Select project natures
- View resources by file type


See also

- *Views* on page 241
- *Detaching a View* on page 242

- *Floating a View* on page 242
- *Creating a Fast View* on page 243
- *Enterprise Explorer* on page 247
- *Editors* on page 247
- *Resources* on page 248
- *Opening a Perspective* on page 237

Showing File Extensions

You can display hidden file extensions in the WorkSpace Navigator.

1. Click **Menu**  in the title bar of the WorkSpace Navigator.
2. Select **Show File Extensions**.

Creating a Working Set

Use working sets in WorkSpace Navigator and Enterprise Explorer to limit the resources that appear.

If you select a working set, only the resources contained in the working set appear in your resource view.

Note: Currently, the working set feature in Enterprise Explorer supports elements in SQL Anywhere, Adaptive Server Enterprise, and connection profiles only.

1. From the WorkSpace Navigator or Enterprise Explorer toolbar, click the arrow to open the drop-down menu.
2. Select **Select Working Set**.
3. In the Select Working Set dialog box, click **New**.
4. Select the working set type that you want to use.
5. Click **Next**.
6. Enter a **Working set name**.
7. Expand the directory structure for **Working Set Contents**, and select the resources and folders for the working set definition.
8. Click **Finish**.
9. Select the working set that you just created and click **OK**.

The Select Working Set dialog box closes. Only the resources and folders that you selected now appear in the WorkSpace Navigator or Enterprise Explorer.

Editing the Active Working Set

You can edit a working set, which restricts displayed resources.

Both the WorkSpace Navigator and Enterprise Explorer support working sets.

1. From the toolbar of the view displaying the working set, click **Menu**.
2. Select **Edit Active Working Set**.
3. Expand the directory structure for **Working Set Contents**, and select the resources and folders to include in the working set.
4. Click **Finish**.

Unselecting a Working Set

If you unselect a working set, the hidden resources display.

Both the WorkSpace Navigator and Enterprise Explorer support working sets.

1. From the toolbar of the view displaying the working set, click **Menu**.
2. Select **Deselect Working Set**.

The working set is deselected and all *resources* display.

Filtering Resources

Filter the resources in the WorkSpace Navigator to show only the resources that you want to see.

1. From the WorkSpace Navigator title bar, click the arrow to open the drop-down menu.
2. Select **Customize View**.
3. Select the **Filters** tab.
4. To use a Workbench-provided filter, from the Filters list, select each resource that you want to filter (hide).
5. Click **OK**.

The WorkSpace Navigator now displays the only the filtered resources.

6. After a filter is defined, you can toggle it both off and on using **Filters Toggle** from the title bar drop-down menu.

Linking a Resource to a Specific Editor

You can associate a specific editor to open with a resource.

If you link a resource to an editor, when you select the editor, the resource is selected in the WorkSpace Navigator. Conversely, when you select the resource in the WorkSpace Navigator, the editor is selected.

1. Select the resource in the WorkSpace Navigator.
2. Do one of the following:
 - Right-click and select **Open With** > <editor name>.
 - Click **Link with Editor** in the Workspace Navigator toolbar.

Enterprise Explorer

The Enterprise Explorer view displays enterprise resources, such as servers, in folders organized by type.

The Enterprise Explorer view contains an icon for each enterprise resource and external server you have set up in your *workspace*. Each is represented by a Connection Profile icon. Use Enterprise Explorer to create connection profiles to access and use the associated servers.

See also

- *Views* on page 241
- *Detaching a View* on page 242
- *Floating a View* on page 242
- *Creating a Fast View* on page 243
- *WorkSpace Navigator* on page 244
- *Editors* on page 247
- *Resources* on page 248
- *Opening a Perspective* on page 237

Editors

The editor area of a perspective is where a *resource* is created or modified.

When you double-click a resource in the WorkSpace Navigator, an editor opens. The resource type determines what type of editor is opened. For example, if you are creating a service, the Service editor opens.

Most editors initially open to the Introduction page.

- Click Start to go to the editor page to start working.
- Click Tutorial to open a cheat sheet that guides you through the task.
- Click Help to open the help topic for the editor.

Editor tabs

More than one editor can be opened at a time, but only one editor is active. Each open editor has a tab labeled with its resource name. To switch to a different editor, click its tab. If numerous editors are open, use the scroll arrows to the left of the editor tabs to view the open editors. When an editor is active, the main menu bar and toolbar display tools applicable to that editor.

Unsaved changes

An asterisk displays on the left-hand side of the editor's tab to indicate that the editor contains unsaved changes. If you are closing either the editor or the application, you are prompted to save these changes. You can close either one selected open editor or all open editors at one

time. If you are closing all open editors, you are prompted to save the changes in any editor that contains unsaved changes.

See also

- *Views* on page 241
- *Detaching a View* on page 242
- *Floating a View* on page 242
- *Creating a Fast View* on page 243
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Resources* on page 248

Resources

A resource is any object that is added to the WorkSpace Navigator, such as a project, mobile business object, or WSDL file.

When you modify a resource in the WorkSpace Navigator, such as copy, move, rename, or delete it, the resource maintains its referential integrity. This means that the path to its referenced files is updated to reflect the change.

See also

- *Views* on page 241
- *Detaching a View* on page 242
- *Floating a View* on page 242
- *Creating a Fast View* on page 243
- *WorkSpace Navigator* on page 244
- *Enterprise Explorer* on page 247
- *Editors* on page 247

Renaming a Resource

You can rename a resource while maintaining integrity with its referenced files.

This means that the path to the modified resource is updated to reflect the change. If the change results in an error, the service referencing the renamed file is marked with a Problem marker.

