SAP Mobile WorkSpace: Mobile Business Object Development

SAP Mobile Platform 2.3

DOCUMENT ID: DC01909-01-0230-01
LAST REVISED: February 2013

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

Contents

Develop Mobile Business Objects

Use SAP® Mobile Platform to develop mobile business objects (MBOs), and generate Object API code that can be used to create native device applications and Hybrid Apps.

Mobile business objects help form the business logic for mobile applications and Hybrid Apps by defining the data you want to use from your backend system and to expose through your mobile application or Hybrid Apps, 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 SAP Mobile Platform implements the concepts in your enterprise.

Product Task Flow

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

Developers Use SAP Mobile 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 SAP Mobile 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 SAP Mobile Server.
3. Create mobile business objects.

   Use SAP Mobile 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. 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.
4. Create device applications or Hybrid Apps:
   a. Generate client object API code in Eclipse, then develop the device application in a native IDE. Implement error handling.
   b. Use Hybrid App Forms Editor to develop a message-based Hybrid App package.
5. Deploy the mobile application project from SAP Mobile WorkSpace to SAP Mobile Server.

6. Deploy the device application (which contains MBOs) to an emulator or mobile device, and test.

System Administrators Use the SAP Control Center administrative console, a Web-based user interface to configure and deploy mobile applications and Hybrid App packages from SAP Mobile 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. SAP Mobile Server manages synchronization between the data source and the mobile device.
   Optionally use Afaria® with SAP Mobile 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 SAP Mobile Server to the data source; and create, update, and delete data records and transactions.
- Use Hybrid App forms to carry out steps in a business process from the mobile device.

Tutorials

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

Check the Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories*, then navigate to the most current version.

Check SDN regularly for tutorials, projects and white papers: *http://scn.sap.com/docs/DOC-8803*.

Samples

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

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

Documentation Roadmap for SAP Mobile Platform

SAP® Mobile Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories*, then navigate to the most current version.

Understanding the SAP Mobile Platform Development Environment

This provides basic information for understanding the SAP Mobile Platform development environment.

Developing with Multiple Tools

Tools within SAP Mobile 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

SAP Mobile Platform provides an Eclipse development environment. The SAP Mobile 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 iOS, Android, BlackBerry, Windows Mobile, and Windows. Develop the user interface in its native integrated development environment (IDE).

Use SAP Mobile WorkSpace to develop Hybrid Apps.

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 SAP Mobile Platform.

Learning SAP Mobile WorkSpace Basics

SAP Mobile WorkSpace features are well integrated in the Eclipse IDE. If you are unfamiliar with Eclipse, you can quickly learn the basic layout of SAP Mobile WorkSpace and the location of online help.

- To access the online help, select **Help > Help Contents**. Some documents are for SAP Mobile WorkSpace, while others are for the Eclipse development environment.
- The Welcome page provides links to useful information to get you started.
  - To close the Welcome page, click **X**.
  - Reopen the Welcome page by selecting **Help** > **Welcome**.
  - To learn about tasks you must perform, select the **Development Process** icon.
- In SAP Mobile WorkSpace, look at the area (window or view) that you will use to access, create, define, and update mobile business objects (MBOs).

Window	Description
WorkSpace Navigator view	Use this view to create Mobile Application projects, and review and modify MBO-related properties. This view displays mobile application project folders, each of which contains all project-related resources in subfolders, including MBOs, datasource references to which the MBOs are bound, personalization keys, and so on.
Enterprise Explorer view	A view that provides functionality to connect to various enterprise information systems (EIS), such as database servers, SAP® back ends, and SAP Mobile Server.

Window	Description
Mobile Application Diagram	The Mobile Application Diagram is a graphical editor where you create and define mobile business objects. Use the Mobile Application Diagram to create MBOs (including attributes and operations), then define relationships with other MBOs. You can: • Create MBOs in the Mobile Application Diagram using Palette icons and menu selections – either bind or defer binding to a datasource, when creating an MBO. For example, you may want to model your MBOs before creating the datasources to which they bind. This MBO development method is sometimes referred to as the top-down approach. • Drag and drop items from Enterprise Explorer to the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the datasource artifact being dropped on the Mobile Application Diagram. Each new mobile application project generates an associated mobile application diagram.
Palette	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.
Properties view	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.
Outline view	Displays an outline of the active file and lists structural elements. The contents are editor-specific.

Window	Description
Problems view	Displays validation errors or warnings that you may encounter in addition to errors in the Diagram editor and Properties view. Follow warning and error messages to adjust MBO properties and configurations to avoid problems, and use as a valuable source for collecting troubleshooting information when reporting issues to Customer Service and Support.
Error Log view	Displays error log information. This is a valuable source for collecting troubleshooting information.

See also

- *Basic and Advanced Developer Profiles* on page 9
- *Mobile Business Objects* on page 10
- *Datasources* on page 10
- *Device Application Types* on page 11
- *Deployment to SAP Mobile Server* on page 12
- *Data Synchronization and Data Refresh* on page 12

Basic and Advanced Developer Profiles

Optionally you can set Basic and Advanced (the default) developer profile preferences for SAP Mobile 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 SAP Mobile WorkSpace features, wizards, and properties, enabling additional MBO customization not provided in the Basic profile.

See also

- *Learning SAP Mobile WorkSpace Basics* on page 7
- *Mobile Business Objects* on page 10
- *Datasources* on page 10
- *Device Application Types* on page 11
- *Deployment to SAP Mobile Server* on page 12
- *Data Synchronization and Data Refresh* on page 12

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 SAP Mobile 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 SAP Mobile WorkSpace Basics* on page 7
- *Basic and Advanced Developer Profiles* on page 9
- *Datasources* on page 10
- *Device Application Types* on page 11
- *Deployment to SAP Mobile Server* on page 12
- *Data Synchronization and Data Refresh* on page 12

Datasources

A datasource 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 datasource. Connection profiles are created to specific datasources 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 SAP Mobile Server.

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

SAP Mobile Platform supports multiple EIS connection types. See *Supported Hardware and Software* for information.

See also

- *Learning SAP Mobile WorkSpace Basics* on page 7

- *Basic and Advanced Developer Profiles* on page 9
- *Mobile Business Objects* on page 10
- *Device Application Types* on page 11
- *Deployment to SAP Mobile Server* on page 12
- *Data Synchronization and Data Refresh* on page 12

Device Application Types

SAP Mobile Platform supports two mobile business object-based application types: native application and Hybrid Web Container-based Hybrid App.

Native Application

The native application model enables the developer to generate Object API code from mobile business objects, and 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 Hybrid App

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 package of platform-independent HTML, JavaScript and CSS resources can be deployed automatically to the Hybrid Web Container, a native application on the device, without writing any code. The Hybrid Web Container hosts an embedded browser and launches the individual Hybrid App. The Hybrid Apps are assigned to users by administrators. Once assigned, those Hybrid Apps can be initiated by the user (client-initiated) or automatically triggered as a result of a back-end event that is sent to the SAP Mobile Server as a data change notification request (server-initiated).

See also

- *Learning SAP Mobile WorkSpace Basics* on page 7
- *Basic and Advanced Developer Profiles* on page 9
- *Mobile Business Objects* on page 10
- *Datasources* on page 10
- *Deployment to SAP Mobile Server* on page 12
- *Data Synchronization and Data Refresh* on page 12

Deployment to SAP Mobile Server

Deploy mobile business objects (MBOs) in a Mobile Application as a deployment package to SAP Mobile 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

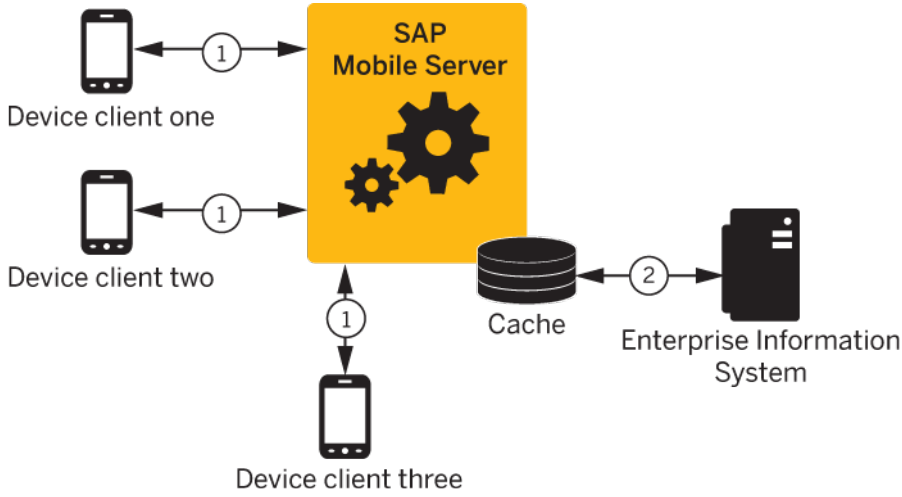
- *Learning SAP Mobile WorkSpace Basics* on page 7
- *Basic and Advanced Developer Profiles* on page 9
- *Mobile Business Objects* on page 10
- *Datasources* on page 10
- *Device Application Types* on page 11
- *Data Synchronization and Data Refresh* on page 12

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 SAP Mobile 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 SAP Mobile 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.



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 SAP Mobile WorkSpace Basics* on page 7
- *Basic and Advanced Developer Profiles* on page 9
- *Mobile Business Objects* on page 10
- *Datasources* on page 10
- *Device Application Types* on page 11
- *Deployment to SAP Mobile Server* on page 12

SAP Mobile Server Cache

The SAP Mobile 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 SAP Mobile Server and device applications, and refreshed data between SAP Mobile Server and the enterprise information system (EIS). Administrators can control cache behaviours in SAP 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 device client specific parameters. When a device application passes a device client specific 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, SAP Mobile 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 SAP Mobile 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 SAP Mobile 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 14
- *Synchronization and Data Refresh Data Flow* on page 17
- *Data Refresh Data Flow* on page 21
- *HTML5/JS Hybrid App Data Flow* on page 24
- *Synchronization and Data Refresh Strategies* on page 26

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

- *SAP Mobile Server Cache* on page 13
- *Synchronization and Data Refresh Data Flow* on page 17
- *Data Refresh Data Flow* on page 21
- *HTML5/JS Hybrid App Data Flow* on page 24
- *Synchronization and Data Refresh Strategies* on page 26

Synchronization Triggers

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

Table 1. Synchronization Methods and Triggers

Method	Description

Push	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:
	• Notifications are sent at a set interval. The default is one minute and ends when the client acknowledges it has received notification.
	• SAP Mobile 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 SAP Mobile Server does not force a synchronization, device application logic determines how to respond to the push notification. The device application developer can:
	  • Register to receive push notifications from SAP Mobile 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.
	Note: SAP Mobile Server initiates notifications only for replicated-based synchronization, while messaging-based synchronization pushes data to the device without notification.

Method	Description
Pull	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: • 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 (for example, by using the system personalization keys "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 SAP Mobile Server settings.

Table 2. Data Refresh Methods and Triggers

Method	Description
DCN (development) | Push Listener (administration)	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: • Notify SAP Mobile 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	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.
Operation cache policies	The MBO developer can add a cache policy to create, update, or delete operations to control how any EIS affecting operation is applied to the CDB.

Method	Description
Load arguments	The MBO developer can create an MBO that uses load arguments to: • Control data refresh for individual MBOs • Create client defined CDB partitions for individual users based on argument values. • Control synchronization if paired with a synchronization parameter.

Synchronization and Data Refresh Data Flow

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

SAP Mobile 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

- *SAP Mobile Server Cache* on page 13
- *Synchronization and Data Refresh Triggers* on page 14
- *Data Refresh Data Flow* on page 21
- *HTML5/JS Hybrid App Data Flow* on page 24
- *Synchronization and Data Refresh Strategies* on page 26

Synchronization Data Flow

Synchronization is when a device application's data is updated with the contents of the SAP Mobile 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.

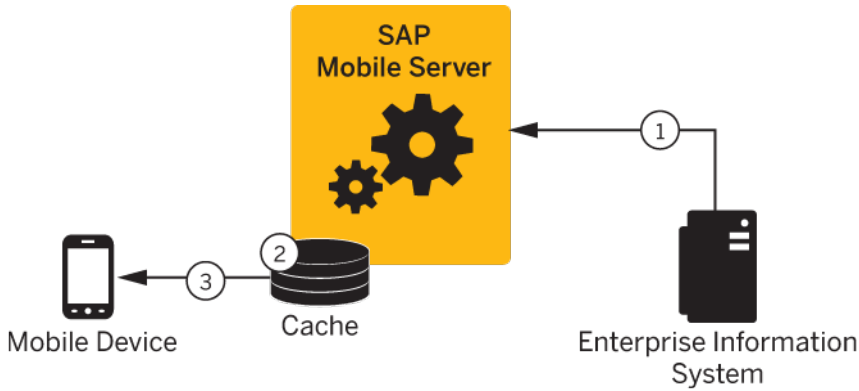
Filtering and Synchronizing Data

Use load arguments and synchronization parameters to synchronize selected subsets of data.

Result Set Filter Data Flow

A `ResultSetFilter` is a custom Java class deployed to SAP Mobile 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.



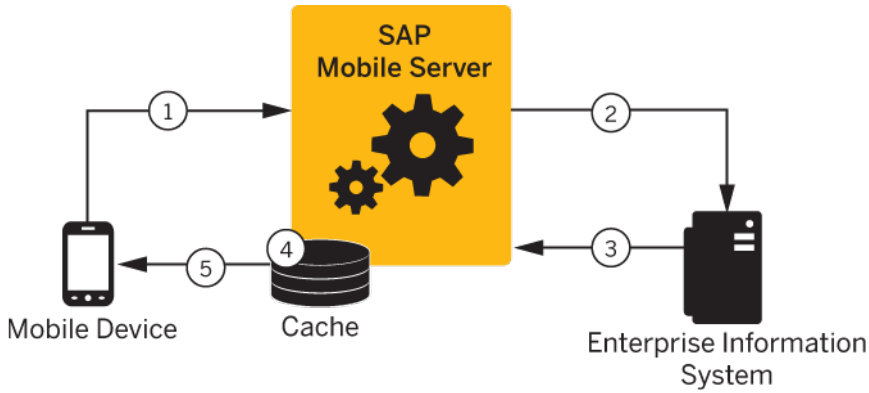
1. Enterprise information system (EIS) data is sent to SAP Mobile Server.
2. The result set filter filters the results, and applies those results to the CDB for a given MBO. For example, the result set filter combines two columns into one.
3. The device application synchronizes with the results contained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

Load Argument Data Flow

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

Pairing a load argument with a synchronization parameter during mobile business object (MBO) development, indicates that the user will supply values for this argument 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 argument is simply used to update the SAP Mobile 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 argument is desired to refine the data requested from the EIS. Mapping the load argument to a synchronization parameter partitions data in the CDB according to values sent from each device client (client defined partition).



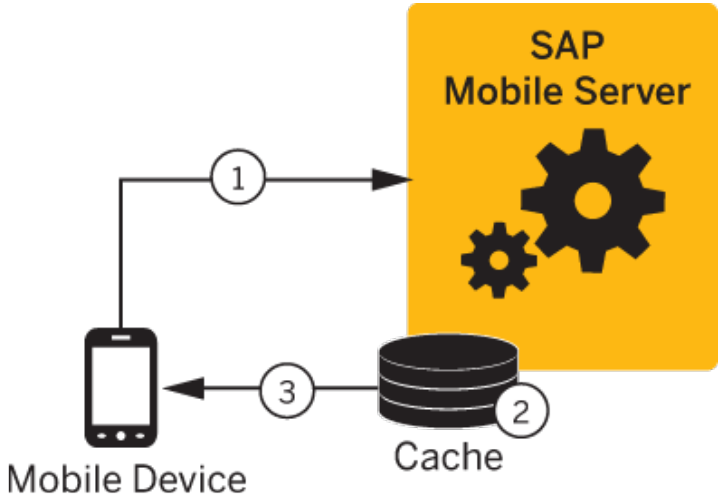
1. The user initiates a synchronization request and includes an argument value, for example, a user name. Be aware that passwords should not be used as arguments or parameters.
2. If personalization keys are used as the load argument, SAP Mobile Server passes the query to the EIS. If the arguments are user credentials, they are validated by the EIS.
3. The EIS refreshes SAP Mobile Server based on the argument value, for example, it refreshes data only for the validated user. If the argument is region, and the argument value is "western," only results for the western region refresh.
4. SAP Mobile 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. SAP Mobile 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.

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

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



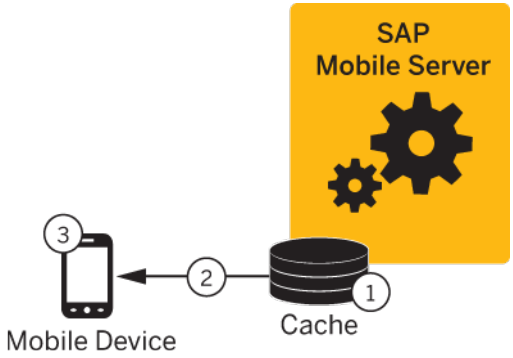
1. The user initiates a synchronization request that includes an attribute value (synchronization parameter).
2. SAP Mobile Server filters the data in the CDB. For example, if the attribute is "country" and the user supplied the value "USA," only rows that contain "USA" are returned to the device.

   If the user later supplies the value "Europe", rows for both "USA" and "Europe" are returned to the device, and so on.
3. SAP Mobile Server synchronizes the device with the results.

Synchronization Initiated by SAP Mobile Server

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

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



1. SAP Mobile Server detects a change in the data cache; for example, through a data change notification (DCN) or a data refresh.
2. SAP Mobile Server notifies registered devices of changes to cached MBO data. If it is configured to do so, SAP Mobile 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 SAP Mobile Server does not force a synchronization.

Data Refresh Data Flow

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

There are two general categories by which data refresh occurs:

- SAP Mobile Server pulls – pulls updates from the EIS either through SAP Mobile 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 SAP Mobile Server.

See also

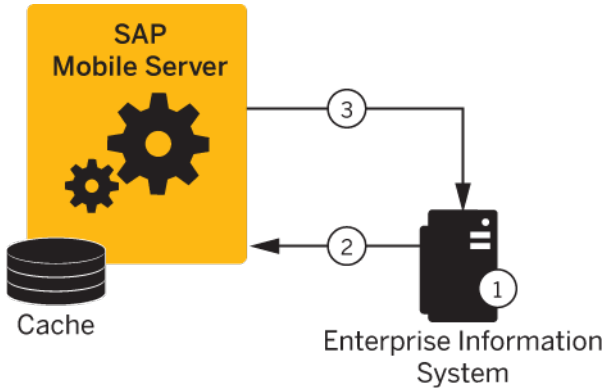
- *SAP Mobile Server Cache* on page 13
- *Synchronization and Data Refresh Triggers* on page 14
- *Synchronization and Data Refresh Data Flow* on page 17
- *HTML5/JS Hybrid App Data Flow* on page 24
- *Synchronization and Data Refresh Strategies* on page 26

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 SAP Mobile Server as HTTP GET or POST operations. Each DCN can instruct SAP Mobile Server to modify cached MBO data.

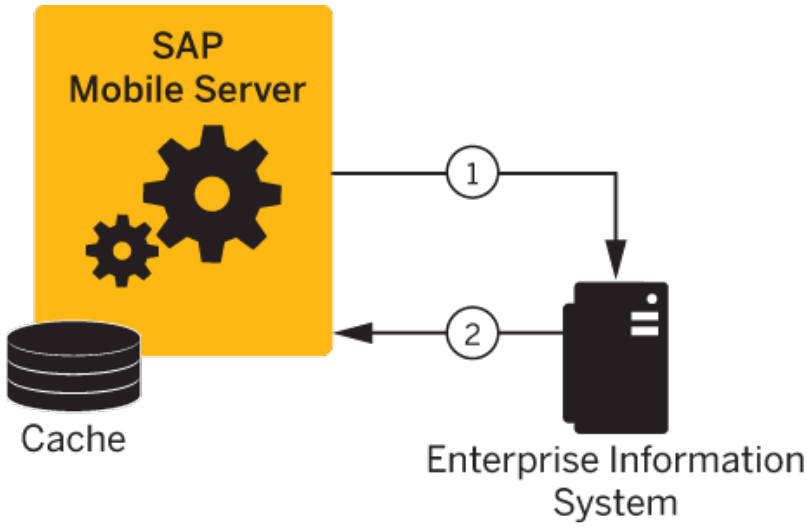
A DCN can be invoked by a database trigger, an EIS event, or an external process. DCNs are more complex to implement than other data refresh methods, but ensure that changes are immediately reflected in the cache.



1. An event initiates the DCN.
2. The DCN (HTTP POST or GET) is issued to SAP Mobile 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.