1. Right-click the resource you want to rename in the WorkSpace Navigator and select **Rename**.
2. Type a new name in the text box using the following naming conventions:
 - Begin the name with a letter. Do not begin the name with a number or an underscore.
 - Use alphabetic or numeric characters and underscores after the initial letter.
 - Do not use spaces in project or folder names.

3. Click anywhere outside the text box to save the changes.

Note: If you are renaming a Java file, only the file is renamed, the underlying class is not changed. You must manually change the class name in the source.

Moving a Resource

You can move a resource from one project to another in the WorkSpace Navigator and maintain its integrity with its referenced files.

This means that the path to the modified resource is updated to reflect the change. If the change results in an error, the service referencing the moved file is marked with a Problem marker.

1. Select the resource you want to move in the WorkSpace Navigator.
2. Drag and drop the resource onto the target project.

Exporting a Resource

Export projects, file system resources, schemas, or zip files to an external file system.

When you export a resource, a copy of the resource is made for the external file system and the original resource remains unaltered; however, caution should be exercised when exporting resources at the file level. Referential integrity may be lost as referenced files are not automatically exported.

Note: Using an external tool to process an exported resource can cause the export and subsequent import to be incomplete and unsuccessful.

1. Select **File > Export** from the main menu bar.

The Export wizard opens.

2. Expand **General**, select **File System**, and click **Next**.

The File System dialog opens.

3. In the left pane, select the project or folder that contains the resource you want to export.

Its contents display in the right pane.

4. In the right pane, select the resources that you want to export.

5. In the **To directory** field, browse for the destination directory that you want to export the file into.

6. Select any of the following options:

- Overwrite existing files without warning
- Create directory structure for files
- Create only selected directories

7. Click **Finish** to export the specified resources to the destination location.

Importing a Resource

Use the Import wizard to import projects, file system resources, PowerDesigner models, schemas, or zip files.

When you import a resource, a copy of the resource is made and brought into Sybase WorkSpace, while the original resource remains unaltered. However, when importing at the file level, referential integrity may be lost as any referenced files are not automatically imported.

1. Select **File > Import** from the main menu bar.
2. Select **File System** and click **Next**.
3. Browse for the following:

Table 60. File system

Field	Description
From directory	<ol style="list-style-type: none">1. Browse for a folder that contains the resource you want to import.2. Expand the folder to display its contents, then select the specific resources you want to import.
Into folder	Browse for the project into which you want to import the resource.

4. Select any of the following options:
 - Overwrite existing resources without warning.
 - Create complete folder structure.
 - Create selected folders only.
5. Click **Finish** to import the specified resources to the destination location.

Deleting a Resource

When you delete a resource, any referenced files are also deleted.

If the change results in an error, the service referencing the deleted file is marked with a Problem marker.

1. Right-click the resource you want to delete in the WorkSpace Navigator and select **Delete**.

A message appears asking you to confirm the deletion.

2. Click **OK**.

Introduction to Sybase Help

Online help topics are contained in several documentation collections that appear in the left pane of the Help Contents. The source of a help topic is identified in the title bar above the content window.

See also

- *Opening a Perspective* on page 237
- *Opening a View* on page 241

Help Features

Familiarize yourself with online help features.

Table 61. Online help features

Feature	Description
Preferences	Using the Preferences dialog box, specify how you want help to display: <ul style="list-style-type: none"> • Using an external browser. • Displaying context-sensitive topic selections in a Help view or through a pop-up window. • Opening context-sensitive topics in the Help view or in the editor area.
Opening the online help bookshelf	You can open the online help in an embedded Web browser from a variety of locations. You can also use an external Web browser. See Online Help Access below for more information.
Opening the Help view	A dynamic Help view provides context-sensitive help for the current task. In wizards, preference windows, launch configurations, searches and other multi-page dialog boxes, press F1 or select the help icon. While the Help view is open, the contents will update automatically as you progress from page to page in the user interface.

Feature	Description
Printing a single help topic	<p>Print selected topics by clicking the Print Page icon in the Help Contents toolbar (right pane).</p> <p>You can also select the Print Topics icon in the Table of Contents toolbar (left pane) and select Print Selected Topic.</p>
Printing a group or collection of topics	<p>Print a group or collection of topics by clicking the Print Topics icon in the table of Contents toolbar (left pane) and select Print Selected Topic and All Subtopics.</p> <p>If you have PDF capabilities, you can choose to print the group of topics to the PDF-designated printer to create a PDF.</p>
Searching for a topic	Use the Search feature to locate topics based on key words. You can perform a help search directly from the bookshelf or using the dynamic Help view.
Collections	The online help is organized into collections of topics in the online bookshelf. To identify a topic, view the topic title bar.
Glossaries	Each collection has its own glossary. To access a glossary, expand the collection category in the Table of Contents (left pane). The glossary is the last topic of each collection.
Breadcrumbs	<p>A hierarchy displays at the top of each topic. You can review the hierarchical sequence of the topic you are viewing in the collection, and you can also select a category and move up the hierarchy.</p> <p>To return to your original topic, click the Back arrow in the tool bar.</p>

Feature	Description
Tutorials, Samples, and Quick Start Videos	<p>You can access information by:</p> <ul style="list-style-type: none"> • Clicking the Tutorials, Samples, or Quick Start Videos icons on the WorkSpace Welcome page. • Selecting Help from the main menu bar. • Reviewing the appropriate online help topic for Tutorials, Samples, or Quick Start Videos. <hr/> <p>Note: Tutorials, Samples, or Quick Start videos may not be available.</p>

For additional information about Eclipse Help system features, see the Eclipse *Workbench User Guide* located at the main level of the online help bookshelf. This documentation is also available on the Eclipse Web site at <http://www.eclipse.org/documentation/>.

Searching the Help

Use the navigator capabilities to search for help.

The help navigator enables you to navigate through help topics, launch general and specific search queries, search from the toolbar, and search using key combinations from the keyboard.

1. Select **Help > Help Contents** from the main menu bar to open the help.
2. Determine the type of search to launch. For example, you may know exactly what you want, or you may want to start with a broad search across several components, and then narrow the search once you see the results.
3. Launch the search.
4. Review the search results, and then refine it if necessary.

Searching for Help Topics From the Bookshelf

You can search for help across all bookshelf documentation sets or narrow your search to selected documentation sets.

To perform a search in Google or Eclipse.org, use the dynamic Help View.

1. To open the bookshelf, select **Help > Help Contents** from the main menu bar.
2. Define the search scope by clicking **Search Scope** in the toolbar.

The Select Search Scope dialog box opens.

3. Select **Search only the following topics** and click **New**.

The New Search List dialog box opens.

4. Define the search scope:
 - a) Enter a descriptive name for your search in the **List name** field.
 - b) Select the topics you want to search in the **Topics to search** list box.
 - c) Click **OK**.

The Select Search Scope dialog box opens with your selection highlighted.

5. Click **OK** to return to the bookshelf.

Note: Selecting a top-level checkbox extends the scope of the search to all subtopics. Use the plus and minus sign to the left of a topic to expand and collapse associated subtopics.

6. Type a query in the **Search** field and click **Go**.

The search results display in the Search Results pane.

Note: The narrowed search scope remains in effect for future searches until it is changed.

7. To search across all documentation sets, type a query in the **Search** field and click **Go**.

The search results display in the Search Results pane.

Searching for Help Topics From the Help View

You can perform a topic search in the online help, Google, or Eclipse.org using the dynamic Help view.

To specify a documentation set in which to search in the help, perform your search directly from the bookshelf.

1. Select **Help > Search** or **Help > Help Contents** from the from the main menu bar.
2. Type a query in the **Search** field, then click **Go**.
3. To refine your search scope, click **Search Scope** in the Help window, or click **Default** in the Help dialog box.

Navigating the Help

Select a documentation set and use the navigator to view its topics.

Use the navigator to locate a topic and navigate through related topics. Key combinations are also available for navigation.

To navigate the help:

1. Select **Help > Help Contents** from the main menu bar to open the help.
2. From the online help bookshelf, select the documentation set that you want to view.
3. Expand the documentation set to find the topic that you want to see.
4. To view the topic, click its link.
5. Use the **Go Forward** and **Go Back** buttons to return to topics you have previously viewed.

6. To synchronize the navigator with the current topic, click **Show in Table of Contents**



Synchronizing is useful if you follow several links or perform a search and you want to match the navigation tree with the current topic.

Opening the Online Help Bookshelf

The bookshelf is organized into documentation sets.

In addition to application-specific documentation sets, the bookshelf may also display other documentation sets:

- Sybase server documentation collections, such as for EAServer, SQL Anywhere, and Adaptive Server Enterprise, are displayed if the servers are installed on the same machine as the application.
- Documentation sets provided by Eclipse, such as the Workbench User Guide and the Java Development User Guide, are also available.

You can open, review, and search all of these documentation sets from the bookshelf.

1. Select **Help > Help Contents** from the main menu bar to open the bookshelf in a Web browser.
2. In the bookshelf, expand the documentation set you want to view.
3. Click a topic to display its contents in the right pane.
4. Do any of the following:
 - Click the **Go Back** and **Go Forward** arrows to return to a previous topic and sequence forward again.
 - Click **Show in Table of Contents** to synchronize the Contents pane with the current topic.

Synchronization is useful if you follow several links or if you perform a search and want to match the navigation tree with the current topic.

Searching all Documentation Sets

Use search queries to launch a search across multiple documentation sets.

1. Select **Help > Help Contents** from the main menu bar to open the help.
2. Enter a query in the **Search** field.

Use the following rules to enter a query:

- Use AND to require the term on each side of the AND operator be present in the topic. There is an implied AND between search terms. Topics that contain every term in the query are listed in the search. For example, if you enter `database service`, topics

that contain both the term database and the term service display. Topics that contains only the term service or only the term database do not display.

- Use **OR** before optional terms. For example, if you enter `database OR service`, topics that contain either the term database or the term service display.
- Use **NOT** before a term that you want to exclude from the search results. For example, `database NOT service` displays topics that contain the term database, but do not contain the term service.
- Use **?** to match any single character. For example, `plu?` displays topics that contain `plug`.
- Use ***** to match any set of characters, including an empty string. For example, `plu*` displays topics that contain `plug` or `plugin`.
- Use double quotation marks to enclose a term that is to be treated as a phrase. For example, `"edit menu"` displays topics with this entire term, not topics with only the term `edit` or the term `menu`.
- Case is ignored. For example, `database service` displays database service, Database Service, and DATABASE SERVICE.
- Punctuation acts as a term delimiter. For example, `web.xml` displays topics that contain `web.xml`, `web`, and `xml`. To display only topics that contain `web.xml`, enclose the term in double quotes.
- In a search query, if you enter `create`, topics that contain `create`, `creates`, `creating`, and `creation` are displayed. To only see the term `create`, enclose the term in double quotes.
- The following English words are ignored in search queries: a, and, are, as, at, be, but, by, in, into, is, it, no, not, of, on, or, s, such, t, that, the, their, then, there, these, they, to, was, will, with.

3. Click **Go**.

The search results display in the Search Results pane.

Narrowing a Search

Use search queries to narrow a search.

The help navigator enables you to narrow your search.

1. Select **Help > Help Contents** from the main menu bar to open the help.
2. Click **Search Scope**.
3. Select **Search only the following topics**.
4. Click **New**.
5. Type a descriptive name for your search in the **List name** field.
6. Select the topics you want to search in the **Topics to search** section.