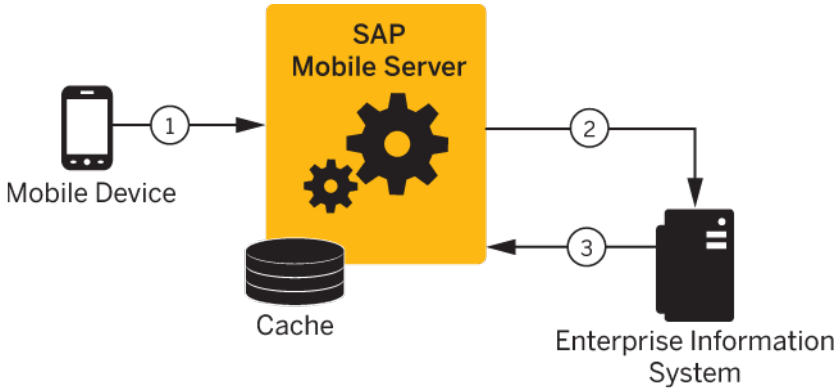
1. The deployed MBO triggers a cache update depending on the cache group to which it belongs.
2. The CDB is updated based on the cache group settings and policy. For example, On-demand, Scheduled, and Cache interval determine refresh timing for cache groups.

Note: Cache groups using an EIS managed policy do not follow the same data flow as described above, instead the EIS manages data refresh through DCN.

Operation Cache Policy Data Flow

An operation cache policy provides options by which data refresh is controlled for a given MBO create, update, or delete operation.

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.

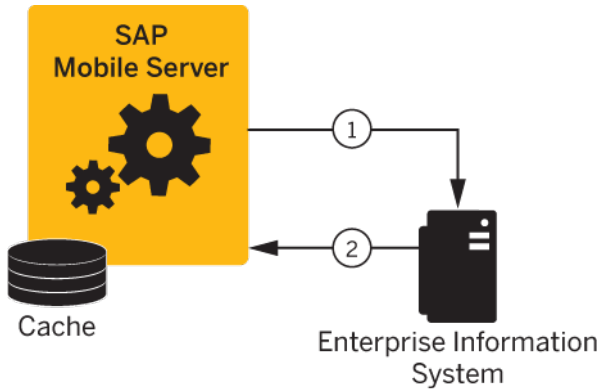


1. The device application initiates a create, update, or delete operation.
2. SAP Mobile Server passes the operation to the EIS, where the operation is executed.
3. The CDB is updated based on the operation cache policy.

Data Refresh Initiated by SAP Mobile Server

Configure SAP Mobile 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 SAP Mobile 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. SAP Mobile Server polls the EIS at an interval determined by the SAP Mobile Server administrator.
2. If data changes, the cache is refreshed.

HTML5/JS Hybrid App Data Flow

In the HTML5/JS Hybrid App model, when you invoke an MBO operation located on SAP Mobile Server using a Submit action, you can specify synchronous behavior (the messaging application waits for a successful or failed response from SAP Mobile Server before proceeding). From an MBO/EIS perspective, SAP Mobile Server updates are asynchronous (the server does not wait for a response).

Some of the differences between Hybrid Apps and replicated mobile applications include:

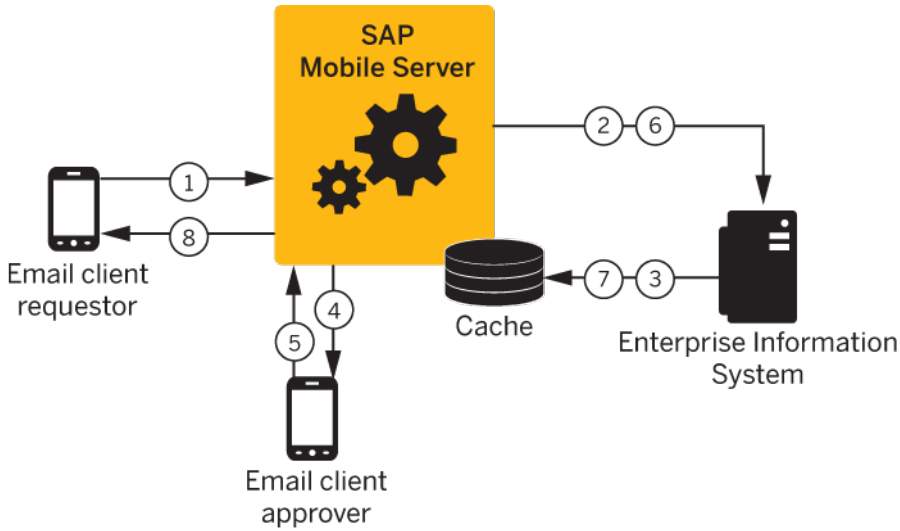
- The MBOs included in the application are no different than any other MBO: same types of parameters, attributes, CDB caching, synchronization methods, and so on.
- There is no permanent storage of the message portion of the Hybrid App. For example, a Hybrid App consists of the MBO portion, which is managed by SAP Mobile Server, and the message portion. The message portion 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 without them having to explicitly know or request it.

This Hybrid App example is a travel approval e-mail based 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 a message to the approver when a new row is inserted into the MBO table.

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 Server contacts the EIS.
3. The cache is updated.
4. Triggers a message to the approver.
5. The travel request is approved through e-mail.
6. The EIS is updated with the approved information.
7. Depending on the data refresh schedule or the operation's cache policy, the cache is updated.
8. The requester receives approval.

See also

- *SAP Mobile Server Cache* on page 13
- *Synchronization and Data Refresh Triggers* on page 14
- *Synchronization and Data Refresh Data Flow* on page 17
- *Data Refresh Data Flow* on page 21
- *Synchronization and Data Refresh Strategies* on page 26

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 SAP Mobile 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

- *SAP Mobile Server Cache* on page 13
- *Synchronization and Data Refresh Triggers* on page 14
- *Synchronization and Data Refresh Data Flow* on page 17
- *Data Refresh Data Flow* on page 21
- *HTML5/JS Hybrid App Data Flow* on page 24

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: • SAP Mobile 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.
Clients must immediately synchronize critical EIS data changes.	You could either implement a cache policy that immediately applies the results of the MBO operation to the CDB. Or, implement a data change notification (DCN) that performs a data refresh of targeted MBO data. SAP Mobile Server then sends notifications to registered device clients, and optionally forces a synchronization to targeted devices. 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. SAP Mobile Server forces synchronization with registered devices.
EIS data changes frequently.	Either: • Implement an SAP Mobile 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. 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.
Device application users change EIS data frequently.	When defining MBO create, update, and delete operations, include a cache policy for EIS modifying operations. Choose a policy that updates the CDB only with necessary changes. 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 "Immediately update the cache" policy when defining the MBO operation.

Scenario	Strategy
A mobile application supports thousands of individual users, each of whom has a set of user specific data.	The MBO developer uses system personalization keys that are used as client parameters ("username" and "password"), which are validated by the EIS. 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.
A mobile application has both static and changeable data.	The MBO developer configures two cache groups, designed to refresh SAP Mobile Server cache (CDB) for each MBO based on the frequency of EIS data changes to which each MBO is bound. 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: **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.	The message-based mobile application uses both replicated data (managed within the MBO) and nonreplicated messaging data. SAP Mobile Server pushes changes/updates to device application users. 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.

The Impact of Synchronization and Data Refresh

When designing and developing your mobile application solutions, consider the impact of various 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.
SAP Mobile Server scheduled data refresh	Easily implemented by the SAP Mobile Server administrator. Less targeted than DCN or a cache group. Uses more system resources since it must periodically query the EIS for changes.

Data refresh method	Implication
Cache group	Easily implemented by the MBO developer. A cache group is a collection of MBOs to which a common refresh policy is applied.
Operation Cache 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 argument	Easily implemented by the MBO developer. Load arguments filter the EIS data and fill the SAP Mobile Server cache database (CDB) with a subset of data from the EIS. They can be used alone or mapped to synchronization parameters to control both data refresh and synchronization.

Table 5. Impact of Various Synchronization Methods

Synchronization method	Implication
Initiated by SAP Mobile Server	Easily implemented by the SAP Mobile Server administrator. The primary consideration is balancing performance with resource usage when determining how frequently synchronization is initiated, and to how many registered devices: • 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.

Synchronization method	Implication
Initiated by the device	Options include: • Filtering results – implemented by the MBO and device application developer. The MBO can be configured through load arguments 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.
Defined during MBO modeling/design	Options include: • Synchronization group – groups MBOs by synchronization needs. • Synchronization tab – defines synchronization requirements for individual MBOs, including synchronization parameters.

Starting and Stopping SAP Mobile Platform Components

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

A set of Windows services supports SAP Mobile 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 *SAP Mobile Platform Windows Services* in *System Administration*.

Starting SAP Mobile WorkSpace

Start SAP Mobile 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 SAP Mobile Server is not running, you can still create and edit MBOs and generate code, but you cannot deploy MBOs.

1. From Windows, select **Start > Programs > SAP > SAP Mobile Platform<*version*> > SAP Mobile WorkSpace**.

   Create a new workspace for SAP Mobile WorkSpace Eclipse Edition the first time you launch it.
2. If you cannot start or stop SAP Mobile Platform Server services through the Windows Start menu, see*SAP Mobile Server Fails to Start* in *Troubleshooting*. .

See also

- *Starting SAP Mobile Platform Services* on page 31

Starting SAP Mobile Platform Services

Start SAP Mobile Server, the sample database, the cache database (CDB), and other essential services from the Windows Start menu.
In Windows, select **Start > Programs > SAP > SAP Mobile Platform > Start SAP Mobile Platform Services**.

The SAP Mobile Server services enable you to access the SAP Mobile Platform runtime components and resources.

See also

- *Starting SAP Mobile WorkSpace* on page 31

Configure

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

Configure - Eclipse Development Environment

Configure your Eclipse Edition development environment by connecting to SAP Mobile 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 an SAP Mobile Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing and Exporting Connection Profiles and Projects* on page 61
- *Importing the Public Certificate* on page 64

Connection Profiles

A connection profile contains the connection property information needed to connect to an enterprise information system (EIS) data source/server. For example, a database server, SAP server, or an application server hosting Web services. 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
- *Creating an SAP Connection Profile* on page 40
- *Creating a Web Service Connection Profile* on page 44

- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *Connecting to a Connection Profile* on page 49
- *Creating an SAP Mobile Server Connection Profile* on page 50
- *Testing a Connection Profile* on page 48

Creating a Database Connection Profile

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

1. In the Enterprise Explorer, right-click **Database Connections** and select **New**.
2. On the Wizard Selection page, select an option from this list and click **Next**:
   - DB2 for Linux, UNIX, and Windows
   - DB2 for i5/OS
   - DB2 for z/OS
   - Generic JDBC
   - 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.

   If you select the **Optional** Properties tab, and define a property that already exists, the **Add** button is disabled. That is, SAP Mobile WorkSpace prevents the MBO developer from defining the same property multiple times.

See also

- *Connection Profiles* on page 33
- *Creating an SAP Connection Profile* on page 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *Connecting to a Connection Profile* on page 49

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 4.0	Go to *http://msdn.microsoft.com/en-us/data/aa937724.aspx* and select *Download the Microsoft JDBC Driver 4.0 for SQL Server*.

2. Shut down SAP Mobile WorkSpace and SAP Mobile Platform Services.
3. For SAP Mobile WorkSpace, put the driver in the correct location.

JDBC driver for:	Action
Oracle	Place the JDBC driver, for example `ojdbc14.jar`, in: *SMP_HOME*`\MobileSDK<Version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_<version>.<plugin version number>\lib`
DB2	Unzip the `db2JdbcJars.zip` file and copy the JAR files to: *SMP_HOME*`\MobileSDK<Version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.db2_<version>.<plugin version number>\lib`
SQL Server JDBC driver 3.0	Copy `sqljdbc4.jar` to: *SMP_HOME*`\MobileSDK<Version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.sqlserver_<version>.<plugin version number>\lib`

4. Copy the appropriate JAR file to the specified SAP Mobile Server location.

JAR file for:	Action
Oracle	Copy `ojdbc14.jar` to the server location: *SMP_HOME* `\Servers\UnwiredServer\lib\3rdparty`
DB2	Copy the JAR files, for example `db2jcc.jar` and `db2jcc_license_cu.jar`, to the server location: *SMP_HOME* `\Servers\UnwiredServer\lib\3rdparty`
SQL Server JDBC driver 3.0	Copy `sqljdbc4.jar` to the server location: *SMP_HOME* `\Servers\UnwiredServer\lib\3rdparty`

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 SAP Mobile WorkSpace and SAP Mobile Platform Services.

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:

SMP_HOME`\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.db2_<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 *SMP_HOME*`\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.db2_<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 6. 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.
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 SAP Mobile 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:

SMP_HOME`\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_.<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.
4. From the driver templates, choose Oracle Thin Driver.
5. From the Jar list tab, click **Edit Jar/zip** and verify that `ojdbc14.jar` points to *SMP_HOME*`\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_<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 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 SAP Mobile WorkSpace to connect automatically to the database (when it is started) and display its contents in Enterprise Explorer.

Field	Description
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.

Installing the Oracle ojdbc6 Driver

This procedure describes how to use the Oracle ojdbc6.jar as the driver to connect to Oracle data sources.

1. Download the Oracle `ojdbc6.jar` file.
2. Edit the `MANIFEST.MF` file located in the `SMP_HOME\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_<version>.<timestamp>\META-INF` directory, and replace this line:

   ```
   Bundle-ClassPath: lib/ojdbc14.jar
   ```

   with

   ```
   Bundle-ClassPath: lib/ojdbc6.jar
   ```

3. Save and close the file.
4. Copy the downloaded `ojdbc6.jar` file to `SMP_HOME\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.com.oracle_<version>.<timestamp>\lib`.
5. Perform a clean startup:
   a) Open a command prompt.
   b) Cd to `SMP_HOME\MobileSDK<version>\Eclipse`.
   c) Enter the command **MobileWorkSpace.bat -clean**
6. From SAP Mobile WorkSpace, verify the Oracle connection profile's properties, including the correct path to the `ojdbc6.jar` driver.
7. To enable the driver on SAP Mobile Server, copy the `ojdbc6.jar` file to the `SMP_HOME\Servers\UnwiredServer\lib\ext` directory.
8. You can now create a connection profile and connect to the Oracle server and create MBOs from the Oracle data source.

Creating an SAP Connection Profile

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

Prerequisites

On Windows 7, Windows Vista, and Windows Server 2008, the Microsoft `msvcr71.dll` and `msvcp71.dll` files must be installed in either the `<WINDOWS_HOME>\System32`(32-bit) or `<WINDOWS_HOME>\SysWOW64` (64-bit) directory.

Task

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 8. SAP Connection Properties Page Connection Tab

Field	Description
Create from properties file You can create the connection using these 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.	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. Example contents are: • jco.client.client=800 • jco.client.user=uepusr01 • jco.client.passwd=SAP123 • jco.client.ashost=i18nsap1 • jco.client.sysnr=01

Field	Description
Application Server	The name of the host on which the server resides. The default is "localhost". For SAP R3 systems that use a router, application server must have the format: /H/proxy_host/H/application_server where "H" is literal, and proxy_host and application_server are variables that represent the proxy host and application server respectively. For example: `/H/sapm20.anr.ms.test.com/S/3299/H/sapm20.anr.ms.test.com` **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.
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 9. SAP Connection Properties Page Advanced Tab

Field	Description
Default Code Page	Select this checkbox to specify 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. If you select **Other Properties**, and define a property that already exists, the **Add** button is disabled. That is, SAP Mobile WorkSpace prevents the MBO developer from defining the same property multiple times.

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

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

   Note: (Optional) When connected to an SAP connection profile, expand the data source to which you are connected, right-click either the root object, to refresh the entire data source, or an individual BAPI operation, and select **Refresh**. The refresh is required only if there is a change to the SAP data source.

See also

- *Binding an SAP Data Source to a Mobile Business Object* on page 90
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 102
- *Implementing SSO for SAP* on page 84

SAP External Libraries Requirements

Understand the requirements for external files you can optionally download from SAP and install into SAP Mobile Platform to enable communication with an SAP EIS.

- **SAP Cryptographic Libraries –** required by SAP Mobile Platform to enable Secure Network Communications (SNC) between SAP Mobile Server or SAP Mobile WorkSpace and the SAP EIS.
- **SAPCAR utility –** required to extract files from the SAP cryptographic library.

Installing the SAPCAR Utility

Unzip and install the latest **SAPCAR** utility on your SAP Mobile Server or SAP Mobile WorkSpace host. You can use **SAPCAR** 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.

1. Go to the SAP Web site at *http://service.sap.com/swdc* (login required).
2. From the SAP Download Center, navigate and log in to **Support Packages and Patches** > **Browse our Download Catalog > Additional Components**.
3. Select **SAPCAR**.
4. Select the current version, for example, **SAPCAR 7.20**, then download the appropriate **SAPCAR** for your platform.

Installing the SAP Cryptographic Libraries

Configure Secure Network Communications (SNC) for SAP Mobile Server SAP JCo connections. SNC may be required by your SAP EIS, if you are using SSO2 tokens or X.509 certificates 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 SAP Mobile 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 architecture of your machine.

1. Go to the SAP Web site at *http://service.sap.com/swdc* (requires login) and download the latest SAP cryptographic library suitable for your platform.
   a) Navigate to **Installations and Upgrades > Browse our Download Catalog > SAP Cryptographic Software > SAPCryptolib for Installation > SAPCRYPTOLIB <version>**.
   b) Select and download the platform-specific file.
2. Create a directory into which to unzip the Cryptographic zip file. For example: `C:\sapcryptolib`.
3. Copy the appropriate Windows cryptographic library for your machine (for example, SAPCRYPTOLIB<version>.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:\SAPCRYPTOLIB<version>.SAR -R C:\sapcryptolib
6. Copy the following into the `C:\sapcryptolib` directory:
   - For Itanium 64-bit processors, copy the `ntia64` subdirectory contents.
   - For Intel 64-bit processors, copy the `nt-x86_64` subdirectory contents.
   - For Intel 32-bit processors, copy the `ntintel` subdirectory contents.
7. Delete the corresponding subdirectory when files have been moved.
8. (Optional) Add the SECUDIR environment variable to the user environment batch file: *SMP_HOME*`\Servers\UnwiredServer\bin\usersetenv.bat`.
9. If you have installed SAP Mobile SDK, you must add the SECUDIR variable to the following batch file: *SMP_HOME*`\MobileSDK<version>\Eclipse\MobileWorkSpace.bat`.

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 10. Web Service Connection Details Page

Field	Description
Select the Web service from either a: • 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.

Field	Description
Preemptive authentication	Select the checkbox if this service requires authentication credentials to be sent as part of an initial request, instead of waiting for an HTTP 401 credentials challenge. The property is used for MBO Preview and Test Execute operations, and is also set in the deployment unit so it does not need to be explicitly set in SAP Control Center.

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

   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 40
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *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 11. 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, http://www.yourcompany.com/ could serve as the common prefix for the resource base URL. A base URL must begin with http:.

Field	Description
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: `customers/{id(int)}` where *int* is used to establish the datatype of the parameter.
Preemptive authentication	Select the checkbox if this service requires authentication credentials to be sent as part of an initial request, instead of waiting for an HTTP 401 credentials challenge. The property is used for MBO Preview and Test Execute operations, and is also set in the deployment unit so it does not need to be explicitly set in SAP Control Center.

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 40
- *Creating a Web Service Connection Profile* on page 44
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *Connecting to a Connection Profile* on page 49

Editing Connection Profile Properties

Edit the properties of a connection profile.

1. Go to Enterprise Explorer.
2. Expand one of the connection profile category/type folders.
3. Right-click a specific connection profile 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 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Renaming a Connection Profile* on page 47
- *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. Go to the 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**.
   The mobile application projects that reference this connection profile are refreshed automatically to reflect the data source reference changes.
5. Click **OK** to save those project changes.

   Be advised that you should not rename the "My SAP Mobile Server" or "My Sample Database" connection profiles. If you do, they are automatically recreated when you restart SAP Mobile WorkSpace.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *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 category folder under which you want the copied connection profile to appear and select **Paste**.
4. Enter a new name for the connection profile in the **Profile Name Input** dialog.

   A copy of the connection profile appears under the category you choose 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 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *Connecting to a Connection Profile* on page 49

Testing a Connection Profile

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

1. Go to 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 12. 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 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *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, SAP Mobile 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 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47

Deleting a Connection Profile

You can delete a connection profile.

Prerequisites

Disconnect from the database server.

Task

1. Go to 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. Also, if you delete the "My Mobile

Server" connection profile, it remains deleted only until you restart SAP Mobile WorkSpace, at which time it is recreated.

See also

- *Connection Profiles* on page 33
- *Creating a Database Connection Profile* on page 34
- *Creating an SAP Connection Profile* on page 40
- *Creating a Web Service Connection Profile* on page 44
- *Creating a REST Web Service Connection Profile* on page 45
- *Editing Connection Profile Properties* on page 46
- *Renaming a Connection Profile* on page 47
- *Connecting to a Connection Profile* on page 49

Creating an SAP Mobile Server Connection Profile

Create a new SAP Mobile Server connection profile to supply runtime connection information.

1. In Enterprise Explorer, right-click **Mobile 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 13. Specify Connection Details page

Field	Description
Host	The name of the host 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. For example, supAdmin.
Password	Enter the password for the server login, if required.
Save Password	Select the checkbox to save the password.