Selecting a top-level checkbox extends the scope of the search to all subtopics. Use the plus and minus signs to the left of topics to expand and collapse the associated subtopics.

7. Click **OK** .

The Select Search Scope dialog box appears with your selection highlighted.

8. Click **OK** .

9. Click **Go** .

The search results display in the Search Results pane.

Note: The narrowed search scope remains in effect for future searches until it is changed.

Search Keyboard Shortcuts

Use keyboard shortcuts to launch search queries.

Table 62. Keyboard Shortcuts

Keyboard Shortcut	Action
Press Tab inside a frame (page)	Move to the next link, button, or topic.
Press Right/Left arrows	Expand or collapse a tree node.
Press Down arrow or Tab	Move to the next topic node.
Press Up arrow or Shift+Tab	Move to the previous topic node.
Press Home or End	Scroll all the way up or down.
Press Alt+Left arrow	Go back.
Press Alt+Right arrow	Go forward.
Press Ctrl+Tab	Go to the next topic.
Press Shift+Ctrl+Tab	Move to previous topic.
Press Ctrl+p.	Print the current page or active topic.

Setting Help Display Preferences

Define how you want to display help topics from an Eclipse-based product to appear.

1. Select **Window | Preferences** from the main menu bar.
2. In the left pane, select **Help**.

The Help options appear in the right pane.

3. Specify how to display help topics.
4. Click **OK**.

Troubleshoot

Use troubleshooting tips to isolate and resolve common issues.

See *Troubleshooting Sybase Unwired Platform* for information about troubleshooting issues with the Eclipse - based user interface or other Sybase Unwired Platform components.

API Documentation

You can use Sybase Unwired Platform APIs to develop and customize mobile applications, data handling, error handling, and system functionality programmatically.

- **Client Object API** – generated business object classes that represent the mobile business object model built and designed in the Unwired WorkSpace development environment. Use the API to synchronize and retrieve data and invoke mobile business object operations. See the developer references, and the Javadocs included in the installation directory:
 - *Developer Guide: BlackBerry Native Applications*
 - *Developer Guide: iOS Native Applications*
 - *Developer Guide: Windows and Windows Mobile Native Applications*
 - *Developer Guide: Mobile Workflow Packages*
- **Unwired Server API** – custom Java classes used to implement advanced data handling features in Unwired Server. See *Developer Guide: Unwired Server* and the Javadocs included in the installation directory.
- **Administration API** – custom Java classes used to integrate Sybase Unwired Platform system management tools with your enterprise system. See *Developer Guide for Unwired Server Management APIs* and the Javadocs included in the installation directory.

Check the Sybase Product Documentation Web site regularly for updates: access <http://sybooks.sybase.com/nav/summary.do?prod=1289>, then navigate to the most current version.

See also

- *Product Task Flow* on page 1

Glossary: Sybase Unwired Platform

Defines terms for all Sybase Unwired Platform components.

administration perspective – Or administration console. The Unwired Platform administrative perspective is the Flash-based Web application for managing Unwired Server. *See* Sybase Control Center.

administrators – Unwired Platform users to which an administration role has been assigned. A user with the "SUP Administrator" role is called a "platform administrator" and a user with the "SUP Domain Administrator" role is called a "domain administrator". These administration roles must also be assigned SCC administration roles to avoid having to authenticate to Sybase Control Center in addition to Unwired Server:

- A domain administrator only requires the "sccUserRole" role.
- A platform administrator requires both the "sccAdminRole" and "sccUserRole" roles.

Adobe Flash Player – Adobe Flash Player is required to run Sybase Control Center. Because of this player, you are required to run Sybase Control Center in a 32-bit browser. Adobe does not support 64-bit browsers.

Advantage Database Server[®] – A relational database management system that provides the messaging database for Sybase Unwired Platform. *See* messaging database.

Afaria[®] – An enterprise-grade, highly scalable device management solution with advanced capabilities to ensure that mobile data and devices are up-to-date, reliable, and secure. Afaria is a separately licensed product that can extend the Unwired Platform in a mobile enterprise. Afaria includes a server (Afaria Server), a database (Afaria Database), an administration tool (Afaria Administrator), and other runtime components, depending on the license you purchase.

APNS – Apple Push Notification Service.

application connection – A unique connection to the application on a device.

application connection template – a template for application connections that includes application settings, security configuration, domain details, and so forth.

application node – In Sybase Control Center, this is a registered application with a unique ID. This is the main entity that defines the behavior of device and backend interactions.

application registration – The process of registering an application with Sybase Unwired Platform. Registration requires a unique identity that defines the properties for the device and backend interaction with Unwired Server.

artifacts – Artifacts can be client-side or automatically generated files; for example: .xml, .cs, .java, .cab files.

availability – Indicates that a resource is accessible and responsive.

BAPI – Business Application Programming Interface. A BAPI is a set of interfaces to object-oriented programming methods that enable a programmer to integrate third-party software into the proprietary R/3 product from SAP®. For specific business tasks such as uploading transactional data, BAPIs are implemented and stored in the R/3 system as remote function call (RFC) modules.

BLOB – Binary Large Object. A BLOB is a collection of binary data stored as a single entity in a database management system. A BLOB may be text, images, audio, or video.

cache – The virtual tables in the Unwired Server cache database that store synchronization data. *See* CDB.

cache group – Defined in Unwired WorkSpace, MBOs are grouped and the same cache refresh policy is applied to their virtual tables (cache) in the CDB.

cache partitions – Partitioning the cache divides it into segments that can be refreshed individually, which gives better system performance than refreshing the entire cache. Define cache partitions in Unwired WorkSpace by defining a partition key, which is a load parameter used by the operation to load data into the cache from the enterprise information system (EIS).

CDB – Cache database. The Unwired Server CDB stores runtime metadata (for Unwired Platform components) and cache data (for MBOs). *See also* data tier.

CLI – Command line interface. CLI is the standard term for a command line tool or utility.

client application – *See* mobile application.

client object API – The client object API is described in the *Developer Guide: BlackBerry Native Applications*, *Developer Guide: iOS Native Applications*, and *Developer Guide: Windows and Windows Mobile Native Applications*.

cluster – Also known as a server farm. Typically clusters are setup as either runtime server clusters or database clusters (also known as a data tier). Clustering is a method of setting up redundant Unwired Platform components on your network in order to design a highly scalable and available system architecture.

cluster database – A data tier component that holds information pertaining to all Unwired Platform server nodes. Other databases in the Unwired Platform data tier includes the cache, messaging, and monitoring databases.

connection – Includes the configuration details and credentials required to connect to a database, Web service, or other EIS.

connection pool – A connection pool is a cache of Enterprise Information System (EIS) connections maintained by Unwired Server, so that the connections can be reused when Unwired Server receives future requests for data.

For proxy connections, a connection pool is a collection of proxy connections pooled for their respective back-ends, such as SAP Gateway.

connection profile – In Unwired WorkSpace, a connection profile includes the configuration details and credentials required to connect to an EIS.

context variable – In Unwired WorkSpace, these variables are automatically created when a developer adds reference(s) to an MBO in a mobile application. One table context variable is created for each MBO attribute. These variables allow mobile application developers to specify form fields or operation parameters to use the dynamic value of a selected record of an MBO during runtime.

data change notification (DCN) – Data change notification (DCN) allows an Enterprise Information System (EIS) to synchronize its data with the cache database through a push event.

data refresh – A data refresh synchronizes data between the cache database and a back-end EIS so that data in the cache is updated. *See also* scheduled data refresh.

data source – In Unwired WorkSpace, a data source is the persistent-storage location for the data that a mobile business object can access.

data tier – The data tier includes Unwired Server data such as cache, cluster information, and monitoring. The data tier includes the cache database (CDB), cluster, monitoring, and messaging databases.

data vault – A secure store across the platform that is provided by an SUP client.

deploy – (Unwired Server) Uploading a deployment archive or deployment unit to an Unwired Server instance. Unwired Server can then make these units accessible to users via a client application that is installed on a mobile device.

There is a one-to-one mapping between an Unwired WorkSpace project and a server package. Therefore, all MBOs that you deploy from one project to the same server are deployed to the same server package.

deployment archive – In Unwired WorkSpace, a deployment archive is created when a developer creates a package profile and executes the **build** operation. Building creates an archive that contains both a deployment unit and a corresponding descriptor file. A deployment archive can be delivered to an administrator for deployment to a production version of Unwired Server.

deployment descriptor – A deployment descriptor is an XML file that describes how a deployment unit should be deployed to Unwired Server. A deployment descriptor contains role-mapping and domain-connection information. You can deliver a deployment descriptor and a deployment unit—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

deployment mode – You can set the mode in which a mobile application project or mobile deployment package is deployed to the target Unwired Server.

deployment profile – A deployment profile is a named instance of predefined server connections and role mappings that allows developers to automate deployment of multiple

packages from Sybase Unwired WorkSpace to Unwired Server. Role mappings and connection mappings are transferred from the deployment profile to the deployment unit and the deployment descriptor.

deployment unit – The Unwired WorkSpace build process generates a deployment unit. It enables a mobile application to be effectively installed and used in either a preproduction or production environment. Once generated, a deployment unit allows anyone to deploy all required objects, logical roles, personalization keys, and server connection information together, without requiring access to the whole development project. You can deliver a deployment unit and a deployment descriptor—jointly called a deployment archive—to an administrator for deployment to a production version of Unwired Server.

development package – A collection of MBOs that you create in Unwired WorkSpace. You can deploy the contents of a development package on an instance of Unwired Server.

device application – *See also* mobile application. A device application is a software application that runs on a mobile device.

device notification – Replication-based synchronization (RBS) clients receive device notifications when a data change is detected for any of the MBOs in the synchronization group to which they are subscribed. Both the change detection interval of the synchronization group and the notification threshold of the subscription determine how often RBS clients receive device notifications. Administrators can use subscription templates to specify the notification threshold for a particular synchronization group.

device user – The user identity tied to a device.

DML – Data manipulation language. DML is a group of computer languages used to retrieve, insert, delete, and update data in a database.

DMZ – Demilitarized zone; also known as a perimeter network. The DMZ adds a layer of security to the local area network (LAN), where computers run behind a firewall. Hosts running in the DMZ cannot send requests directly to hosts running in the LAN.

domain administrator – A user to which the platform administrator assigns domain administration privileges for one or more domain partitions. The domain administrator has a restricted view in Sybase Control Center, and only features and domains they can manage are visible.

domains – Domains provide a logical partitioning of a hosting organization's environment, so that the organization achieves increased flexibility and granularity of control in multitenant environments. By default, the Unwired Platform installer creates a single domain named "default". However the platform administrator can also add more domains as required.

EIS – Enterprise Information System. EIS is a back-end system, such as a database.

Enterprise Explorer – In Unwired WorkSpace, Enterprise Explorer allows you to define data source and view their metadata (schema objects in case of database, BAPIs for SAP, and so on).

export – The Unwired Platform administrator can export the mobile objects, then import them to another server on the network. That server should meet the requirement needed by the exported MBO.

hostability – *See* multitenancy.

IDE – Integrated Development Environment.