Field	Description
Secure	Select the checkbox to enable a secure connection with the server, which is different than a secure connection session. To establish a secure session with SAP Mobile Server: **1.** Enable **Secure management port** on SAP Mobile Server's administration listener. See the SAP Control Center help topic for information. **2.** Enter 2001 as the **Port**. **3.** Add the certificate file path to the `SMP_HOME\MobileSDKversion\Eclipse\MobileWorkSpace.bat` file. **Note:** Unless the SAP Mobile 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 SAP Mobile Server connection: "unable to find valid certification path to requested target".
Cluster information	The read-only properties for this Server's cluster (or non-cluster) SAP Mobile Server connection information. Connection information includes server host, protocol, and communication ports. Depending on whether the cluster uses a load balancer, you may alternatively set the name of the load balancer host.

4. Click **Next** to review a summary of this connection profile, or **Finish** to create the connection profile. The connection profile displays under Mobile 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 61
- *Importing the Public Certificate* on page 64

Preferences

Use preferences to configure the behavior and appearance of the Eclipse environment.

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 an SAP Mobile Server Connection Profile* on page 50
- *Importing and Exporting Connection Profiles and Projects* on page 61
- *Importing the Public Certificate* on page 64

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 **SAP AG > Mobile Development**.
3. In the Preferences dialog, configure preferences for:
   - Developer Profile
   - Logging
   - Miscellaneous
   - Mobile Application Diagram
   - Mobile Business Object
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 59
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Developer Profile Preferences

Set Basic and Advanced developer profile preferences for SAP Mobile WorkSpace.

SAP Mobile WorkSpace provides two developer profiles, Basic and Advanced (default). 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 14. Developer Profile Properties

Property	Description
Current profile	Select which profile to use during the current session. Advanced is the default.
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: • Cache – Advanced • Code generation – Both (cannot be modified) • Deployment – Both (cannot be modified) • Local business object – Advanced • Mobile business object – Both (cannot be modified) • 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 15. 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 wizard. 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 59
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Logging Preferences

Set mobile development logging preferences for logging events in the SAP Mobile 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 16. 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: • **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 17. 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: • **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 18. Mobile Development Logging: File

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

Property	Description
Log level	Select the level of verbosity for console logging: • **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 SAP Mobile 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 59
- *Mobile Development Miscellaneous Preferences* on page 56

Mobile Development Miscellaneous Preferences

Set miscellaneous mobile development preferences for the SAP Mobile WorkSpace environment.

Table 19. Miscellaneous Preferences

Property	Description
Version	Shows version number on mobile application project

Property	Description
Preview	Determines how a mobile business object component is previewed: • Show warning when executing – displays a pop-up message indicating Preview may affect EIS data. • 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 the Java Nature to the current project (if not already added) when you create a result set filter or result checker.
Actions	Determines whether or not the MBO Developer: • Confirms any refresh or remap actions, or delete actions that cascade to MBOs sharing the same read operation. • Is prompted to confirm removing input/output mappings from the root of a composite graph after certain actions • Is notified when a structure is generated with compatible fields for large object fields.
Code generation	• 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. **Note:** The default heap size is set to 128MB, but if you get errors, set the heap size larger than the default 128MB.

Property	Description
Object query	• 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.
Data length	Sets the default for various attribute and parameter datatype values: • Default string length – default preference for string length is 20 when String(%n) is selected. • Default binary length – default preference for a binary datatype length is 10 when Binary(%n) is selected. **Note:** If String or Binary is selected without "(%n)", SAP Mobile WorkSpace sets them to String(300) and Binary(32762) respectively, and warns the MBO developer to make adjustments to avoid wasting space or data truncation.
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	• 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 59

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

Show objects in diagram – selecting the option allows it to display in the Mobile Application Diagram:

- Structures – complex datatypes/structure MBOs. This option can also be enabled (disabled) directly from the Mobile Application diagram by right-clicking in the diagram and selecting **Show > Structures**.

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**, **Snap to grid for new diagrams**, or **Snap to shapes 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	• Choose the part to create when dragging a data source directly on the diagram:
	  • Prompt – you are prompted to create attributes and operations.
	  • Attributes – creates attributes only.
	  • Operation – creates an operation only.
	• Do not show Quick Create dialog again – create the MBO and operations without displaying the Quick Create dialog.
	• Do not show warning dialog for external database again – do not ask for confirmation when adding a mobile business object using a non-Sybase database. The message indicates there may be issues with generated SQL, which may require manual editing.
	• Do not show Role Assignments dialog again – when dropping a role onto an object in the diagram, assign the role to the object without showing the Role Assignments dialog.

Naming prefix	Assign a default prefix for newly created mobile business objects, attributes, operations, or client parameters.
Remove	Removes any associated structures along with the MBO when the MBO is deleted.

See also

- *Mobile Development Developer Profile Preferences* on page 53
- *Mobile Development Logging Preferences* on page 54
- *Mobile Application Diagram Preferences* on page 59
- *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 SAP Mobile WorkSpace and import them into another, which can be useful when troubleshooting various client and runtime issues.

See also

- *Creating a Data Source Connection Profile* on page 33
- *Creating an SAP Mobile Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing the Public Certificate* on page 64

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 SAP Mobile WorkSpace installations or when migrating to a more current version of SAP Mobile 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 a Project

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

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 SAP Mobile WorkSpace installations, or when migrating to a more current version of SAP Mobile WorkSpace, or as part of collecting troubleshooting information.

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.
2. Select the project you want to export in the WorkSpace Navigator, then select **File > Export** from the main menu bar.

   The Export wizard opens.
3. From the General node, select one of the following:

Table 20. Export Options

Option	Description
Archive file	Select this method to export resources to an archive file.
File system	Select this method to export resources to a file system directory.

4. Click **Next**.
5. Follow these steps, depending on the export destination:

Table 21. Export Destination

Export Destination	Description
File system	Do the following: **1.** Select the project resources that you want to export. **2.** In the To directory field, browse for the directory that you want to export the project into. **3.** Select any of the following options: • Overwrite existing files without warning. • Create directory structure for files. • Create only selected directories.
Archive file	Do the following: **1.** Select the project resources that you want to export. **2.** In the To archive field, browse for the directory that you want to export the project into. **3.** Select any of the following options: • Save in zip format. • Save in tar format. • Compress the contents of the file. • Create directory structure for files. • Create only selected directories.

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

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 SAP Mobile WorkSpace installations or when migrating to a more current version of SAP Mobile 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 the 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 connection profiles from the specified file.

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

Importing a Project

You can import or copy a project into your workspace from an archive file or from the file system.

Warning! If you import a project from a different workspace, but do not choose to copy it into the new workspace, 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. Select **File > Import** from the main menu bar.

   The Import wizard opens.
2. Select **Existing Project into Workspace** and click **Next**.
3. Select one of the following options, and follow the corresponding steps:

Table 22. Import a Project

Option	Description
Select root directory	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. 3. (Optional) Select **Copy projects into workspace**.
Select archive file	Browse for the archive that contains the project that you want to import.

4. Click **Finish**.

   The project is now available for use.

Importing the Public Certificate

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

Prerequisites

You must first configure SAP Mobile Server to accept HTTPS connections. You can then import the public certificate generated during that process and use it to secure HTTPS communications with SAP Mobile 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 SAP Mobile Server using HTTPS. The host is your SAP Mobile WorkSpace installation, or the host on which you develop .NET client applications; for example, `<SAP Mobile Platform_InstallDir>\JDK<version>\jre`). This task prepares your system to run J2SE SAP Mobile 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=SAP, L=Concord, ST=NH, C=US
   Issuer: CN=UEP, OU=ITS, O=SAP, 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
   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 generate 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 an SAP Mobile Server Connection Profile* on page 50
- *Preferences* on page 52
- *Importing and Exporting Connection Profiles and Projects* on page 61
- *Certificate Generation Command Line Utility Reference* on page 66

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 SAP Mobile 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**.

Option	Description
-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 SAP Mobile 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:

Field	Description
Cipher	For 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.

Field	Description
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 SAP Mobile Server in a Development Environment

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

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

In a shared development environment, SAP Mobile Server is typically installed by a SAP Mobile 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.

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, relationships, cache policies, and synchronization strategies), 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, or uses only DCN as the refresh mechanism).

Prerequisites

Before developing MBOs, understand the key concepts and principals described in *Fundamentals*. Also, see the companion guide, *Mobile Data Models: Using Mobile Business Objects*, for a deeper understanding of how to build an efficient MBO model.

Task

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, or the MBOs data is to be loaded only through Data Change Notification (DCN). 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 it's 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 an MBO using a DCN cache group policy without data source binding – when an MBO's CDB data is to be filled only through DCN, a data source binding is not necessary. In these cases, the MBO must reside in a cache group that uses the DCN policy.
- 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 117
- *Packaging and Deploying Mobile Business Objects* on page 211
- *Mobile Business Object Overview* on page 70

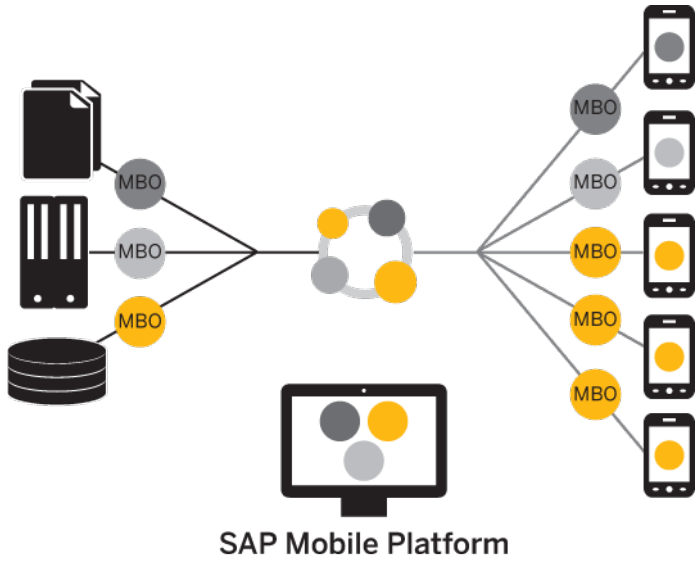
Mobile Business Object Overview

The cornerstone of the solution architecture is the mobile business object (MBO). For native Object API applications and Hybrid Apps, mobile business objects form the business logic by defining the data you want to use from your back-end system and exposing it through your mobile application or Hybrid App.

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 who are familiar with the data and transactional requirements of the mobile application, and how that application connects to existing datasources.

An MBO is derived from a datasource, such as a database server, Web service, or SAP® server. MBOs are deployed to SAP Mobile Server, and accessed from mobile device application client code generated from SAP Mobile WorkSpace or by using command line tools. MBOs:

- Are created using the SAP Mobile WorkSpace graphical tools, which 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, existing MBOs can be used.
- Provide a layer of abstraction from the SAP Mobile Server interaction with heterogenous back ends/devices, as shown in the following diagram.



MBOs can include:

- Implementation-level details – metadata columns that include information about the data from a datasource.
- Abstract-level details – attributes that correspond to instance-level properties of a programmable object in the mobile client, and map to datasource output columns. Parameters correspond to synchronization parameters on the mobile client, and map to datasource arguments.

  MBO operations include arguments that map to datasource input parameters. The argument's value that is passed to the enterprise information system (EIS) at runtime can come from an MBO attribute, personalization key, client parameter, or a default/constant value.
- Relationships – defined between MBOs by linking attributes and load arguments in one MBO to those in another MBO.

Developers define MBOs either by first designing attributes and load arguments, then binding them to a datasource; or by specifying a datasource, then automatically generating attributes and load arguments from it.

A mobile application package includes MBOs, roles, datasource connection mappings, cache policies, synchronization related information, and other artifacts that are delivered to the SAP Mobile 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 SAP Mobile Server. Other MBO artifacts are used to develop a mobile application using Native Object API or Hybrid Web Container API — when the application is deployed to a device, the MBO data model set resides on the device (in API code form). On-device data changes are synchronized to the MBO on the server, then to the

EIS backend. Backend changes are in turn 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 a Mobile Application Project* on page 72
- *Switching Between Developer Profiles* on page 75
- *Creating Mobile Business Objects* on page 75
- *Binding Mobile Business Objects to Data Sources* on page 82
- *Deploying Custom Classes to SAP Mobile Server* on page 117
- *Developing a Mobile Business Object* on page 69
- *Working with Mobile Business Objects* on page 117
- *Packaging and Deploying Mobile Business Objects* on page 211

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: • 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 topic *Mobile Business Object Validation Rules and Error Messages* in *Troubleshooting* for a complete list of guidelines.

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**.

   The Mobile Application Project is created and an empty Mobile Application Diagram opens.
4. (Optional) Modify the Mobile Application Project configuration properties by right-clicking the project in WorkSpace Navigator, selecting **Properties**, and selecting **Mobile Application Project Configuration**. When modifying the configuration properties, keep in mind that:
   - The default application ID and Display name are the same as the project. The description is "Default application ID".
   - Follow these guidelines when changing application ID, application name, display name, and description:
     - **ID –** less that 64 bytes, begin with an alphabetic character, followed by alphanumeric characters, a dot, or underscores, and not contain consecutive dots or underscores.
     - **Display name –** string, length less than 80 bytes.
     - **Description –** string, length less than 255 bytes.

     All added applications must have a name (display name and description can be empty), but are assigned a name at runtime when the application is created.

See also

- *Mobile Business Object Overview* on page 70
- *Switching Between Developer Profiles* on page 75
- *Creating Mobile Business Objects* on page 75
- *Binding Mobile Business Objects to Data Sources* on page 82
- *Deploying Custom Classes to SAP Mobile Server* on page 117

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**.

Copying a Project

Make a copy of an existing mobile application project from WorkSpace Navigator.

1. Right-click the mobile application project and select **Copy**.
2. Right-click and select **Paste**. Enter a new project name in the Copy Project dialog, then click **OK**.

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 23. 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.

Switching Between Developer Profiles

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

SAP Mobile 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 (default) – includes all SAP Mobile 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.

Switch to the basic profile if you need to use only basic SAP Mobile WorkSpace features.

If you want to use the basic profile by default, modify your developer profile preference settings. See *Mobile Development Developer Profile Preferences*.

- 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 > SAP AG > Mobile Development > Developer Profile**.

See also

- *Mobile Business Object Overview* on page 70
- *Creating a Mobile Application Project* on page 72
- *Creating Mobile Business Objects* on page 75
- *Binding Mobile Business Objects to Data Sources* on page 82
- *Deploying Custom Classes to SAP Mobile Server* on page 117

Creating Mobile Business Objects

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

To achieve optimal performance of insert, update, and delete operations, as well as client synchronization, the MBO definition must include a well defined primary key. A warning displays if the MBO does not include primary key attribute(s). Local MBOs do not require a primary key definition.

See also

- *Mobile Business Object Overview* on page 70

- *Creating a Mobile Application Project* on page 72
- *Switching Between Developer Profiles* on page 75
- *Binding Mobile Business Objects to Data Sources* on page 82
- *Deploying Custom Classes to SAP Mobile Server* on page 117

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.

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-SAP database data source, you must modify the SQL definition as indicated by the warning dialog to create non-SAP MBOs or operations using the Mobile Application Diagram (where you can manually enter 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.

   An MBO is created and data source information is automatically filled in with the appropriate operation and argument mappings, and/or attribute mappings.

   Note:
   - If you drag-and-drop a data source that includes a column with a computed default value, arguments 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 table with computed columns, or columns that contain computed default values, SAP Mobile WorkSpace does not generate operations automatically. You must define them with the Operation Creation Wizard.
   - Drag-and-drop may not generate correct MBOs for ASE/ASA views and non-SAP databases.

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 Attributes for a Mobile Business Object*

   Select the Mobile Business Object menu item from the Palette to invoke the New Mobile Business Object wizard to create a mobile business object (MBO) and its attributes.
2. *Creating Operations for a Mobile Business Object*

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

Select the Mobile Business Object menu item from the Palette 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 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Entity Read Operations* on page 140

- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Creating Multiple Mobile Business Objects From a Single Read Operation

Create multiple mobile business objects (MBOs) from a single read 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 output tables – EIS operations that contain multiple business application programming interface/remote function calls BAPI/RFC operations that contain multiple output tables. Each selected SAP output table is regarded as an independent result set, and mapped to corresponding MBOs. Those MBOs share a single BAPI/RFC operation.
- 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 First 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 24. Multiple Mobile Business Object Data Source

Data source	Description
SAP	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. 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. When you select <HEADER FIELDS> as the **Select an output table to map attributes option**, SAP Mobile 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: • 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	A method bundled with the multiple XSLTs. All XSLTs, except the one selected as the first MBO, can be used to generate subsequent MBOs.

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 SAP 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 arguments, and so on) of subsequent MBOs can be modified only through the first MBO.

Table 25. Multiple MBO Attribute Properties

Property tab	First/subsequent MBO
Share	Identifies the enterprise information system (EIS) operation to which the MBOs are bound. The share tab appears for: • Any MBO (except JDBC and REST) with multiple result sets. • First MBOs with at least one subsequent MBO. • Subsequent MBOs. The first MBO Share tab includes Add, Delete, and Delete All buttons, which you can use to add or delete subsequent MBOS. 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, SAP Mobile WorkSpace supports switching MBOs between normal and first MBOs.
Data source	Available only from the first MBO. • **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.

Property tab	First/subsequent MBO
Definition	The definition only can be changed in the first MBO, other than the ability to modify subsequent result set filters. After changing the definition, SAP Mobile WorkSpace attempts to match and merge the changes, which may include adding, refreshing, or deleting subsequent MBOs. First and subsequent MBOs share the same credential properties.
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 arguments	Available from the first MBO and is read-only for subsequent MBOs. For each MBO, the load argument 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

Local business objects can be deployed to SAP Mobile Server and include Object API code generated that performs business functions local to the device application. For example, the local business object can call object queries or persist application specific configuration information locally across application restarts

A local business object is not bound to any data source, has no server runtime artifacts like result set filters or result checkers, and has no effect on the SAP Mobile Server cache database (CDB) and operation replay. For example, a local business object cannot be filtered or synchronized with the CDB, and does not belong to any synchronization or cache groups.

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 Update and/or Delete operations if needed by the local business object (The Create operation is always selected).
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 running and accessible. 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 SAP Mobile 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 operations.

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

Option	Description
New Mobile Business Object or New Operation wizard	Bind to a data source when adding operations and attributes to a new mobile business object.

Option	Description
Properties view	For an existing mobile business object that you have created in a top-down fashion (in which you create attributes and/or operations 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 Object Overview* on page 70
- *Creating a Mobile Application Project* on page 72
- *Switching Between Developer Profiles* on page 75
- *Creating Mobile Business Objects* on page 75
- *Deploying Custom Classes to SAP Mobile Server* on page 117
- *Creating Operations for a Mobile Business Object* on page 134

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 26. 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, a SOAP service or a resource URL

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

Use client credentials (including certificates and SSO tokens on EIS types that support them) 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 user name and password properties to system-defined "user name" 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, RESTful Web Service, or SOAP MBO), enter the client credentials in the User name and Password fields. The runtime data source credential values (user name and password) that SAP Mobile 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) System personalization keys 'user name' and 'password'.
   d) 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 user name and password portions for the selected server connection is replaced by the user name 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 SAP Control Center.
- When you create a new security configuration that includes the SAPSSOTokenLoginModule, and deploy it to a new domain, if the Hybrid App uses the MBOs associated with the new security configuration, you must specify an SAP Mobile Server domain that corresponds to the domain using the security configuration. See *Security* for more information about security configurations

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 and configure these properties that are described in the related tasks:

- 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 40
- *Binding an SAP Data Source to a Mobile Business Object* on page 90
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 102

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 JDBC 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 configure and bind data source information to mobile business object attributes and operations. Some fields are specific to attributes, while others are common to both attributes and operations. For example, Result set filters are attribute specific.

Field	Action
Data source specific information	In the wizard, enter the SQL statement used to access the database information. Click: • SQL Statement Type – either a SQL query statement or a stored procedure. • Validate syntax – to validate the syntax of the SQL statement. • Preview – to view the results of the SQL statement against the data source.

Field	Action
Runtime Data Source Credential	(Optional) The **User Name** and **Password** required to gain access to the runtime data source. 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.
Result set Filters	Optionally add a result set filter to the MBO. 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 `javax.sql.RowSet`, which is used to implement the filter interface. See *Result Set Filters*.
Parameters	From the Parameters page you can configure these MBO argument properties that are modeled from the data source's remote operation argument: • Argument – the MBO argument name • Datatype – datatype of the argument • Nullable – identifies if NULL is a valid value • Default value – the default value, if any, of the MBO argument Due to JDBC driver meta data retrieval related API limitations, SAP Mobile WorkSpace does not always set the correct attribute length, especially for operation and load argument types and lengths, and you must set them manually to avoid runtime errors.

Field	Action
Attributes Mapping	Attributes mapping is generated as follows: **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. (Properties view only) The methods that you use to map attributes include: **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. From the Attributes Mapping screen, you can modify the mappings by selecting: • Up or Down – adjust attribute position.

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.

AutoCommit Option for JDBC MBOs Using an ASE Data Source

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 SAP Mobile 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 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 27. Stored Procedures With Output Parameters Definition Examples

Stored Procedure's Output Parameter	SQL Query/Definition

Scalar output parameter (ASA)	`{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`

Cursor as output parameter (Oracle)	`{CALL TESTCURSORSPOP(["pcursor"=":pcursor"])}`
	
	`create or replace procedure testCursorSPOP( p_cursor out types.cursorType ) as`
	`begin`
	`  open p_cursor for select * from tblSPOP;`
	
	`end;`

Stored Procedure's Output Parameter	SQL Query/Definition

Cursor as return result set (DB2)	`{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`

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.

Example: Parameters and Stored Procedures

Learn how to create a mobile business object (MBO) from a stored procedure that contains parameters, and how those parameters are used as MBO load arguments.

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 arguments.

Binding MBOs to ASE Data Sources That Contain Timestamps

Understand how SAP Mobile WorkSpace treats Timestamp datatypes when binding to Adaptive Server Enterprise data sources.

SAP Mobile WorkSpace filters out timestamp datatype arguments from Create, Update, and Delete operations when the MBO Developer drags an ASE database table and drops it onto the Mobile Application Diagram. That is, SAP Mobile WorkSpace does not generate the related operation (create/update/delete) argument for timestamp datatypes of ASE database table. This filtering does not apply to Read operations.

It is considered a user error to create an operation from the Mobile Application Diagram pallette using the timestamp column in the definition.

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.

Field	Action
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	**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. 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 **Refresh** in the Search dialog to refresh the SAP data source and display current BAPIs/RFCs. This is only necessary if there is a change to the data source and need to display the current BAPIs/RFCs. Select the operation and click **OK**.
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 provide access. You can also set default values, or use personalization keys for all JCo connection properties.
Result checker	Optionally add a result checker to the MBO. A result checker is a custom Java class that implements error checking for mobile business objects (MBOs). See *Adding a Result Checker*.

Field	Action
Result set filters	Optionally add a result set filter to the MBO. 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 `javax.sql.RowSet`, which is used to implement the filter interface. See *Result Set Filters*.
Preview	Click **Preview** to preview the newly defined mobile business object attributes.
Refresh	(Optional) Select **Refresh** to refresh the parameters for the selected BAPI operation. Any changed parameters are decorated with an asterisk "*" at the beginning of the name. This is required only if there is a change to the BAPI.

3. Click **Finish** to use default mappings, or **Next** to modify the columns to attributes mapping, or parameter mapping.
4. From the Parameters page you can configure these MBO argument properties that are modeled from the data source's remote operation argument:

   - Argument – the MBO argument name
   - Datatype – datatype of the argument

     The datatype of a data source column is read-only, and changes only if the **Map to** column changes.
   - Nullable – identifies if NULL is a valid value
   - Default value – the default value, if any, of the MBO argument

   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 40
- *Configuring an SAP Exposed Web Service MBO to Use Credentials* on page 102
- *Implementing SSO for SAP* on page 84

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 SAP Mobile WorkSpace

SAP Mobile 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: • 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.
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.

Field	Action (Web service data source)

Configure XSLT	To map Web service methods to mobile business object attributes, the Web service response message is converted to a table format. Select the **Edit** or **Add** button to invoke the **Edit XSLT** dialog, where you can define or adjust the XSLT that is used to flatten the response messages. Options include:
	• **Generate XSLT from response message elements selection** – SAP Mobile WorkSpace automatically creates the XSLT and selects all elements by default. Select/unselect the desired elements to be mapped to the MBO attributes. 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** – a default XSLT is generated by default. Manually edit the elements and attributes:
	  • **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 an existing 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.
	Note:
	• A Web service response containing an `<s:any>` node may result in two columns being generated. Manually remove the least relevant entry in these cases.
	• When SAP Mobile WorkSpace generates a default XSLT for transformation of the output of a Web service operation, each result field in the XSLT includes an `op_bindpath` setting. If you modify the XSLT, leave the `op_bindpath` 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.

Field	Action (Web 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	Optionally add a result checker to the MBO. A result checker is a custom Java class that implements error checking for mobile business objects (MBOs). See *Adding a Result Checker*.
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/Test Execute	Click **Preview/Test Execute** to preview the results of the WSDL method invocation. You may need to enter valid values for the WSDL method parameters for the preview to succeed. Optionally, you can save multiple sets of arguemnt values into Preview/test execute configurations for reuse later. Once the preview results are verified with the provided input argument values, you can select **Save as default values** to save those values as the default values for the operation arguments/load arguments.

Field	Action (SOAP Service data source)
Input SOAP message	The input SOAP message from which the mobile business object attributes are derived.
Destination URL	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). 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**.

Field	Action (SOAP Service data source)

Configure XSLT	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:
	• **Generate XSLT from response message elements selection** – SAP Mobile WorkSpace automatically creates the XSLT and selects all elements by default. Select/unselect the desired elements to be mapped to the MBO attributes. 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** – a default XSLT is generated by default. Manually edit the elements and attributes:
	  • **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 an existing 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.
	Note: A Web service response containing an `<s:any>` node may result in two columns being generated. Manually remove the least relevant entry in these cases.

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. To use the device user runtime credentials, you need to select system defined personalization keys ('username' and 'password'), in the corresponding fields.
Result Checker	Use the predefined **Default** or **None** result checker, or define and use a **Custom** result checker. See *Adding a Result Checker*.

Field	Action (SOAP Service data source)
Result Set Filters	Add a custom Java class to manipulate (filter) rows or columns of data returned for an MBO. See *Result Set Filters*.
Preview/Test Execute	Click **Preview/Test Execute** to preview the results of the SOAP method invocation. You may need to enter valid values for the SOAP method parameters for the preview to succeed. Optionally, you can save multiple sets of arguemnt values into Preview/test execute configurations for reuse later. Once the preview results are verified with the provided input argument values, you can select **Save as default values** to save those values as the default values for the operation arguments/load arguments.

3. Click **Finish** to use default mappings, or **Next** to modify the columns to attributes mapping, or parameter mapping.
4. From the Parameters page you can configure these MBO argument properties that are modeled from the data source's remote operation argument:

   - Argument – the MBO argument name
   - Datatype – datatype of the argument
   - Nullable – identifies if NULL is a valid value
   - Default value – the default value, if any, of the MBO argument

   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. After the Definition page of the New Operation wizard, you can modify operation and client parameters from the Parameters page by selecting the corresponding tab:
   - Operation Parameters – lists all operation arguments, associated datatypes, nullability, and the arguments value can be filled from attributes, client parameters, or default values.
   - Client Parameters – add client parameters used by device application users to provide values to the operation arguments during runtime.
7. The **Role Assignments** screen allows you to **Create**, **Add**, and **Remove** role assignments from the mobile business object. Click **Finish** when done.

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 create (insert) operations: the OrderItem.create 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 (create) operation:

- Ensure that Order.create operation contains a response XSLT that has the newly created Order.Id as one of the elements.
- Chain the two create 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 argument of OrderItem.create.

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 create operation.

   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 create 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 argument name has a corresponding **Fill from attribute** value defined.

      All arguments 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 argument name of the create operation 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.
- 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

- For WSDL and SOAP Web service data sources , if an Update operation that uses the **Immediately update the cache** operation cache policy does not have an XSLT, an error message displays:

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

- If namespaces are used, the XSL namespace must be defined in the WSDL to parse the namespaced SOAP response message nodes.

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 SAP Mobile 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 application to use either X.509 or SSO2 credentials.

See also

- *Creating an SAP Connection Profile* on page 40
- *Binding an SAP Data Source to a Mobile Business Object* on page 90
- *Implementing SSO for SAP* on page 84

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 SAP Mobile Server truststore into an SAP Mobile WorkSpace truststore.

Perform this configuration on the SAP Mobile 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 `<SAP Mobile Platform_InstallDir>\MobileSDK<version>\Eclipse` directory.

2. Use the Java **keytool** command to add the same SAP server certificate you imported into the SAP Mobile Server truststore into `truststore.jks`.
3. Add this argument to the `<SAP Mobile Platform_InstallDir>\MobileSDK<version>\Eclipse\MobileWorkSpace.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 ..\JDK<version>\jre\lib\security\cacerts truststore.jks
@echo *
@echo * Answer Yes to "Trust this certificate?".
@echo *
keytool -import -keystore truststore.jks -file sap-doe-
vm1.sybase.com.PEM.crt -storepass changeit
@echo *
@echo * Add to eclipse vmargs: -Djavax.net.ssl.trustStore=
%ECLIPSE_ROOT%\truststore.jks
@echo *
```

5. Restart SAP Mobile 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 operations, the response representation (XSLT) is mandatory for MBO attributes. For Create, Update, Delete and Other operations, the request and response representation is optional. You can also define multiple-XSLTs for a shared read MBO (multiple MBOs load from a single EIS 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

Field	Action
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	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. 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: • boolean • string • binary • char • byte • short • int • long • integer • decimal • float • double • date • time • dateTime 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.
HTTP method	The method called, either GET, PUT, POST, or DELETE.
Expected status code	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. 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. If the returned status code does not match the specified expected status code, the EIS operation fails.

Tab	Description
Representation	Allows you to specify the request or response representation. Select the representation type: • Request • Response Enter or select: • **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. Click **Edit** to specify the XML schema for a request or response representation, instead of referencing an existing representation: • **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. Only one XSD can be defined for a given representation. **Define XSLT manually** – for response representation, after the XSD definition and root element was specified, SAP Mobile 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 shared read MBOs is also supported. All available XSLT definitions are listed in the table in the XSD definition dialog: • **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 an existing 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. Select **OK** to save your definition and exit the dialog.

Tab	Description
Authentication	Supports HTTP Basic Authentication: • **User name –** authenticated user • **Password –** password of the authenticated user. 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. To use the device user runtime credentials, you need to select system defined personalization keys ('username' and 'password'), in the corresponding fields from the drop-down list.

HTTP header	Captures all properties needed to execute or manage the REST Web Service.
	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:
	• 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, SAP Mobile 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.
	The Input/Output/Both setting determines if Variable and Value are enabled:
	• Input – Variable is enabled:
	  • 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. From the Parameters page you can configure these MBO argument properties that are modeled from the data source's remote operation argument:

   - Argument – the MBO argument name
   - Datatype – datatype of the argument
   - Nullable – identifies if NULL is a valid value
   - Default value – the default value, if any, of the MBO argument

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. After the Definition page of the New Operation wizard, you can modify operation and client parameters from the Parameters page by selecting the corresponding tab:
   - Operation Parameters – lists all operation arguments, associated datatypes, nullability, and the arguments value can be filled from attributes, client parameters, or default values.
   - Client Parameters – add client parameters used by device application users to provide values to the operation arguments during runtime.
7. The **Role Assignments** screen allows you to **Create**, **Add**, and **Remove** role assignments from the mobile business object. Click **Finish** when done.

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 principles:

- Provides an ID (structure-like URIs) for every resource
- Links resources together
- Uses standard HTTP methods
- Supports resources with XML representation (Transfer XML)
- 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 SAP Mobile 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 28. 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`

REST Web Service Mobile Business Object Limitations

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

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

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

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

```
The response representation of the ''UPDATE'' operation
 'Customer- > updateStudent()' with ''Immediately update
the cache'' cache 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 the connection profile associated with an MBO's attributes or operations to another connection profile of the same type. This allows the MBO developer to switch to a different data source, assuming the MBO attributes and operations metadata are identical. For example, MBO developers use a development database data source to develop the MBO, then another team chooses a different database server with the same metadata/configuration for testing.

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

If you change the connection profile for an MBO definition that also has operations that use the same connection profile, SAP Mobile WorkSpace prompts you to apply the connection

profile change to the operations as well as the definition. SAP recommends that you apply the connection profile change to the MBO operations, if not, the operations are applied to a different data source than the one from which the MBO reads.

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 *Mobile Data Models: Using Mobile Business Objects* 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 SAP Mobile 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	The result checker depends on the data source type: • SAP – `com.sybase.sup.sap3.SAPOperationHandler`. If a RETURN parameter is found in the selected operation, this option is automatically selected. • Web service (SOAP) – `com.sybase.sup.ws.soap.SoapOperationHandler`. The default checker always returns the status as successful. • Web service (RESTful) – `com.sybase.sup.ws.rest.RestOperationHandler`. The default checker always returns the status as successful.

Option	Description

None	Return the status as successful with no message. The result checker used depends on the data source type:
	• SAP:
	`package com.sybase.sap3;`
	`import com.sybase.sup.sap3.SAPOperationHandler;`
	`/*`
	` * Abstract SAPOperationHandler already defines the no-op methods`
	` */`
	`public class NoopSAPOperationHandler extends SAPOperationHandler {`
	`}`
	• Web service (SOAP):
	`package com.sybase.sup.ws;`
	`import com.sybase.sup.ws.soap.SoapOperationHandler;`
	`public class NoOpSoapOperationHandler extends SoapOperationHandler {`
	`}`
	For SOAP, use the **Default** result checker instead of **None** to parse the SOAP fault and return it to the client as a log record.
	• Web service (RESTful):
	`package com.sybase.sup.ws;`
	`import com.sybase.sup.ws.rest.RestOperationHandler;`
	`public class NoOpRestOperationHandler extends RestOperationHandler {`
	`}`

Custom	Specify a custom result checker. See the topic *Writing a Custom Result Checker* in *Mobile Data Models: Using Mobile Business Objects* for information about writing a custom result checker.

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 `Filters` folder from your project. Click **Browse** to locate the source folder for the Java class.
Package	Click **Browse** to locate the package for the new Java class. **Note:** SAP recommends that you do not leave this field blank. Otherwise, the JDK<version> Java class in the default package cannot be resolved in other packages.
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	1. Click **Browse**. 2. In the Superclass Selection dialog, enter: • Choose a Type • Matching Items 3. Click **OK**.
Interfaces	By default, this is populated with the corresponding interface for the selected data source type. Click **Add** to select interfaces implemented by the new class.
Which Method Stubs Would You Like to Create	• **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.

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**.

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

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**.

Adding a Result Set Filter

Add a result set filter to manipulate the rows or columns of data returned to SAP Mobile Server from the enterprise information system (EIS) by an MBO read operation.

See *Mobile Data Models: Using Mobile Business Objects* for information about writing a custom result filter.

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 SAP Mobile 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 SAP Mobile WorkSpace.

3. Implement the new class by writing the real implementation on top of the skeleton created as documented in *Writing a Custom Result Set Filter* in *Mobile Data Models: Using Mobile Business Objects*.
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 SAP Mobile 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.

Viewing the Filter Class Output Stream

You can set debugging options to view the output stream when using System.out.printIn in filter classes to help you debug your filter classes.

1. Go to `<SMP Installation Root>\MobileSDK<version>\Eclipse` and open the `MobileWorkSpace.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 SAP Mobile Server side, all debug statements are saved to the `ml.log` file.

Deploying Custom Classes to SAP Mobile Server

Deploy result checker and result set filter classes to SAP Mobile Server when you deploy mobile business objects (MBOs) that contain them.

If any of your MBOs includes a result checker, result set filter (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 SAP Mobile 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 SAP Mobile Server. Follow the prompts to include the custom classes.

See also

- *Mobile Business Object Overview* on page 70
- *Creating a Mobile Application Project* on page 72
- *Switching Between Developer Profiles* on page 75
- *Creating Mobile Business Objects* on page 75
- *Binding Mobile Business Objects to Data Sources* on page 82

Working with Mobile Business Objects

Create attributes and operations for mobile business objects, create relationships between mobile business objects, bind them to 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 211
- *Mobile Business Object Overview* 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 29. 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.

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 > SAP Mobile WorkSpace**, or select the **Ctrl +H** keys.
2. In the Search dialog, define your search criteria and click **Search**.

Table 30. 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. SAP Mobile WorkSpace supports approximate string matching (fuzzy searches). For example, if you enter "test" as your MBO search string, the results display all MBOs that contain the string "test."
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?".
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.

Option	Description
Scope	Limit the scope by mobile application project: • 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 31. Mobile Application Diagram Properties

Tab	Contents
Appearance	From the Properties view, define various aspects of the Mobile Application Diagram's appearance, including: • 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.

Mobile Application Diagram Related Features

You can generate and save an image of an MBO from the Mobile Application Diagram.

Generating a file image of an MBO

1. Right-click the MBO and select **File > Save as Image File** .
2. Select the name and other properties of the image and click **OK**.

Note: PDF is not a supported image format.

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

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

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

- Creating MBOs – when creating MBO operations and attributes in these locations:
  - Attributes Mapping wizard
  - Client Parameters page (when deferring binding to a datasource)
  - Attributes page (when deferring binding to a datasource)
- Editing MBOs – when editing MBO attributes and parameters from the Properties view in these locations:
  - Load Arguments tab
  - Attributes Mapping 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 datasource 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).

SAP Mobile WorkSpace supports various categories of datatypes.

Table 32. Datatype Categories

Category	Description
Simple	int, string, date, bigString/bigBinary, and so on.

Category	Description
List of simple types	An array of simple types: int[], string[], date[], and so on.
Structure (complex)	Implemented with a structure, and includes: • SAP input structure or input table • Nested complex types in Web Services that handle type structures as input (repeating and nested elements) **Note:** Parameters, personalization keys, and default values all support complex types. Attributes do not, except for those that represent relationships.
List of structure types	An array of objects: customer[], account[], and so on.

The following table describes how to input values for various simple datatypes when specifying default values of parameters of these types in SAP Mobile WorkSpace.

Table 33. Simple Datatype Description

Datatype	Description
binary (%n)	Select either: • 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. 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: **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. **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.
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(%n)	To set the string 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: **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. **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.

Datatype	Description
BigString/BigBinary	Allows you to transfer large binary or string data from/to SAP Mobile Server by using the ObjectAPI streaming methods to optimize memory consumption. For example, you can set an MBO's attribute/arguments as BigString/BigBinary and get/set the data by a streaming method such as seek/write/read/flash in the client code. To use either BigString or BigBinary types and the associated streaming objectAPI, an MBO operation argument must be mapped to a Fill from attribute that uses the large object type. An error is generated if an MBO operation argument has a BigString or BigBinary type but does not have the associated Fill from attribute with the same large object type. Not all attribute/arguments support BigString/BigBinary: **1.** Only operation argument/structure object attributes can be set as BigString/BigBinary, which have no length limitation. Use a streaming I/O mechanism to optimize memory consumption at runtime. **2.** Personalization key/sync parameter/load argument/object query parameter/client parameter are not allowed to set BigString/BigBinary datatype. **3.** In the attribute mapping section of properties view, primary key attributes are not allowed to have BigString/BigBinary data type. If you set a primary key attribute as BigString/BigBinary, SAP Mobile WorkSpace displays a validation error. The BigString/BigBinary options are still available since you can unselect the primary key checkbox. **4.** To indicate an operation argument needs to use BigString/BigBinary, change the mapped attribute's datatype. **Note:** Ensure that your SAP Mobile Server host is a 64-bit machine with a heap size larger than 512 MB to avoid "java.lang.OutOfMemoryError: Java heap space" errors when processing BigString or BigBinary datatypes.
All others	Enter the appropriate value in the **Value** field.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147

- *Composite Operations* on page 148

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.

Mobile Business Object Properties

Modify mobile business object properties from the Properties view.

Table 34. Mobile Business Object Properties

Tab	Contents
General	• 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	If the mobile business object attributes are bound to a data source, the contents of the Attributes tabs include: • 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. Result Set Filters shows resultset filters and their paths. • Load Arguments – see *Adding a Load Argument* for details. • Attributes Mapping – Includes: • **Attributes** • Name – name of the attribute. • Datatype – attribute datatype • Nullable – a Null value for the attribute is valid. • Primary key – analogous to a relational database table's primary key, when set, MBO data can be searched/found using the attribute's value. When set, a findByPrimaryKey object query is generated by default, and the generated device client code contains FindByPrimaryKey methods (along with associated parameters and return types). At runtime, FindByPrimaryKey methods return a collection of objects that match the specified search criteria defined in the object query. **Note:** The SAP Mobile 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. • **Data Source** • 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. **Note:** For SAP and Web service data sources, the Nullable and Datatype columns are read-only. You cannot change them, in-

Tab	Contents
	stead, SAP Mobile WorkSpace correctly identifies and sets them from the back-end data source. But for a JDBC data source, SAP Mobile WorkSpace may not be able to correctly identify the Datatype and nullability, so you need data source knowledge to correct them if needed. • 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 unmapped 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 attribute to tabular view column mapping displays. • 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 *Object Queries* for details. • Share – lists MBOs and data source bindings for any attributes that share an EIS read operation. Using **Delete** and **Delete All** deletes the shared-read MBO.
Operations	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. Select **Bind** to bind an operation to a data source, or **Rebind** to change an existing binding to a different data source. See *Mobile Business Object Operation Properties* for details.
Relationships	Lists all relationships defined between this and other mobile business objects. Select **Add** to define a new relationship, highlight an existing relationship and select **Edit** to modify an existing relationship, or use **Delete** and **Delete All** to remove relationships.
Synchronization	Define various synchronization aspects of the MBO from the Advanced Developer profile. See *Defining Synchronization Properties for Individual Mobile Business Objects*.
Appearance	Modify certain look-and-feel aspects of the mobile business object from the Advanced Developer profile.