JDE – BlackBerry Java Development Environment.

key performance indicator (KPI) – Used by Unwired Platform monitoring. KPIs are monitoring metrics that are made up for an object, using counters, activities, and time which jointly for the parameters that show the health of the system. KPIs can use current data or historical data.

keystore – The location in which encryption keys, digital certificates, and other credentials in either encrypted or unencrypted keystore file types are stored for Unwired Server runtime components. *See also* truststore.

LDAP – Lightweight Directory Access Protocol.

local business object – Defined in Unwired WorkSpace, local business objects are not bound to EIS data sources, so cannot be synchronized. Instead, they are objects that are used as local data store on device.

logical role – Logical roles are defined in mobile business objects, and mapped to physical roles when the deployment unit that contain the mobile business objects are deployed to Unwired Server.

matching rules – A rule that triggers a mobile workflow application. Matching rules are used by the mobile workflow email listener to identify e-mails that match the rules specified by the administrator. When emails match the rule, Unwired Server sends the e-mail as a mobile workflow to the device that matches the rule. A matching rule is configured by the administrator in Sybase Control Center.

MBO – Mobile business object. The fundamental unit of data exchange in Sybase Unwired Platform. An MBO roughly corresponds to a data set from a back-end data source. The data can come from a database query, a Web service operation, or SAP. An MBO contains both concrete implementation-level details and abstract interface-level details. At the implementation-level, an MBO contains read-only result fields that contain metadata about the data in the implementation, and parameters that are passed to the back-end data source. At the interface-level, an MBO contains attributes that map to result fields, which correspond to client properties. An MBO may have operations, which can also contain parameters that map to arguments, and which determines how the client passes information to the enterprise information system (EIS).

You can define relationships between MBOs, and link attributes and parameters in one MBO to attributes and parameters in another MBO.

MBO attribute – An MBO attribute is a field that can hold data. You can map an MBO attribute to a result field in a back-end data source; for example, a result field in a database table.

MBO binding – An MBO binding links MBO attributes and operations to a physical data source through a connection profile.

MBO operation – An MBO operation can be invoked from a client application to perform a task; for example, create, delete, or update data in the EIS.

MBO relationship – MBO relationships are analogous to links created by foreign keys in a relational database. For example, the account MBO has a field called *owner_ID* that maps to the *ID* field in the owner MBO.

Define MBO relationships to facilitate:

- Data synchronization
- EIS data-refresh policy

messaging based synchronization (MBS) – A synchronization method where data is delivered asynchronously using a secure, reliable messaging protocol. This method provides fine-grained synchronization (synchronization is provided at the data level—each process communicates only with the process it depends on), and it is therefore assumed that the device is always connected and available. *See also* synchronization.

messaging database – The messaging database allows in-flight messages to be stored until they can be delivered. This database is used in a messaging based synchronization environment. The messaging database is part of the Unwired Platform data tier, along with the cache, cluster, and monitoring databases.

mobile application – A Sybase Unwired Platform mobile application is an end-to-end application, which includes the MBO definition (back-end data connection, attributes, operations, and relationships), the generated server-side code, and the client-side application code.

Mobile Application Diagram – The Mobile Application Diagram is the graphical interface to create and edit MBOs. By dragging and dropping a data source onto the Mobile Application Diagram, you can create a mobile business object and generate its attribute mappings automatically.

Mobile Application Project – A collection of MBOs and client-side, design-time artifacts that make up a mobile application.

mobile workflow packages – Mobile workflow packages use the message-based synchronization model. The mobile workflow packages are deployed to Unwired Server, and can be deployed to mobile devices, via the Unwired Platform administrative perspective in Sybase Control Center.

monitoring – Monitoring is an Unwired Platform feature available in Sybase Control Center that allows administrators to identify key areas of weakness or periods of high activity in the particular area they are monitoring. It can be used for system diagnostic or for

troubleshooting. Monitored operations include replication-based synchronization, messaging-based synchronization, messaging queue, data change notification, device notification, package, user, and cache activity.

monitoring database – A database that exclusively stores data related to replication and messaging synchronization, queues status, users, data change notifications, and device notifications activities. By default, the monitoring database runs in the same data tier as the cache database, messaging database and cluster database.

monitoring profiles – Monitoring profiles specify a monitoring schedule for a particular group of packages. These profiles let administrators collect granular data on which to base domain maintenance and configuration decisions.

multitenancy – The ability to host multiple tenants in one Unwired Cluster. Also known as hostability. *See also* domains.

node – A host or server computer upon which one or more runtime components have been installed.

object query – Defined in Unwired WorkSpace for an MBO and used to filter data that is downloaded to the device.

onboarding – The enterprise-level activation of an authentic device, a user, and an application entity as a combination, in Unwired Server.

operation – *See* MBO operation.

package – A package is a named container for one or more MBOs. On Unwired Server a package contains MBOs that have been deployed to this instance of the server.

palette – In Unwired WorkSpace, the palette is the graphical interface view from which you can add MBOs, local business objects, structures, relationships, attributes, and operations to the Mobile Application Diagram.

parameter – A parameter is a value that is passed to an operation/method. The operation uses the value to determine the output. When you create an MBO, you can map MBO parameters to data-source arguments. For example, if a data source looks up population based on a state abbreviation, the MBO gets the state from the user, then passes it (as a parameter) to the data source to retrieve the information. Parameters can be:

- Synchronization parameters – synchronize a device application based on the value of the parameter.
- Load parameters – perform a data refresh based on the value of the parameter.
- Operation parameters – MBO operations contain parameters that map to data source arguments. Operation parameters determine how the client passes information to the enterprise information system (EIS).

personalization key – A personalization key allows a mobile device user to specify attribute values that are used as parameters for selecting data from a data source. Personalization keys

are also used as operation parameters. Personalization keys are set at the package level. There are three type of personalization keys: Transient, client, server.

They are most useful when they are used in multiple places within a mobile application, or in multiple mobile applications on the same server. Personalization keys may include attributes such as name, address, zip code, currency, location, customer list, and so forth.

perspective – A named tab in Sybase Control Center that contains a collection of managed resources (such as servers) and a set of views associated with those resources. The views in a perspective are chosen by users of the perspective. You can create as many perspectives as you need and customize them to monitor and manage your resources.

Perspectives allow you to group resources ways that make sense in your environment—by location, department, or project, for example.

physical role – A security provider group or role that is used to control access to Unwired Server resources.

Problems view – In Eclipse, the Problems view displays errors or warnings for the Mobile Application Project.

provisioning – The process of setting up a mobile device with required runtimes and device applications. Depending on the synchronization model used and depending on whether or not the device is also an Afaria client, the files and data required to provision the device varies.

pull synchronization – Pull synchronization is initiated by a remote client to synchronize the local database with the CDB. On Windows Mobile, pull synchronization is supported only in RBS applications.

push synchronization – Push is the server-initiated process of downloading data from Unwired Server to a remote client, at defined intervals, or based upon the occurrence of an event.

queue – In-flight messages for a messaging application are saved in a queue. A queue is a list of pending activities. The server then sends messages to specific destinations in the order that they appear in the queue. The depth of the queue indicates how many messages are waiting to be delivered.

relationship – *See* MBO relationship.

relay server – *See also* Sybase Hosted Relay Service.

resource – A unique Sybase product component (such as a server) or a subcomponent.

REST web services – Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

RFC – Remote Function Call. You can use the RFC interface to write applications that communicate with SAP R/3 applications and databases. An RFC is a standalone function. Developers use SAP tools to write the Advanced Business Application Programming (ABAP)

code that implements the logic of a function, and then mark it as "remotely callable," which turns an ABAP function into an RFC.

role – Roles control access to Sybase Unwired Platform resources. *See also* logical role and physical role.

role mapping – Maps a physical (server role) to a logical (Unwired Platform role). Role mappings can be defined by developers, when they deploy an MBO package to a development Unwired Server, or by platform or domain administrators when they assign a security configuration to a domain or deploy a package to a production Unwired Server (and thereby override the domain-wide settings in the security configuration).

RSOE – Relay Server Outbound Enabler. An RSOE is an application that manages communication between Unwired Server and a relay server.

runtime server – An instance of Unwired Server that is running. Typically, a reference to the runtime server implies a connection to it.

SAP – SAP is one of the EIS types that Unwired Platform supports.

SCC – Sybase Control Center. A Web-based interface that allows you to administer your installed Sybase products.

schedule – The definition of a task (such as the collection of a set of statistics) and the time interval at which the task must execute in Sybase Control Center.

scheduled data refresh – Data is updated in the cache database from a back-end EIS, based on a scheduled data refresh. Typically, data is retrieved from an EIS (for example, SAP) when a device user synchronizes. However, if an administrator wants the data to be preloaded for a mobile business object, a data refresh can be scheduled so that data is saved locally in a cache. By preloading data with a scheduled refresh, the data is available in the information server when a user synchronizes data from a device. Scheduled data refresh requires that an administrator define a cache group as "scheduled" (as opposed to "on-demand").

security configuration – Part of the application user and administration user security. A security configuration determines the scope of user identity, authentication and authorization checks, and can be assigned to one or more domains by the platform administrator in Sybase Control Center. A security configuration contains:

- A set of configured security providers (for example LDAP) to which authentication, authorization, attribution is delegated.
- Role mappings (which can be specified at the domain or package level)

security provider – A security provider and its repository holds information about the users, security roles, security policies, and credentials used by some to provide security services to Unwired Platform. A security provider is part of a security configuration.

security profile – Part of the Unwired Server runtime component security. A security profile includes encryption metadata to capture certificate alias and the type of authentication used by server components. By using a security profile, the administrator creates a secured port over which components communicate.

server connection – The connection between Unwired WorkSpace and a back-end EIS is called a server connection.

server farm – *See also* cluster. Is the relay server designation for a cluster.

server-initiated synchronization – *See* push synchronization.

SOAP – Simple Object Access Protocol. SOAP is an XML-based protocol that enables applications to exchange information over HTTP. SOAP is used when Unwired Server communicates with a Web service.

solution – In Visual Studio, a solution is the high-level local workspace that contains the projects users create.

Solution Explorer – In Visual Studio, the Solution Explorer pane displays the active projects in a tree view.

SSO – Single sign-on. SSO is a credential-based authentication mechanism.

statistics – In Unwired Platform, the information collected by the monitoring database to determine if your system is running as efficiently as possible. Statistics can be current or historical. Current or historical data can be used to determine system availability or performance. Performance statistics are known as key performance indicators (KPI).

Start Page – In Visual Studio, the Start Page is the first page that displays when you launch the application.

structured data – Structured data can be displayed in a table with columns and labels.

structure object – Defined in Unwired WorkSpace, structures hold complex datatypes, for example, a table input to a SAP operation.

subscription – A subscription defines how data is transferred between a user's mobile device and Unwired Server. Subscriptions are used to notify a device user of data changes, then these updates are pushed to the user's mobile device.

Sybase Control Center – Sybase Control Center is the Flash-based Web application that includes a management framework for multiple Sybase server products, including Unwired Platform. Using the Unwired Platform administration perspective in Sybase Control Center, you can register clusters to manage Unwired Server, manage domains, security configurations, users, devices, connections, as well as monitor the environment. You can also deploy and MBO or workflow packages, as well as register applications and define templates for them. Only use the features and documentation for Unwired Platform. Default features and documentation in Sybase Control Center do not always apply to the Unwired Platform use case.