See also

- *Creating Attributes for a Mobile Business Object* on page 133

- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Creating Attributes for a Mobile Business Object

Select the Mobile Business Object menu item from the Palette to invoke the New Mobile Business Object wizard to create a mobile business object (MBO) and its attributes.

1. Name the MBO and optionally provide comments in the Mobile Business Object page, and click **Next**.
2. Select the **Bind data source later** radio button in the Data Source page, and click **Next**.
3. Select **Add** to add attributes and adjust datatypes and nullability in the Attributes page.

   Note: An MBO may not always contain attributes. For example, the developer can create an empty MBO with Other type operations.
4. (Optional) Select the logical roles assigned to the MBO in the Role Assignment page.
5. Click **Finish**.

See also

- *Creating Operations for a Mobile Business Object* on page 134
- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Mobile Business Object Attribute Properties

Modify a mobile business object (MBO) that contains a single attribute using the Properties view by clicking a specific MBO attribute in the Mobile Application Diagram.

Table 35. Single Attribute Properties

Tab	Contents
General	• 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 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148
- *Load Arguments* on page 187
- *Combining Load Arguments and Synchronization Parameters* on page 190

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, Entity Read used in conjunction with Create and Update to refresh a single instance of an MBO, 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. 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* on page 133
- *Binding Mobile Business Objects to Data Sources* on page 82
- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Mobile Business Object Operation Properties

Modify mobile business object operation properties during operation creation, or later, using the Properties view.

Table 36. Operation Properties

Tab	Contents

General	• Operation name – the name of the operation.
	• Operation type – the operation to be performed. Operation types include Create, Update, Delete, Entity Read and Other.
	  • When defining an operation of type 'Other', attributes are not available for input mapping because attributes are not associated with these operations.
	  • See *Entity Read Operations* for information about how and when Entity Read operations are used.
	• Comment – describes the operation.

Data Source	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.
Definition	Operation definition varies, depending on the data source type to which the operation is bound. See *Binding Mobile Business Objects to Data Sources* for data source-specific information. View, modify, and test operations: • 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 – test 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. Be advised that using Test execute or Preview for large data or object types (BigString/BigBinary), can result in out of memory errors in SAP Mobile WorkSpace. **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.

Tab	Contents

Input	View or configure the values that map into remote operation arguments. Select **Refresh** to update the displayed arguments.
	The Input tab includes these subtabs:
	• **Input Mapping –** Configure input mappings for remote operation arguments. Create, Update, and Delete operations can define mappings for all nodes of a composite object graph.
	Mappings and values can be:
	  • arguments or sub-arguments bound to attributes or set of attributes – fills the argument's value with that of the selected attribute or set of attributes.
	  • Personalization key – fills the argument's value with that of the mapped personalization key's value.
	  • Client Parameter – fills the argument's value with that of the mapped client parameter's value. Beginning with SAP Mobile WorkSpace version 2.2 SP02, when creating or rebinding an operation, client parameters are no longer automatically created and mapped - except for operations of type 'Other'.
	See *Defining Input Mappings* and *Composite Operations* for more information.
	• **Default Values –** the default value of the argument. See *Input Mapping Default Value Precedence* for details.
	• **Client Parameters –** available from the device application and allows the user to enter values that are passed to the enterprise information system (EIS) operation argument during operation replay. Client parameters can be mapped into operation arguments to determine how the client passes information to the EIS:
	  • Name – name of the client parameter. Names cannot contain C# or Java reserved words.
	  • Datatype – the argument's datatype. BigString/BigBinary datatypes are not supported, and an error displays if the MBO developer selects a structure type that contains BigString or BigBinary fields.
	To provide client parameters for BigString/BigBinary datatypes, or a structure with BigString/BigBinary fields, use compatible types: String for BigString, Binary for BigBinary. For a structure, copy the original structure in WorkSpace Navigator, and change the large object type fields to String/Binary, then use the new structure as the datatype of the client parameter.
	  • Nullable – accepts null as a valid value.

Tab	Contents
	Note: See *Defining Input Mappings* for additional information, including a shortcut for adding and mapping a client parameter using drag and drop.
Output	(Create, Update, and Entity Read operations) Configure the remote operation results-to-MBO mapping by selecting the remote operation result and connecting it to the MBO attribute to which it maps. Alternatively, map the entire output to MBO by connecting the high-level result to the MBO. See *Defining Output Mappings* for additional guidelines. Most guidelines for multiple mappings (composite relationship) apply to a single result.
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. Entity Read operations do not support roles.
Cache Policy	Determine how the results of an MBO operation are applied to the SAP Mobile Server cache. See *Operation Cache Policies*.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Old Value Argument

Use the Old Value Argument field to map a remote operation argument to a second (old) argument in update or delete operations. Only MBOs bound to JDBC data sources support the Old Value Argument.

When updating columns with operation argument values, or deleting rows based on operation argument values, use the **Old value argument** field to access column values known to the device before changes. For example, if you use an update operation to modify the value of the lname column 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 Unmap option unmaps the old value.

The old value argument does not support bigstring or bigbinary datatypes.

The **Old value argument** field is available from a number of Properties view locations, including:

- Default Values horizontal tab of the Input vertical tab.
- Operation Editing dialog.
- Test execute dialog supports selection of old.*argument* parameters when testing update and delete operations.

Avoiding synchronization conflicts with the old value argument

If a mobile business object (MBO) performs an update or delete 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 value has become stale, the EIS operation can avoid conflicting updates, also known as Optimistic Concurrency Control (OCC).

Old value argument with an Update operation

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.

Old value argument with a Delete operation

This example illustrates the steps for support of the delete operation old value argument:

1. Create an MBO from a JDBC data source using this SQL statement:
   ```
   SELECT * FROM contact
   ```
2. Create a Delete operation defined as:
   ```
   DELETE FROM sampledb.dba.contact
   WHERE (id = :id)
    AND (:first_name IS NOT NULL)
    AND (first_name = :old.first_name)
   ```
3. Deploy the MBO and create a client application that loads all data from the enterprise information system (EIS) to the client.
4. For a given row, for example, `id=1`, client one modifies `first_name (Jane)` to Mary.

5. Client two fails when attempting to delete the changed row whose `id=1`, since the `first_name` has changed.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Entity Read Operations

Entity Read MBO operations refresh a single instance of an MBO composite graph in the SAP Mobile Server cache for Create and Update operations, and are typically used when the Create or Update operation cannot return the entire graph.

The input of the Entity Read operation is a primary key. The Entity Read operation is not called directly by a client; MBO Create and Update operations invoke it if the "Immediately update the cache" operation cache policy and the "Apply output of ENTITY READ operation" option are selected, to immediately apply the results of the Entity Read operation to the CDB.

For example, a mobile application client executes a Create operation. After the EIS to which the MBO is bound is updated, an immediate refresh of the cache for a composite object graph based on the definition of the Entity Read operation is triggered, if "Immediately update the cache" and "Apply output of ENTITY READ operation" were selected by the MBO developer for the Create operation.

The Entity Read operation uses operation arguments filled from MBO primary-key attributes to create a findByPrimaryKey Read operation.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136

- *Datatype Support* on page 125
- *Old Value Argument* on page 138

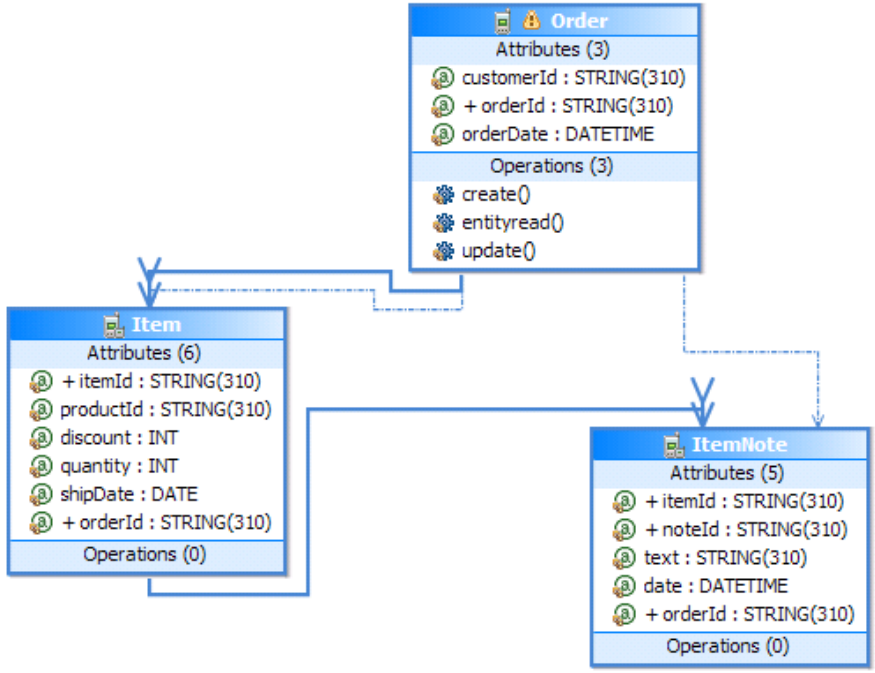
Entity Read Example

A composite relationship of REST Web service MBOs with an Entity Read operation that returns multiple XSLTs.

The *Mobile Business Object: restdemo Example Project* illustrates SAP Mobile Platform features including entity read operations, output mappings, and composite relationships, and can be downloaded and installed from the SAP® Community Network: *http://scn.sap.com/docs/DOC-8803*.

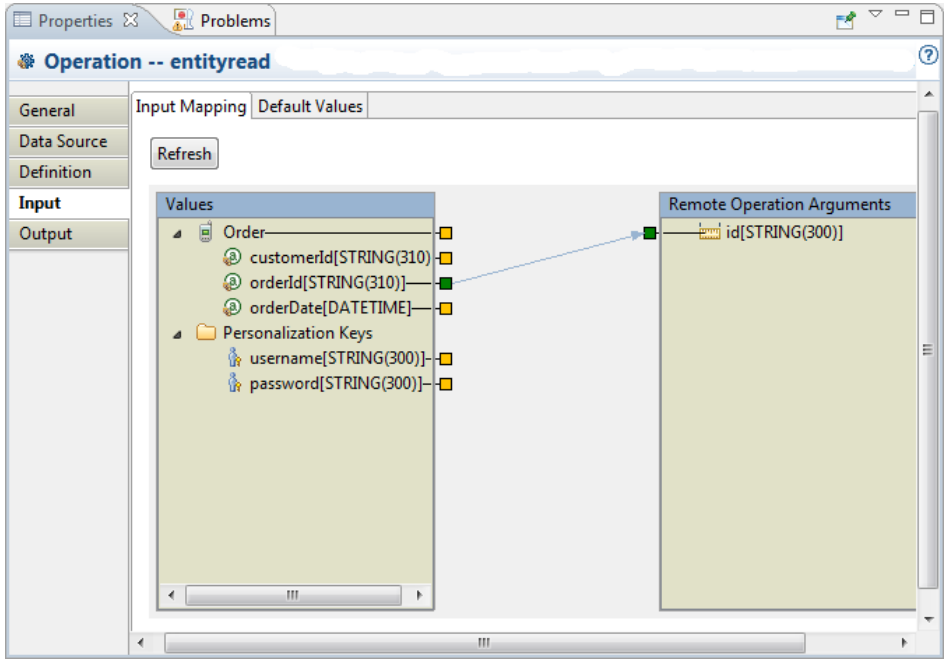
This example shows a composite relationship in which:

- The Order and Item MBOs have a one-to-many composite relationship through their orderId attributes.
- The Item and ItemNote MBOs have a one-to-many composite relationship through their orderId and itemId attributes.

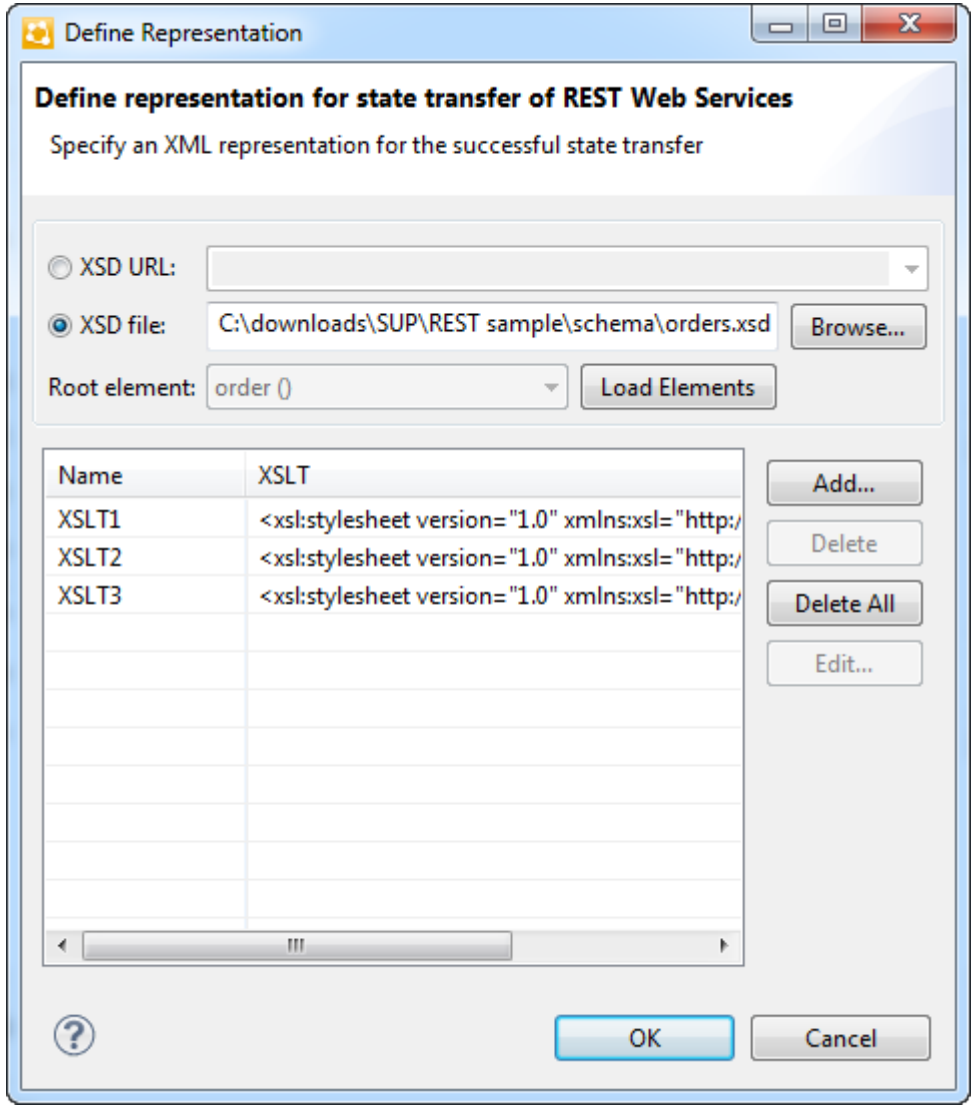


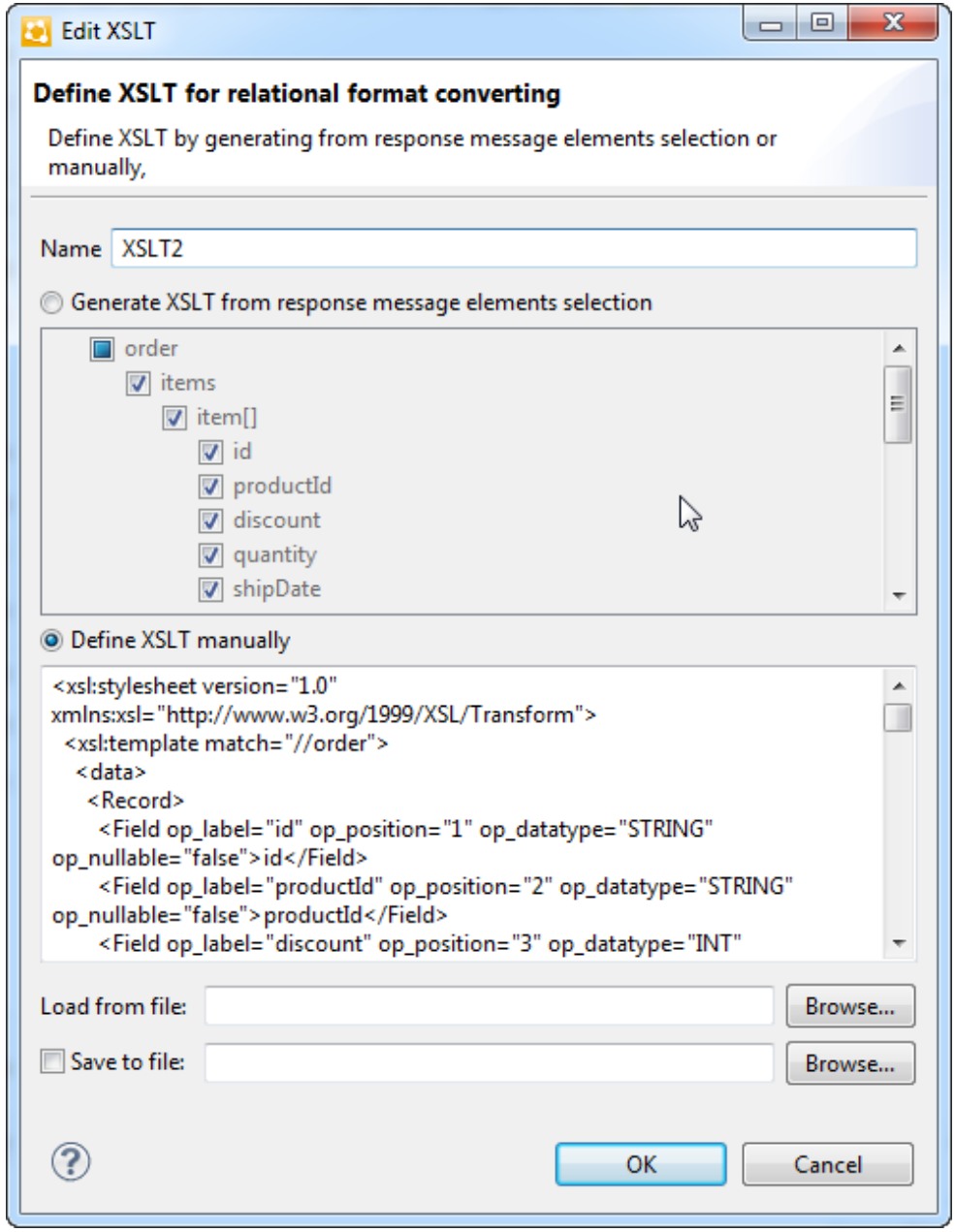
The Entity Read operation for the Order MBO includes:

- A parameter with a single argument (id) that is filled from the orderId attribute (Primary key).

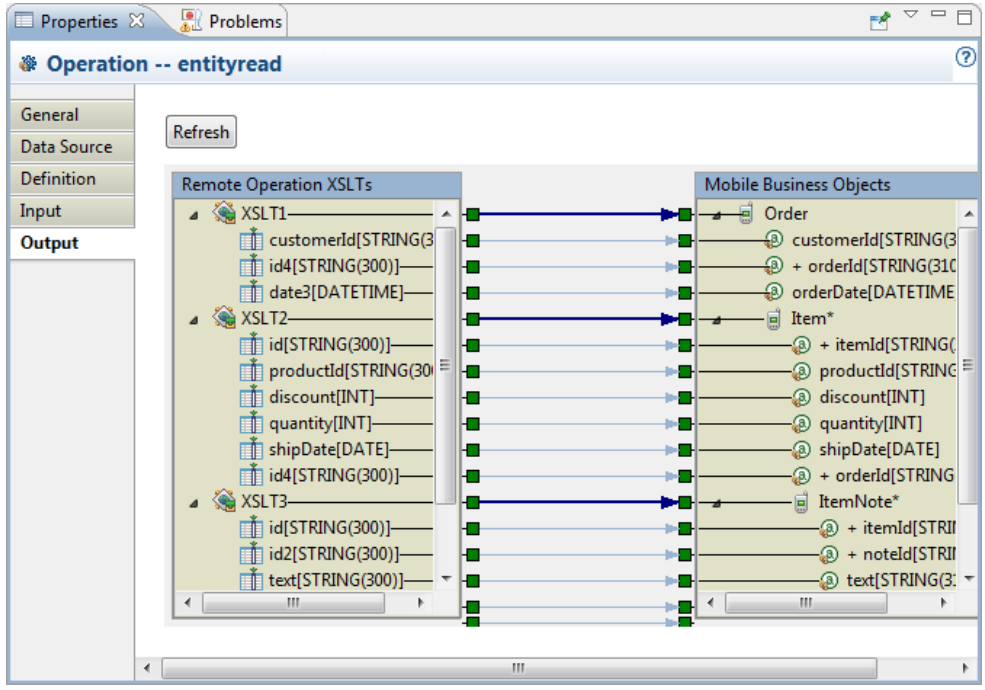


- A definition that includes three response representations; each of which maps to an MBO in the composite graph. For example, XSLT2 maps to the Item MBO.


Define Representation
Define representation for state transfer of REST Web Services
Specify an XML representation for the successful state transfer
XSD URL:
XSD file:
C:\downloads\SUP\REST sample\schema\orders.xsd
Browse...
Root element:
order ()
Load Elements
Name
XSLT
XSLT1
<xsl:stylesheet version="1.0" xmlns:xsl="http:/
XSLT2
<xsl:stylesheet version="1.0" xmlns:xsl="http:/
XSLT3
<xsl:stylesheet version="1.0" xmlns:xsl="http:/
Add...
Delete
Delete All
Edit...
OK
Cancel



The Output tab displays the mappings between the multiple XSLTs and corresponding MBOs and attributes. An asterisk appears next to MBO names that have one-to-many relationships.



Creating an Entity Read Operation for a Mobile Business Object

Use the Operation Creation wizard to create an Entity Read operation for an existing mobile business object. Once created, you can edit Entity Read operations from the Properties view.

See *Immediate Refresh with Entity Read Data Flow* in *Mobile Data Models: Using Mobile Business Objects* for additional information about Entity Read operations.

1. Either:
   - 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, or
   - Select an existing MBO in the Mobile Application Diagram, select the **Operations** tab in the Properties view, and select **Add**.
2. Name the operation and select **ENTITY READ** as the operation type.
3. Select **Specify a data source** and click **Next**.
4. Define the Entity Read operation.

   The Definition screen varies depending on the type of data source to which the operation is bound. See the corresponding binding topic for detailed information.
5. (Optional) Click **Test Execute** to verify the operation against the data source to which it is bound. Click **Next**.

6. Configure the arguments and click **Next**.

   All primary key attributes must be mapped into arguments. See *Entity Read Guidelines and Limitations* for additional requirements.
7. Map all attributes, except those propagated from load arguments, to the EIS output results and click **Finish**.

You can now assign the "Apply output of ENTITY READ operation" cache policy option to Update and Create operations that use the "Immediately update the cache" cache policy.

Entity Read Guidelines and Limitations

Guidelines and limitations when configuring an Entity Read operation.

- You receive a warning if you define an Entity Read operation, but there is no Create or Update operation at the root of the composite graph using the "Apply output of ENTITY READ operation" cache policy option.
- You must fill all arguments of the Entity Read operation with a default value, personalization key or Fill-from-Attribute. Any unmapped arguments use the default value.
- You must map the entire MBO primary key into the input arguments for the Entity Read operation, except primary-key attributes propagated from load arguments. SAP Mobile WorkSpace displays a warning if all MBOs do not have a primary key.
- All MBOs in the composite graph must have all attributes mapped to results from the Entity Read operation, except attributes propagated from load arguments.
- For all EIS types, a composite graph can use only one data source.
- For all object-oriented EIS types (SAP, Web Service, REST) capable of shared read, a composite graph should use only one load group.
- An MBO can have only one Entity Read operation.
- For all EIS types, an Entity Read operation can only be defined for the root of a composite graph – an MBO graph can have more than one root (two nodes can have relationships with the same “child” MBO), but only one of the relationships can be a composite. Entity Read operations can only be defined on the root node of a composite object graph – since the Entity Read operation executes when the root operation executes, the results of subsequent child MBO operations are not incorporated into the cache through the Entity Read operation.
- An error is generated if any operation in a composite graph uses the “Apply output of ENTITY READ operation” cache policy, without an Entity Read operation defined.
- Entity Read operations are not available to MBOs that belong to an Online cache group. If an MBO has an Entity Read operation and the MBO is moved to an Online cache group, the Entity Read operation is removed.
- A child node in a composite graph cannot use the “Apply output of ENTITY READ operation” option.

Defining Output Mappings

You can map results returned from a remote operation into an MBO, including mapping multiple results to the MBOs of a composite graph, for example, when configuring Entity Read Operations.

The Output screen includes an output mapping figure that illustrates the mappings and allows you to modify them in these ways:

1. When an MBO developer maps an MBO to an XSLT, output table, or result set, a line connects the two objects, and the attributes of the MBO are mapped to the corresponding columns automatically according to column-to-attribute matching.
2. You can also change the mapping between individual attributes and columns.
3. If you remove the mapping between an MBO and the output, all the individual attribute and column mappings between the MBO and the output are removed.

Additional guidelines include:

- For SAP and Web services, SAP Mobile WorkSpace retrieves the output metadata (columns of the operation result) without executing an operation, but for JDBC the operation is executed to retrieve the metadata.
- Each mapped result set must be a one-to-one mapping to the MBOs in the composite graph (relationship). This is also true for Entity Read operations, the difference is that Entity Read's output mapping must cover all nodes in the composite graph. An error displays if you map columns from one result set into attributes of different MBOs, or if you map columns from different result sets into attributes of the same MBO.
- The Entity Read operation supports only a one-to-one mapping between an output column and MBO attribute.
- The datatype and nullability of an attribute must match that of the column to which it maps.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Creating Relationships Between Mobile Business Objects* on page 166

Composite Operations

The MBO designer can model a create, update, or delete operation on the root node of a composite relationship graph that accepts the entire composite graph as input.

The composite operation allows mapping of EIS operation arguments to any or all of the MBOs contained in the composite MBO graph such that child nodes are not required as explicit parameters to parent operations. Creation, deletion, or modification of child nodes in a composite operation results in the addition of these nodes to the operation replay graph sent to SAP Mobile Server when the composite operation on the root MBO is submitted. Composite operations involving child nodes in a composite graph do not result in separate replay operations for the child nodes. In other words, the device application developer can create a composite MBO tree in one create call from the root MBO.

You can define CUD operations on child MBOs in a composite graph, but those operations are not (and are not part of) composite operations and must be explicitly invoked by the client (with their own replay operations).

The operation replay includes only composite graph nodes which have changed in the operation replay message, which minimizes network traffic from the client device. SAP Mobile Server is responsible for assembling any missing portions of the graph from the cache prior to invoking the EIS operation.

After the EIS operation executes, the operation cache policy determines how those results are applied to the SAP Mobile Server cache.

See also

- *Creating Attributes for a Mobile Business Object* on page 133
- *Creating Operations for a Mobile Business Object* on page 134
- *Creating the Mobile Business Object using the Mobile Business Object Palette item* on page 77
- *Previewing Mobile Business Objects* on page 163
- *Modifying Load Arguments* on page 188
- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Creating Relationships Between Mobile Business Objects* on page 166

Creating a Composite Operation

Create a composite operation by defining composite relationships between MBOs, then define input mappings for the operation on the root MBO to include child nodes of the composite object graph.

See *Composite Operation Data Flow* in *Mobile Data Models: Using Mobile Business Objects* for additional information about composite operations.

1. Define composite relationships between MBOs.
2. Add an operation. For example, you could select the **Operation** icon from the Palette and add it to the root MBO of the composite graph. Name the operation and select the Type.
3. Specify a data source and click **Next**.
4. Specify the definition for the operation and click **Next**.

   For example, for a REST Web service, you would define a request representation that specifies the operation's details.
5. Define the input mappings for the operation to include child MBOs from the composite graph.

   You can manipulate input mappings (including adding client parameters and setting default values) after the operation is created, from the Properties view.

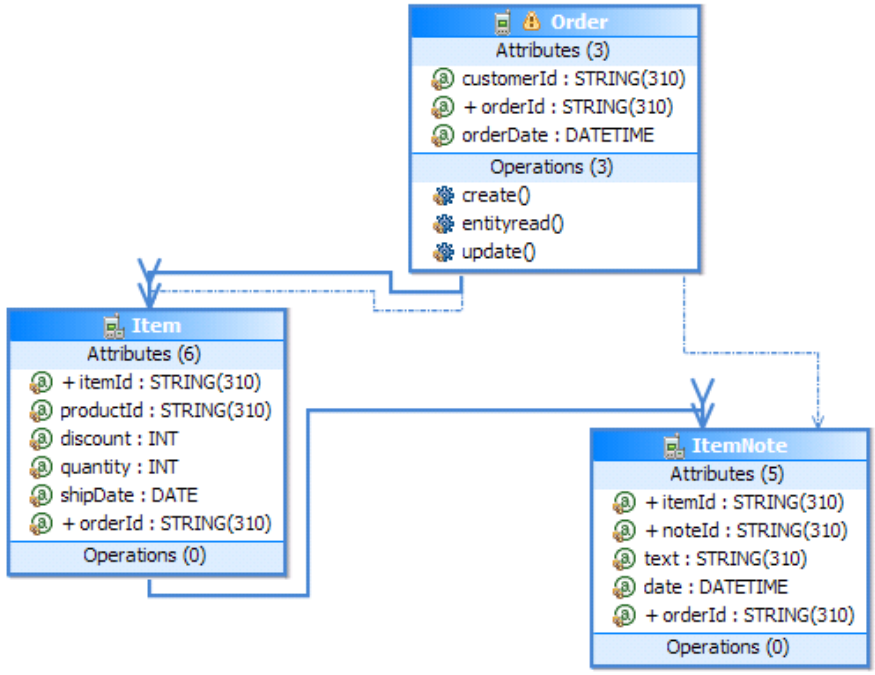
Composite Operation Example

This example illustrates a composite operation configured against REST Web service MBOs.

The *Mobile Business Object: restdemo Example Project* illustrates SAP Mobile Platform features including input mapping and composite operations, and can be downloaded and installed from the SAP® Community Network: *http://scn.sap.com/docs/DOC-8803*.

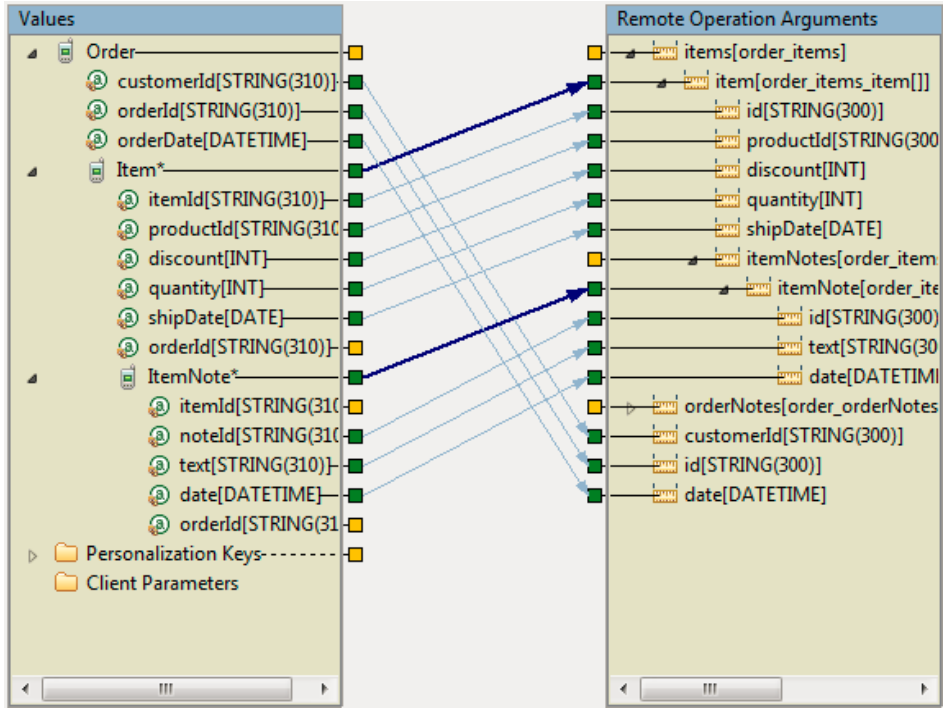
This example shows a composite graph in which:

- The Order and Item MBOs have a one-to-many composite relationship through their orderId attributes.
- The Item and ItemNote MBOs have a one-to-many composite relationship through their orderId and itemId attributes.



The Order MBO is the composite parent to the Item MBO, which is the composite parent of the ItemNote MBO. You can view the hierarchy for the Order MBO Update operation and modify the default input mappings from the operation's Properties view Input tab:

- Input Mapping tab – view or modify the mappings from values into remote operation arguments. An asterisk appears next to MBO names that have one-to-many relationships. This example illustrates a composite operation (input mappings from parent and child MBOs):



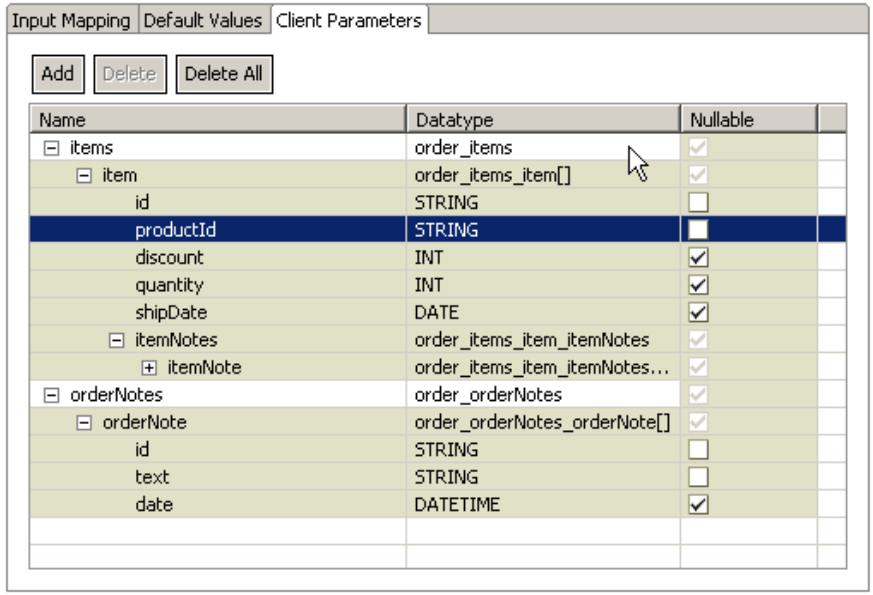
- Default Values tab – view or modify the default values for the remote operation arguments and child elements:

Input Mapping | Default Values | Client Parameters

Refresh

Argument	Datatype	Nullable	Old Value Argum...	Default Value
items	order_items	☑		{[{"","",<NULL>,<NULL>,<NULL>
item	order_items_item[]	☑		<NULL>
id	STRING	☐		
productId	STRING	☐		
discount	INT	☑		<NULL>
quantity	INT	☑		<NULL>
shipDate	DATE	☑		<NULL>
itemNotes	order_items_item...	☑		<NULL>
orderNotes	order_orderNotes	☑		{[{"","",<NULL>}]}
orderNote	order_orderNotes...	☑		<NULL>
customerId	STRING	☐		
id	STRING	☐		
date	DATETIME	☑		<NULL>

- Client Parameters tab – view or modify client parameters. Client parameters can be mapped into remote operation arguments on the Input Mapping tab:



Note: Defining client parameters on this tab does not automatically pass them to the EIS, it just defines them; they can then be used to define input mappings.

Composite Operation Requirements and Limitations

Understand requirements and limitations when modeling composite operations.

- Composite operations support client parameters, personalization keys, default values, and composite MBO attributes bound to a single composite operation invocation.
- In general it is possible to combine any number of child updates, creates, and deletes in a single operation replay message. A complete composite graph of arbitrary depth and breadth can be constructed (potentially spanning save operations) and be submitted as a single replay operation. The client API constructs the replay message such that it contains all the modified nodes in the graph and their ancestor nodes.
- SAP Mobile Server checks the graph version in the cache and compares it to the graph version in the operation replay message to be sure they agree prior to invoking the EIS. If the versions do not match, a conflict is detected and an error is generated in the SAP Mobile Server log file.
- Propagation of load arguments into MBO attributes is only supported on the root node of an MBO graph.
- Composite operations are only allowed on the root of a composite graph.

  Operations defined on a child MBO in a composite graph only allow mapping the attributes of the child MBO when defining the input mappings.
- JDBC does not support composite operations.

- Entity Read operations cannot be composite.
- Operations cannot have the names create, update, or delete for a child MBO in a composite graph, if the child MBO is used in a composite operation.

Defining Input Mappings

Allows mapping of remote operation arguments to input values, including all nodes of a composite graph.

The Input Mapping screen includes an input mapping figure that allows the MBO developer to work with input mappings in these ways:

1. Add a link from an MBO to an argument structure, and individual mappings from attributes into sub-arguments are added based on matching name and type. Some automatic mappings may be added incorrectly, for example, if you had changed the name or data type of attributes.
2. Removing a link from an MBO removes all individual mappings from the attributes of the MBO.
3. Add or remove individual mappings from attributes into arguments.
4. Add or remove mappings from Personalization Keys into arguments or argument structures. Dragging the argument or argument structure to the left hand side and dropping it onto the Personalization Keys folder opens the New Personalization Key wizard, with the type defaulted to match the argument type. Finishing the New Personalization Key wizard creates the Personalization Key, and adds a mapping into the argument.
5. Add or modify Personalization Keys in the WorkSpace Navigator, and the changes will be reflected in the Input Mapping tab.
6. Add or remove mappings from Client Parameters into root arguments. Dragging the argument or argument structure to the left hand side and dropping it onto the Client Parameters folder creates a Client Parameter matching the type of the argument, and adds a mapping into the argument. When you map a Client Parameter into an argument, the type of the Client Parameter is changed to match the type of the argument. You can add or modify Client Parameters on the Client Parameters tab, and the changes will be reflected in the Input Mapping tab.

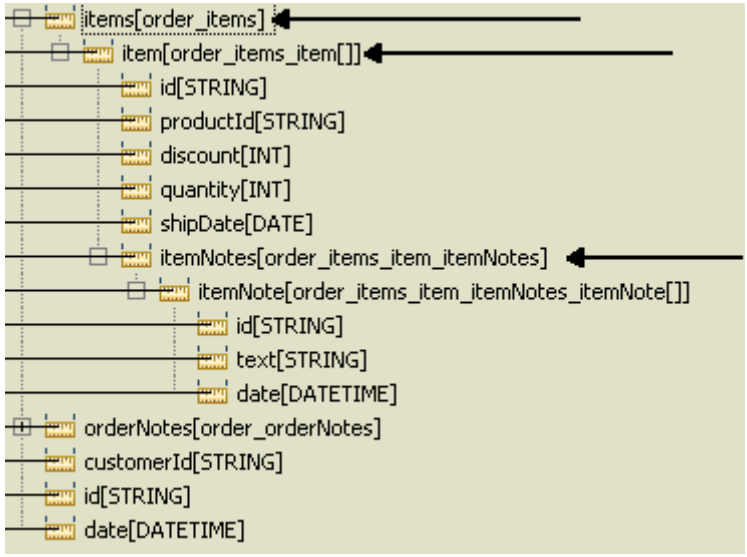
Composite Operation and Input Mapping Terminology and Configuration Guidelines

Understand composite operation related terminology to better understand how to configure composite operation input mappings.

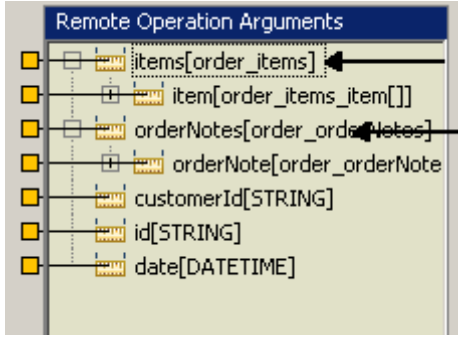
The example MBO project used to illustrate the following terminology is located at the SAP® Community Network: *http://scn.sap.com/docs/DOC-8803*.

- **EIS operations accept one or more EIS arguments –** an EIS argument is conceptually similar to a parameter accepted by a Java method call. The physical representation of an EIS argument varies with each endpoint type. For example, the interface to a REST Web service is defined via an XSD and each root element contained in the XSD represents an EIS argument.