Sybase Control Center X.X Service – Provides runtime services to manage, monitor, and control distributed Sybase resources. The service must be running for Sybase Control Center to run. Previously called Sybase Unified Agent.

Sybase Hosted Relay Service – The Sybase Hosted Relay Service is a Web-hosted relay server that enables you to test your Unwired Platform development system.

Sybase Messaging Service – The synchronization service that facilitates communication with device client applications.

Sybase Unwired Platform – Sybase Unwired Platform is a development and administrative platform that enables you to mobilize your enterprise. With Unwired Platform, you can develop mobile business objects in the Unwired WorkSpace development environment, connect to structured and unstructured data sources, develop mobile applications, deploy mobile business objects and applications to Unwired Server, which manages messaging and data services between your data sources and your mobile devices.

Sybase Unwired WorkSpace – Sybase Unwired Platform includes Unwired WorkSpace, which is a development tool for creating mobile business objects and mobile applications.

synchronization – A synchronization method where data is delivered synchronously using an upload/download pattern. For push-enabled clients, synchronization uses a "poke-pull" model, where a notification is pushed to the device (poke), and the device fetches the content (pull), and is assumed that the device is not always connected to the network and can operate in a disconnected mode and still be productive. For clients that are not push-enabled, the default synchronization model is pull. *See also* messaging based synchronization.

synchronization group – Defined in Unwired WorkSpace, a synchronization group is a collection of MBOs that are synchronized at the same time.

synchronization parameter – A synchronization parameter is an MBO attribute used to filter and synchronize data between a mobile device and Unwired Server.

synchronization phase – For replication based synchronization packages, the phase can be an upload event (from device to the Unwired Server cache database) or download event (from the cache database to the device).

synchronize – *See also* data refresh. Synchronization is the process by which data consistency and population is achieved between remote disconnected clients and Unwired Server.

truststore – The location in which certificate authority (CA) signing certificates are stored. *See also* keystore.

undeploy – Running **undeploy** removes a domain package from an Unwired Server.

Unwired Server – The application server included with the Sybase Unwired Platform product that manages mobile applications, back-end EIS synchronization, communication, security, transactions, and scheduling.

user – Sybase Control Center displays the mobile-device users who are registered with the server.

view – A window in a perspective that displays information about one or more managed resources. Some views also let you interact with managed resources or with Sybase Control Center itself. For example, the Perspective Resources view lists all the resources managed by the current perspective. Other views allow you to configure alerts, view the topology of a replication environment, and graph performance statistics.

Visual Studio – Microsoft Visual Studio is an integrated development environment product that you can use to develop device applications from generated Unwired WorkSpace code.

Welcome page – In Eclipse, the first set of pages that display when you launch the application.

workspace – In Eclipse, a workspace is the directory on your local machine where Eclipse stores the projects that you create.

WorkSpace Navigator – In Eclipse, the tree view that displays your mobile application projects.

WSDL file – Web Service Definition Language file. The file that describes the Web service interface that allows clients to communicate with the Web service. When you create a Web service connection for a mobile business object, you enter the location of a WSDL file in the URL.

Index

A

application types 10
 attributes
 filtering 174

B

BAPI exposed as Web service
 MBO 106
 bulk loads 29

C

cache
 update groups 15
 update policies 15
 cache groups
 assigned 193
 creating 190
 defining 188
 modify 191
 cache loading strategies
 bulk loads 29
 cache update policy
 definition 195
 requirements 198
 setting 196
 collecting troubleshooting information 6
 complex types
 editing 135
 configure
 Unwired Platform 33

D

data
 change notifications
 See DCN
 refresh schedule 15
 data change notifications
 See DCN
 data sources 9
 DCN 15
 defining an MBO
 for real-time data lookup 187

deploying
 result checker classes 117
 developer profile
 advanced 77
 basic 77
 properties 53
 switching 76
 documentation roadmap 3

E

EIS
 data sources 9

G

glossaries
 Sybase Unwired Platform terms 263

H

Hybrid Web Container 10

L

list types
 editing 136

M

MBOs
 mobile business object 70
 overview 70
 mobile business objects
 accessing from an HTTPS port 107
 local 84
 multiple objects from a single source 80
 SAP BAPIs exposes as Web services 106
 SAP connection properties as MBO parameters
 99
 See also MBOs
 multilevel insert operations
 creating for Web-service MBOs 104

Index

O

- Object API code
 - generating 230
- object queries
 - defining manually 204
 - definition 202
 - generating from primary key attributes 203
 - guidelines 205

P

- parameter
 - adding to a mobile business object 177
- parameters
 - and partitioned cache 184
 - and stored procedures 182
 - synchronization
 - datetime and time restrictions 175
 - usage 180
- parameters, load
 - filtering 181
- partition key 182
- postinstallation configuration 32

R

- refactoring a result checker 118
- REST web service
 - binding to 108
- result checker 113
- result checker classes, deploying 117
- result checkers
 - deleting 118
 - moving 118
 - renaming 118

S

- SAP
 - configuring Unwired Platform components for 41, 42
 - operations, committing 98
- SAP BAPI
 - exposed as a Web service 107

- SAP connection properties
 - as MBO parameters 99
- single sign-on for SAP 87
- SSO for SAP data source 87
- stored procedures
 - and parameters 182
- sup_ec 77, 131, 134
- Sybase Unwired WorkSpace
 - starting 31
- synchronization groups
 - creating 175
 - definition 175
 - deleting 176
- synchronization schedule
 - defining 173

T

- terms
 - Sybase Unwired Platform 263
- troubleshooting information 6

U

- Unwired Platform components
 - starting 31
 - stopping 31

W

- web service
 - REST 45
- Web service MBO
 - limitations 105, 112
- Web service MBOs
 - multilevel insert operations, creating for 104

X

- X.509
 - installing libraries 41, 42