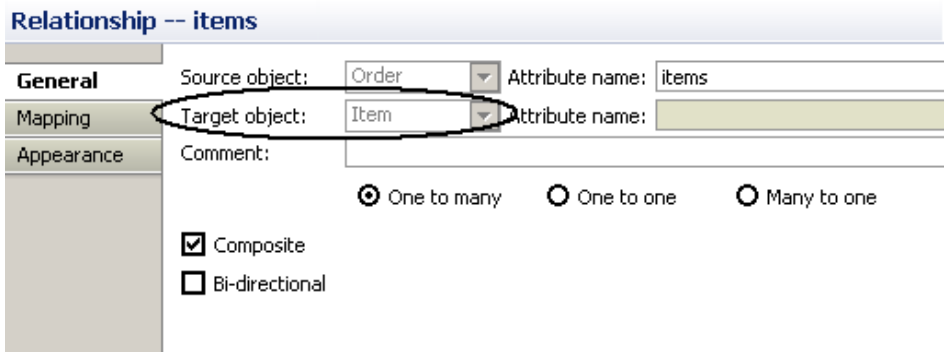
- **EIS arguments have one or more levels such that child levels are nested under parent levels –** EIS arguments are represented as a hierarchy consisting of one or more levels:



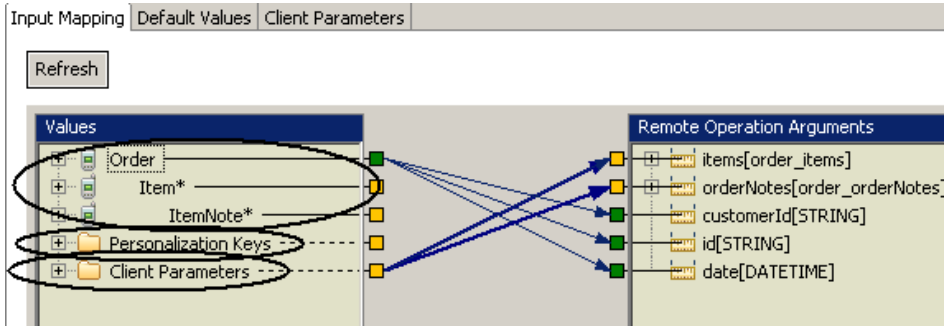
- **Top level EIS argument –** the root level of the EIS argument hierarchy:



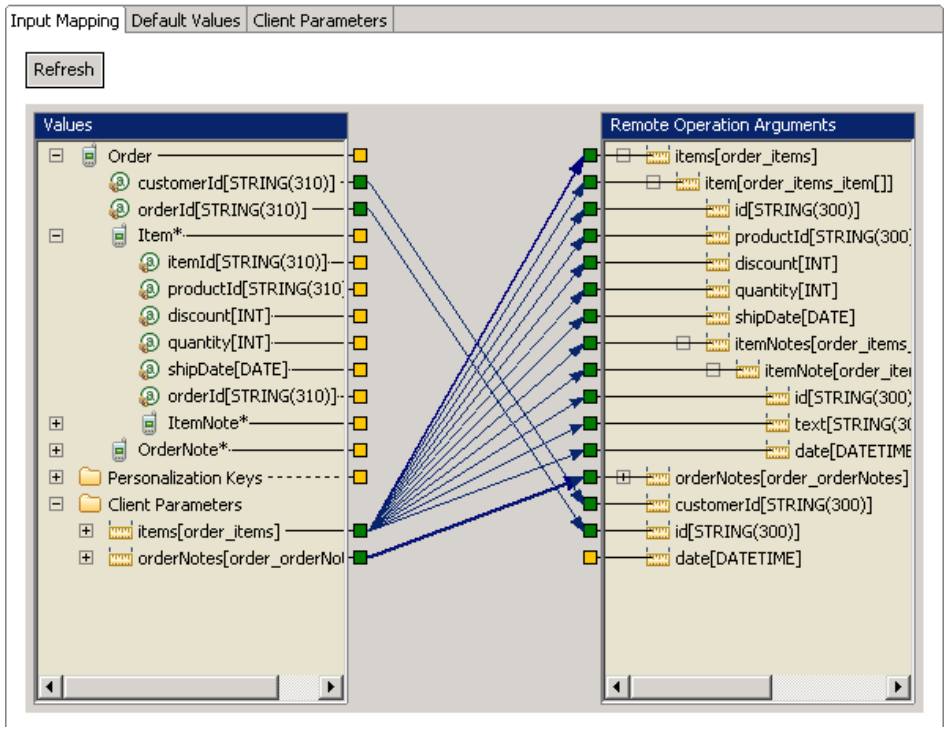
- **Repeating MBO –** an MBO which is on the many side of a relationship in the composite graph, or which has an ancestor in the composite graph which is on the many side of a relationship.



- **Input source –** the input source is one of the following: an MBO composite graph, a client parameter, or a personalization key.

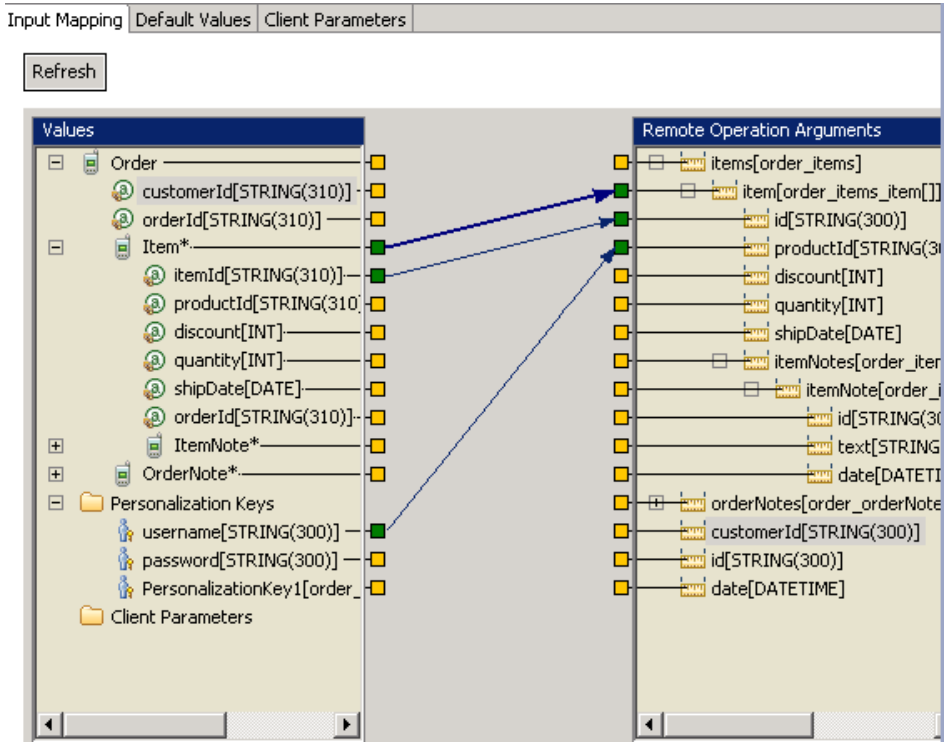


Client parameters can be mapped only to the top level (not fields in the top level) of an EIS argument:



To map a personalization key into a repeating argument element, there must be an attribute mapped into the same level.

For example, using the REST example MBO model, with the arguments required by an operation on the Order MBO: A personalization key cannot be mapped into items.item.productid unless some repeating attribute is mapped into one of the other primitives contained by items.item:



Note: SAP Mobile WorkSpace version 2.2 SP01 and later does not allow manually adding links from personalization keys and client parameters into the same remote operation argument. If a migrated project includes that scenario, SAP Mobile WorkSpace maintains the linking but generates a warning message similar to:

```
Client parameter 'parameterName' might not be used,
 as the mapped argument has 'Fill from Attribute' or
'Personalization Key' specified."
```

After migrating the project, remove the additional link (either the personalization key or the client parameter) from the input mapping. In cases where both a client parameter and personalization key are mapped, the personalization key value is used. SAP Mobile WorkSpace no longer allows mapping attributes into arguments that are already mapped to personalization keys or client parameters.

Input Mapping Default Value Precedence

Understand default value usage and runtime precedence during remote operation execution as they relate to input mappings.

View operation input default value properties by selecting the operation in the Mobile Application Diagram, then selecting **Input** > **Default Values** from the Properties view:

- Argument – name of the operation argument.

When an old value argument is unmapped, it is added to the list of arguments in the Default Values tab, and the Arguments in the Input Mapping tab. Personalization keys and client parameters may be mapped into unmapped old value arguments in the Input Mapping figure, but not attributes.

- Datatype – the argument's datatype.
- Nullable – selected if Null is a valid value. If a create, update, or delete (CUD) argument is mapped to an attribute and to a personalization key, you cannot set < NULL> value for this argument. If you want to pass < NULL> value for an operation argument which is fill-from-attribute, model additional operations without personalization keys or default values bound to the argument.
- Old value argument (update and delete operations mapped to JDBC data sources only) – maps a parameter to a second (old) argument for MBOs. The old value argument is assigned the previous value of the attribute to which the corresponding operation argument is bound. See *Old Value Argument*.
- Default Value – the argument's default value.
  For MBO operations with a non-nullable argument, the argument's value must come from either a fill-from-attribute, personalization key, client parameter, or default value. If a non-nullable argument maps to a fill-from-attribute or client parameter, the default value is ignored, even though SAP Mobile WorkSpace allows you to input a default value. Also, if the personalization key or fill-from-attribute is used to fill the value of the argument, the client parameter might not be used.

Multi-level Insert Operations

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

Note: For non-JDBC datasources, SAP recommends using composite operations for creating and updating MBOs within a composite object graph, since they are more efficient and provide better overall performance.

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 *Developer Guide: <Device Platform> Object API Applications* 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, SAP Mobile 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;
   ```

   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 parent'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 parent and child in which the source attribute/parameter in parent is a primary key and the target parameter in child is a foreign key to the parent.
- 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: SAP Mobile 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 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 foreign 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 Customer MBO and add a **select** statement that returns the "@@identity" row. The ID column returned via this select query serves as the ID used by the chained insert statement into the Sales_Order 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.

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.

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

- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Previewing Mobile Business Object Attributes

Configure load arguments 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.
3. From the Preview dialog, configure the attribute parameters used to test and preview the mobile business object and preview the results.

Table 37. Preview Dialog Instructions

Field	Description
Load arguments	• Argument name – (read-only) the name of the argument. • Datatype – (read-only) the parameter's datatype. • Nullable – (read-only) accepts NULL 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, bigbinary, bigstring, date, datetime, or time, select the input dialog in which you can select a value. If the datatype is TABLE, click 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. The parameter displays only if you have defined it; otherwise the table is empty. You can order the load arguments by selecting the column heading by which you want to order.
Existing Configurations	Select either an existing configuration that contains attribute parameter value settings, or preview without using a configuration: • 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.
Preview	Select **Preview** to pass the parameters to the data source and view the results in the Preview Result screen. **Note:** • If the MBO definition requires a default value, and you do not provide one, the data source returns an error which SAP Mobile WorkSpace displays, when you select Preview. • By default, 100 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.

Field	Description
Save as default values	Saves the argument values during the preview as default values for the load argument, which are reflected in the Load Arguments tab in the Properties view.

Testing Mobile Business Object Operations

Configure operation arguments 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, or from the operation's Definition tab in the Properties view.
3. From the Test Execute dialog, configure the operation arguments used to test the operation.

Table 38. Test Execute Dialog Instructions

Field	Description
Operation arguments	• Argument name – (read-only) the name of the data source argument. • Datatype – (read-only) the argument's datatype. • Nullable – (read-only) accepts NULL if selected. • value – supply a value for the operation argument 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, bigbinary, bigstring, date, datetime, or time, select the input dialog in which you can select a value. You can order the operation arguments by selecting the column heading by which you want to order.
Existing Configurations	Select either an existing configuration that contains test settings, or test without using a configuration: • 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.

Field	Description
Test Execute	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.
Save as default values	The argument values provided during test execution are used as the operation argument default values, and reflected in the Operation Parameters tab in the Properties view.

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 (also called an MBO graph, object graph, or composite object graph for composite relationships). You can also create relationships between MBOs through load arguments.

Task

Use relationships to associate related data between MBOs and maintain data synchronization on the mobile application.

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:
   - **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, load argument, or structured load argument and dragging it to the corresponding target to which you want to map. Structured load argument relationships are not true relationships, but allows mapping of structures. SAP Mobile WorkSpace also supports attribute-to-load argument, load argument-to-attribute, and load argument-to-load argument mappings.
4. Click **OK** when finished.

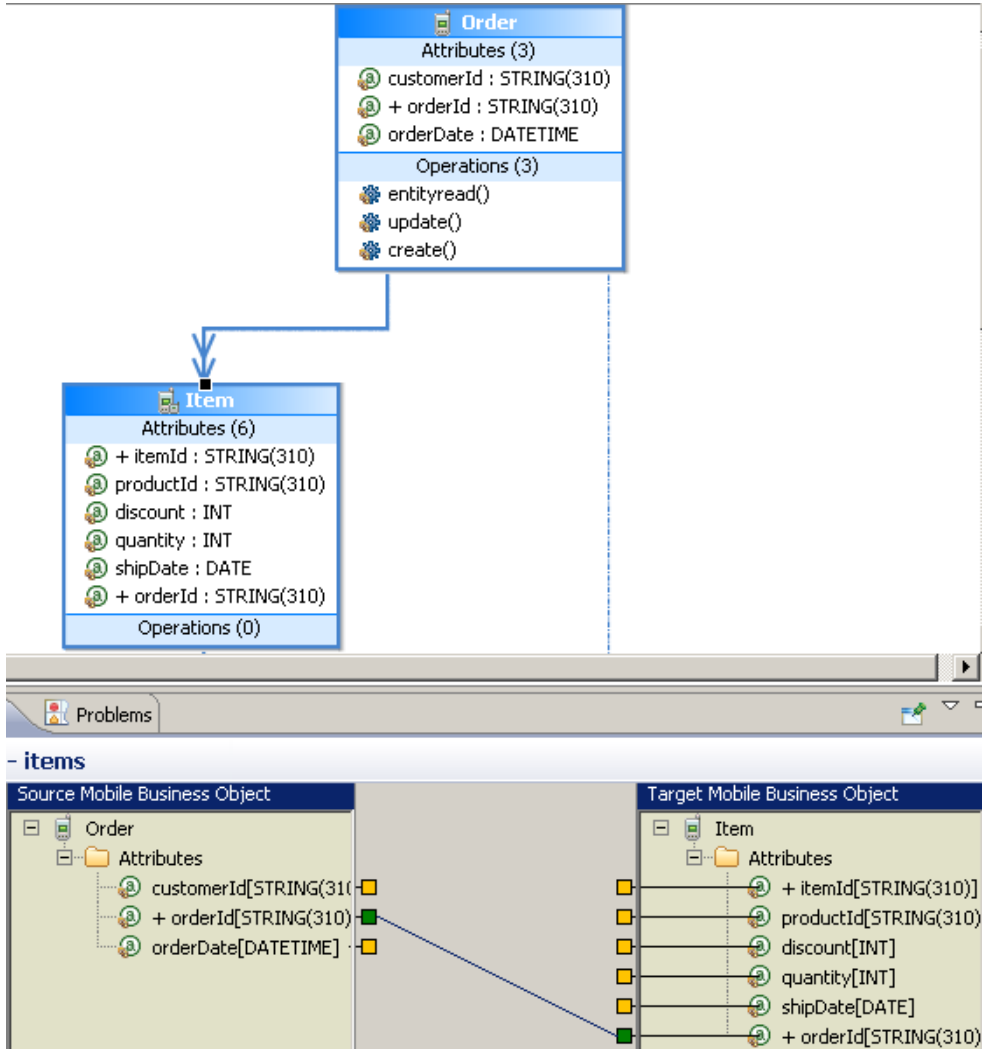
   Relationships are identified in the Mobile Application Diagram by a line connecting the related MBOs, with a different type of arrow for each type of relationship (one-to-one, one-to-many, or many-to-one).

One to many, unidirectional, composite

This example illustrates a relationship between the Order MBO and Item MBO:

- **One to many –** a single order can contain many items. The orderId identifies the order, and the Item includes the orderId attribute, whose value identifies the order to which it belongs.
- **Composite –** changes to the order are propagated to the Item MBO. For example, a new item is added to the order.

- **Unidirectional –** updates flow from Order to Item only.



See also

- *Composite Operations* on page 148
- *Defining Output Mappings* on page 147

Mobile Business Object Relationship Properties

Modify the relationship of two mobile business objects using the Properties view.

Table 39. Relationship Properties

Tab	Contents
General	• 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.
Mapping	View or change the mappings for this relationship. While attribute-to-attribute mapping is explicitly supported, you can also map: • Argument-to-argument mapping, including structure arguments (structure MBOs) – requires that you set a **propagate to attribute** for the argument, then map the argument from the relationship mapping tab. For example, the Customer MBO has an attribute named state and an argument named state_name, from the attributes Load Arguments tab, select state as the propagate to attribute for the state_name argument. Then map the state_name argument in a relationship. You must map sub-arguments when mapping structured arguments. For example, if Customer[] is the structured argument, and it contains state as a sub-argument, you must map the state sub-argument, and not the Customer[] argument. • Attribute-to-argument and argument-to-attribute mappings are also supported.

Tab	Contents
Appearance	Modify the appearance of the text and line that represents the relationship. For example: • 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 and Restrictions

Follow these guidelines when configuring MBOs and enterprise information systems (EISs) used in relationships.

Table 40. Relationship Guidelines

Guideline	Example

These examples use SQL Anywhere to illustrate how to properly configure the EIS so delete operations behave as expected	• Example one consists of two MBOs:
	`Address: addr_id, street, city, country`
	
	`Employee: emp_id, name, addr_id (foreign key references Address.addr_id)`
	
	The MBOs have a relationship defined, for example, a composite, uni-directional, one-to-one relationship:
	`Employee(Parent) --> Address(Child)`
	
	At runtime, if you delete Employee, SAP Mobile 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.
	This requirement applies regardless of the type of relationship defined between the MBOs.
	• Example two also uses the Employee and Address MBOs in a relationship that has mutual references:
	`Address: addr_id, street, city, country, emp_id(foreign key references Employee.emp_id)`
	
	`Employee: emp_id, name, addr_id(foreign key references Address.addr_id)`
	
	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.

If the foreign key in a relationship points to a `char(n)` column, and it is the only column in the table, use the **rtrim(column name)** function in the MBOs column definition	This example has a relationship defined as:
	`States(Parent) --> Sales_region(Child)`
	
	Sales_region contains one column named Region with a data type of char(7). The MBO definition should be:
	`SELECT rtrim(region) as region`
	`FROM sampledb.dba.sales_regions`

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 `sales` MBO, which changes frequently, that has a relationship with the `catalog` MBO, which changes infrequently.
SAP Mobile WorkSpace does not support Circular relationships	Do not create circular relationships that work from parent-to-child then, eventually, back to the parent. For example: A relationship that includes three MBOs: A, B, and C, where `A----->B----->C----->A` is not supported.

Guideline	Example

Composite relationship (composite graph) behavior	• A child object refers to its parent via a foreign key reference from the child to a primary key value in the parent. When modeling your relationships, choose load parameters carefully to load children based on which parents are in the cache, so that foreign key references can always be resolved. At runtime SAP Mobile Server must be able to resolve foreign key references. A poorly modeled relationship could have MBO instances pointed to by foreign keys that have not been loaded into the cache. Examples include:
	  • A composite parent that hasn't been loaded which results in the child not being inserted into the cache because composite orphans are not allowed – a child MBO can have multiple parents (many to one), but only one parent can have a composite relationship to the child:
	  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, SAP Mobile WorkSpace randomly selects a parent through which the child synchronizes. SAP recommends that you define one parent MBO as the composite parent to avoid unintended behavior. For example:
	  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.
	  • An MBO modeling error where the relationship reference cannot be resolved results in an error message. For example:
	  • The SAP Mobile Server log indicates a Null foreign surrogate key value where a child in a non-composite relationship points to a non-existent parent MBO, which could indicate that the MBO is a child in two separate relationships (one composite and one non-composite) and load parameters are improperly configured:
	  `Serious performance and data integrity issues may ensue.`
	`Consider modifying/re-sequencing cache definitions and/or load queries`
	`to ensure relationship integrity in the cache.`
	  • SAP Control Center error log:
	  `Composite parent not found`
	• 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.

Guideline	Example
	• MBOs in a composite parent/child relationship where the child is loaded into the SAP Mobile Server cache without a parent (composite orphans) are not supported. For example, a composite relationship between department(parent) and employees(child) that attempts to load employees for which there is no corresponding department is not allowed at runtime. In this scenario you could change the model to use a non-composite relationship or change the MBO model so the application always loads the parent entities before the children. • 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. • A composite graph should not span multiple load groups if loading from a structure-oriented EIS. • A composite graph should not span more than one endpoint definition.
First/subsequent MBO relationship behavior	Multiple MBOs created from a single operation result: • Subsequent MBOs as child entities (if the relationship mapping is based on a subsequent MBO load argument) 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 argument of the first MBO are not supported.
Local business objects	• A relationship between two local business objects that does not have a primary key set is invalid. You cannot create relationships between local business objects using a composite primary key.
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.

Guideline	Example
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**, SAP Mobile WorkSpace automatically sets the primary key, if not, SAP Mobile 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 argument of a child MBO, the personalization key and default value cell of the load argument 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 runtime.
It is a modeling error to have multiple relationships with the same set of links (every link has the same source and target attribute), for every relationship type (one-to-one, one to many, many to one, bidirectional, or composite), and generates a validation error.	For example, a relationship exists between customer and salesorder MBOs as: customer->id to salesorder->cust_id. The MBO developer cannot create another relationship using the same link: customer->id to salesorder->cust_id. Instead, either change the link or add new links to create the second relationship.

Refreshing Attributes and Arguments

The refresh option generates new mappings for attribute/columns or operation arguments/ load arguments based on the metadata of the data source to which the mobile business object (MBO) is bound

For MBOs already bound to data sources, selecting **Refresh** refreshes and/or remaps operation arguments, load arguments, and attributes based on the EIS data source metadata changes. Refresh also affects how Preview refreshes attributes and operation arguments. Internally, selecting **Refresh**:

- Queries the data source metadata from scratch.
- Creates new columns or arguments using the fresh metadata information (name, type, default value).
- Generates new data source mappings for attributes, load arguments, or operation arguments.
- Establishes "Fill from attribute" links for operation arguments.

Input and Output Mapping pages – refresh does not actually update metadata from the EIS. In order to get current metadata, the MBO developer edits the operation definition: it is not necessary to make any changes, just open the Edit Definition dialog, and click **OK**. At that point, a refresh prompt appears. Choosing:

- **Yes** (or clicking Don't Show again, and Edits the definition again in the future) – updates attributes/arguments.
- **No** – the MBO developer can later update attributes/arguments using Refresh on the Input/Output tabs. But until the MBO developer does so, the Input/Output tabs do not reflect any EIS changes.

You have an MBO with attributes bound 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 EIS 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. Note that after you refresh, the "Address" attribute is unbound. You can manually add attributes and map them to the columns in the tabular view, or select **Remap** to create the new attributes and map 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 or String whose length is larger than 300.

       Note: If you add a default value, for these data types, the default value is initially read only. To edit this value, click the elipsis (...) 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 SAP Mobile Server
     - Client – on the device client
     - Transient – only saved in memory of the current login session
   - Description – (optional) the description of this personalization key

Personalization Key Example

This example demonstrates how the personalization key value storage settings affect mobile application behavior.

Personalization values are scoped to a named user. For example, the mobile application developer creates a weather application that includes a personalization key named "zipcode". In this example:

1. John sets the zipcode personalization value to "94568" and automatically gets the weather for Dublin, CA. If the weather application is installed on both John's iPhone and laptop and storage is set to "server", the value is shared across all his applications.

   If John sets the zipcode to 94301 from his iPhone, the next time he views the weather application from his laptop, it shows the weather for Palo Alto instead of Dublin.
2. If storage is set to "client", each instance of the application maintains its own setting for the value. So if John changes the zipcode on his iPhone to 94301, the laptop still shows Dublin weather.
3. If storage is set to "transient", the value must be reset each time the application runs - it is not persistently stored anywhere.

   If the weather application used "GPSLocation" personalization key instead of zipcode, the weather application could retrieve current location using iOS native O/S calls to set the transient GPSLocation personalization value before synchronizing the weather MBO.

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**.
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 Guidelines and Limitations

Understand the limitations when configuring MBOs that contain personalization keys.

- **Maximum length –** the value of personalization keys are BASE64 encoded. After encoding, the sum of the length of the personalization key values cannot exceed 16*4000, or a synchronization exception occurs during synchronization.
- **Personalization keys mapped to parameters and arguments –** if a personalization key is nullable and does not have a default value, and is mapped to an MBO sync parameter, load argument, or operation's argument, the default value for that sync parameter, load argument, or operation's argument can be modified, otherwise the default value is empty and read-only.
- **Mapping a personalization key to operation/synchronization parameter/load argument –** if a personalization key is mapped to either an operation argument, load argument, or synchronization parameter, the Properties view default value selection is disabled and the default value is removed.
- **Setting the personalization key in the device application –** the priority with regard to the device application user setting the personalization key value is:

  1. The user sets the personalization key value. If the personalization key equals NULL,
  2. Use the personalization key default value. If the personalization key default value equals NULL,
  3. Use the parameter default value.

  In cases where the personalization key is not nullable:

  1. the default value for the parameter is never to be used and the default value for the synchronization/operation parameter is read-only.

2. The actual synchronization/operation parameter is not visible from the device application. The device application user cannot overwrite the value with any arbitrary value.
3. Updating the personalization key value results in an update of the original subscription, and not the creation of a second subscription.

Managing Roles and Permissions

Logical roles provide authorization for mobile business objects and operations during development.

The SAP Mobile 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 SAP Mobile 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 SAP Mobile Server, you can map logical roles to existing physical roles that are located on the SAP Mobile 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 SAP Mobile Server, since it is likely the SAP Mobile 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 SAP Mobile Servers do use the same set of roles, you can map the logical role name directly to the physical role when deploying the MBO. See *Packaging and Deploying Mobile Business Objects*.

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** > **SAP** > **Mobile Development** > **Role**.
2. Specify:

Table 41. 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 SAP Mobile Server and device applications access MBO data.

Synchronization

Determine the amount of data (filter), and under what conditions (timing and triggers), mobile devices upload MBO data to and download data from the SAP Mobile Server cache (CDB).

Synchronization properties are unavailable for MBOs in cache groups that use an Online policy.

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 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 parameter configuration:

   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.

Note: Synchronization parameters do not support structure types.

- **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.
- **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.
- **Default value** – enter a valid value.
- **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.

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 '>'.

This example illustrates default synchronization behavior:

1. Create an MBO with this SQL definition for a database MBO using the My Sample Database, which creates a load argument for state:

   ```
   select * from customer where state = :state
   ```
2. Create a synchronization parameter and map the state load argument to this new synchronization parameter.
3. The device application user enters CA and synchronizes the first time. The application displays all CA customers.
4. The user then enters NY and synchronizes, which displays CA and NY customers.

Next

(Optional) For a parameter to serve as both a load argument and a synchronization parameter, select **Synchronization parameter** for the argument to which the synchronization parameter is mapped from the attribute's **Load Arguments** tab after you define the synchronization parameter.

See also

- *Combining Load Arguments and Synchronization Parameters* on page 190
- *Load Arguments* on page 187

Customizing Data Download with TOP, ORDER BY, and UNION

Customize data downloads using TOP, ORDER BY, and UNION SQL operations.

By default, the internally generated download SQL statement, based on the synchronization parameters, includes only the '=' operator. For example:

```
SELECT * FROM entity a WHERE a.x = :x and a.y = :y
```

SAP Mobile Platform allows customization of the download data SQL through SAP Mobile WorkSpace to support:

- Where clause operators other than '='. For example, (> , < , < > , != , > =, < =, ! > , ! <, and so on).
- Joining multiple entities using ‘UNION’, ‘TOP’ and ‘ORDER BY’ keywords in the customized SQL, and supporting them in the download-data SQL.

1. Define synchronization parameters and customized data download SQL in SAP Mobile WorkSpace using the desired SQL operations. For example, “TOP”, “>=”, <=” and “ORDER BY”, and deploy the MBO package to SAP Mobile Server.

   For example, the Employee MBO contains three synchronization parameters: topNum, salary, and startDate along with other attributes. From the Synchronization tab in the Properties view, you could select **Customized download data** and enter this SQL statement:

   ```
   SELECT TOP :topNum x.*FROM Employee x WHERE x.start_date
   >= :startDate
   AND x.salary <= :salary ORDER BY x.emp_lname
   ```

2. create a native device application, and specify the synchronization parameters values to download data to the device. For example:

   ```
   EmployeeSynchronizationParameters empSP =
         Employee.GetSynchronizationParameters();
   empSP.StartDate = new DateTime(1989, 07, 01);
   empSP.Salary = 50000.00M;
   empSP.TopNum = 5;
   empSP.Save();
   TestDB.SubmitPendingOperations();
   TestDB.Synchronize();
   ```

   This downloads the top five records where start_date >= 1989/07/01 and salary <= 50000.00 to the device. There are 43 records in the sampledb that match the query:

   ```
   SELECT x.* FROM Employee x WHERE x.start_date >= ‘1989-07-01’ AND
   x.salary  <= 50000.00 ORDER BY x.emp_lname
   ```

Note: TOP and ORDER BY should always be used together:

- Using TOP without ORDER BY results in undetermined returned values.

- Using ORDER BY without TOP adds no value since it does not change the data set to be downloaded.

Creating Synchronization Groups

A synchronization group includes a policy that defines the synchronization schedule for the mobile business objects (MBOs) within it. 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:
   - Switch to the Advanced developer profile, then right-click the Synchronization Groups folder under a mobile application project from the WorkSpace Navigator, and select **New > Synchronization group**, or
   - From SAP Mobile 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 SAP Mobile 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. Change the default synchronization group by selecting this option for the group you want to define as the 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.

     SAP Mobile WorkSpace includes a "Default" synchronization groups folder (with a change detection level set to 10 minutes), which ensures there is always at least one folder available.
3. The properties for all synchronization groups can be modified by right-clicking the synchronization group folder and selecting **Properties**.

The synchronization group folder you just added appears under the Synchronization groups parent folder in the mobile application project for which it was created. You can drag-and-drop MBOs between synchronization group folders.

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 Arguments

Load arguments control the amount of data refreshed between the enterprise information system (EIS) and the cache database (CDB), and each load argument creates its own client defined partition in the CDB based on load argument value (partition key). Partitions are refreshed concurrently, thus improving performance. In contrast, synchronization parameters filter CDB data downloaded to the mobile device during device application synchronization.

Set load arguments in the Properties view, from the **Attributes > Load Arguments** 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 use:

- A synchronization parameter and a separate load argument – refresh data based on an argument independent of synchronization, or
- A load argument that maps to a synchronization parameter – use the same value for both refreshing and synchronizing data. Basically, one synchronization parameter induces one client defined partition. This provides more fine-grained CDB partitioning and concurrency, but may introduce more partition refresh overhead and less data sharing across devices when there are too many different values from synchronization parameters.

Figure 1: Synchronization Parameter

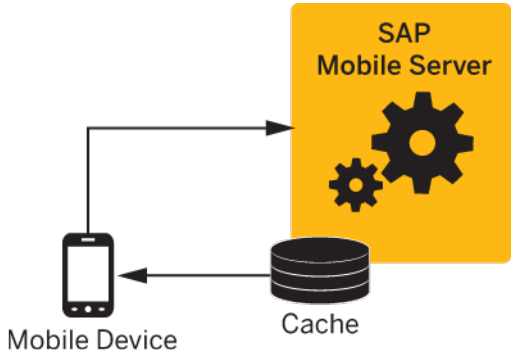
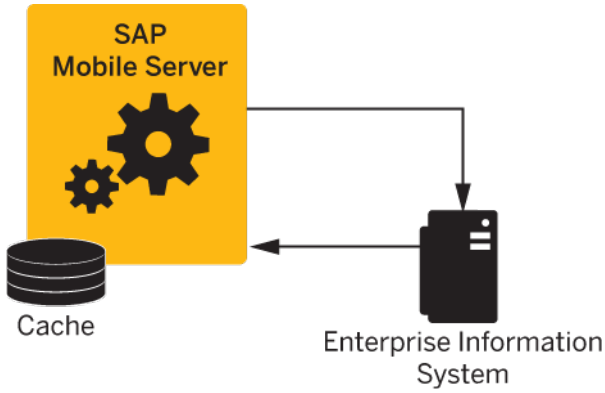


Figure 2: Load Argument



See also

- *Mobile Business Object Attribute Properties* on page 134
- *Defining Synchronization Properties for Individual Mobile Business Objects* on page 183

Modifying Load Arguments

Manage certain aspects of load arguments such as mapping them to synchronization parameters or personalization keys.

Use load arguments to control data refresh, create client defined 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 arguments with operation arguments, 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 Arguments** tab.

   Any arguments defined as part of the MBO definition automatically appear as load arguments, and you cannot arbitrarily add new arguments. 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 argument and **state** is the **Propagate to attribute**. You can modify load arguments from the Load Arguments tab.

3. For each argument, you can specify:

Property	Description
Data Source	
Argument	Data source argument name to which the load argument is mapped.
Datatype	Datatype of the data source argument to which the parameter is mapped. You can change the argument's datatype only if the MBO is bound to a JDBC data source.
Nullable	Select this option only for data source arguments that support NULL as a valid value.
Value	
Propagate to attribute	(Optional) Select an attribute to use its value for that of the argument. **Note:** When the MBO belongs to a cache group that uses an Online policy, the selected attribute is used as the query parameter of the automatically generated findByParameter object query.
Personalization key	(Optional) You can select a personalization key to map to the argument, 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.

Property	Description
Synchronization parameter	(Optional and unavailable for MBOs in cache groups that use an Online policy) Map the load argument to an existing synchronization parameter (defined from the Synchronization tab) to use the synchronization parameter's value to refresh and filter data between the EIS and CDB, and download to mobile devices. See *Combining Load Arguments and Synchronization Parameters* for details. The **Default value** and **Personalization key** options are ignored if you select synchronization parameter.
Default value	(Optional) Enter a default value for the parameter. If **Synchronization parameter** or **Personalization key** is selected, this option is unavailable.

See also

- *Mobile Business Object Properties* on page 130
- *Mobile Business Object Attribute Properties* on page 134
- *Mobile Business Object Operation Properties* on page 136
- *Datatype Support* on page 125
- *Old Value Argument* on page 138
- *Entity Read Operations* on page 140
- *Defining Output Mappings* on page 147
- *Composite Operations* on page 148

Combining Load Arguments and Synchronization Parameters

Combine load arguments and synchronization parameter settings to control how data is cached in the CDB, and filtered and returned to a device application.

Design and implement load and synchronization for efficient data control, especially where large sets of data are involved, or SAP Mobile Server and device applications may perform poorly. For example:

- SAP Mobile Server – load arguments 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 arguments 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 arguments or synchronization parameters are defined.	All table data is downloaded to the device. The SQL definition is: `select * from sampledb.dba.customer`
The region synchronization parameter is defined, but not used as a load argument.	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: `select cust_id, cust_name, region from sampledb.dba.customer where region=:region`
The region load argument is defined, and used 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 arguments 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: `select cust_id, cust_name, region from sampledb.dba.customer where region=:region`, for user A.

See also

- *Mobile Business Object Attribute Properties* on page 134
- *Defining Synchronization Properties for Individual Mobile Business Objects* on page 183

Mapping a Load Argument to a Synchronization Parameter

Map a load argument to a synchronization parameter to control both enterprise information system (EIS) caching on SAP Mobile Server when a mobile business object (MBO) is accessed by a device application and filter 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 Arguments** tab, located on the top.
3. To use the load argument for synchronization, select the **Synchronization parameter** from the drop-down list.

   If the load argument 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 SAP Mobile Server.

   If the parameter includes a default value, a client-supplied value is optional.

Configuring Mobile Business Objects for Hybrid App Online Data Access

You can define load arguments that determines what data to return to a Hybrid App application.

You can map load arguments to either transient personalization keys or propagate-to attributes to define load arguments used within a Hybrid App, 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 argument always comes from the 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 Load Arguments from Mapped Propagate to Attributes

Create an MBO with at least one load argument, map as propagate to attributes, then assign the MBO to a cache group that uses an Online policy.

1. From SAP Mobile WorkSpace, create an MBO that has at least one load argument. For example, you could define an Employee MBO as:

   ```
   SELECT emp_id, emp_fname, emp_lname, dept_id
   FROM sampledb.dba.employee WHERE dept_id = :deptIdLP
   ```

2. In the MBO Properties view, select the **Attributes > Load Arguments** tab, map each load argument to be used as an operation load argument for the Hybrid App package to a Propagate to Attribute. This example requires you to map the deptIdLP load argument to

the empDeptId attribute. You must also verify that data types are INT and the default value is a valid INT.

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 arguments that have propagate-to attributes:
4. Deploy the project that contains the MBO to SAP Mobile Server.

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 SAP Mobile 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 (cache interval) 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 arguments filled from transient personalization keys cannot be used with a scheduled cache type, because transient personalization key values are stored in the device application session, and unknown to SAP Mobile Server.
  - **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.

    You can define MBOs without any load operations (not bound to a datasource), only if the MBO belongs to a cache group that uses a DCN policy.

    See the *Mobile Data Models: Using Mobile Business Objects* for details about implementing DCN.
  - **Online –** only can be used with message-based Hybrid Apps. See *Online Cache Group Policy*.

- **EIS managed –** DCN controls cache refresh, the ability to define cache partitions, update MBO data within those partitions, and subscribe users to those partitions.

Each cache group contains a cache policy, which in turn contains cache refresh/update properties. When a refresh occurs, the SAP Mobile 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, SAP Mobile 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.

Creating Cache Groups

Each cache group has a cache group policy that defines the data refresh properties/schedule 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. To launch the New Cache Group wizard, either:
   - Right-click the Cache Groups folder from the WorkSpace Navigator, and select **New** > **Cache Group**, or
   - From SAP Mobile WorkSpace, select **File** > **New** > **Cache Group**.

   Note: If you do not see the Cache Groups folder, switch to the Advanced Developer profile.
2. 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 SAP Mobile 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 SAP Mobile 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.
Online	Used strictly with Hybrid App applications where access to real-time EIS data is required, bypassing the SAP Mobile Server cache (CDB). See *Online Cache Group Policy*

Policy	Description
EIS managed	The EIS manages data refresh through DCN messages, including managing partitions, user subscriptions to those partitions, and data updates. Selecting this option enables: • **Notify EIS to fetch operation –** (Optional) SAP Mobile Platform invokes a REST operation with a predefined interface when there are operation replay records ready to be processed by the EIS. Upon receipt of the notification the expectation is that the EIS calls back into SAP Mobile Server to retrieve the operation replay records. Select an existing REST connection profile or create a new one. Any associated templates are ignored, and the predefined template is added upon deployment to SAP Mobile Server. • **Disable membership tracking –** (Optional) determines whether or not SAP Mobile Server needs to maintain client-side state in order to clean up a client device in the event a client is unsubscribed from an EIS partition. If enabled and a client is unsubscribed from an EIS partition, the MBO instances that belong to that partition are removed from the device during the next synchronization. If disabled (default) the rows remain on the device after the next synchronization even though the user is no longer subscribed to the EIS partition. If membership tracking is not required, SAP recommends disabling it since the server-side overhead is considerable.

- **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 SAP Mobile 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 42. Cache Group Properties

Property	Description
Common	• **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 SAP Mobile 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 Client Defined Data Partitioning by User and Device Identity

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 SAP Mobile Server cache (CDB) for client defined partitions 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 for client defined partitions.

Table 43. 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.
Relationship	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: • SalesOrderMBO • LineItemMBO • ProductMBO The LineItemMBO has a foreign key reference to the SalesOrderMBO and a foreign key reference to the ProductMBO. 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). The modeler does not specify the ProductMBO as requestor-partitioned since doing so needlessly duplicates reference data in the cache.
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 SAP Mobile 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.

EIS Managed Cache Group Policy

Adding MBOs to a cache group that uses the EIS managed policy indicates that the MBOs are to be updated only by DCN messages that create and manage the EIS defined partitions and user subscriptions.

SAP Mobile Platform behavior and usage for MBOs using the EIS managed group policy includes:

- This option is available to all data sources and MBOs without load operations.
- Only Other operations are allowed. SAP Mobile WorkSpace does not allow modeling of Create, Update, Delete, or Entity Read operations, since they have semantic meaning to SAP Mobile Platform but are meaningless to the EIS system, and since these types of operations are not invoked by SAP Mobile Platform for an EIS defined cache partition they are not allowed.
- You can create only one EIS managed group for each mobile application project.
- **Notify EIS to fetch operation** behavior - when any operation replay records are received for these MBOs by SAP Mobile Server, SAP Mobile Platform Runtime queues them internally to be fetched by the EIS. If there is an endpoint defined for the “EIS Managed” cache, a notification is also sent to the EIS signaling the availability of the records to be fetched. When the endpoint is specified for the “EIS Managed” cache, SAP Mobile Platform Runtime generates a service operation at the time of deployment with the hard coded URI template. The EIS must support a REST Web service with the specified template to successfully receive the notification. For example:

```
Web Method: Get routes associated with specified user
HTTP Action: GET
URI Template:  routes?
cluster={clusterName}&domain={domainName}&package=

{packageName}&username={username}&remoteId={remoteId}&count={repl
ayCount}
Request Representation: n/a
```

```
Response Representation: See routes.xsd element 'routes'.
Response Status Code: 200
```

"EIS Managed" indicates that the EIS initiates the connection to SAP Mobile Server to retrieve updates. When "Notify EIS to Fetch Operation" is selected, SAP Mobile Server sends a notification to the EIS to inform the availability of the updates. However, the EIS is still responsible to fetch the operations from SAP Mobile Server using the HTTP interface.

Note: SAP Mobile Server does not guarantee notifications are delivered in order. It is possible that the "upload" notifications are sent to the EIS out of order if clients are uploading multiple times before the EIS is able to process any of the generated "upload" notifications. It is also possible that the EIS receives the same notification twice due to HTTP usage as the delivery protocol. Duplicate notification is not considered harmful since each operation contains an identifier, but the EIS may want to implement logic that does not react to a notification if it is a duplicate.

When "Notify EIS to Fetch Operation" is not selected, the EIS still fetches the operations from SAP Mobile Server. The difference is that the schedule to fetch is EIS determined and not configured in SAP Mobile Server.

Operation Cache Policies

Setting an operation cache policy for mobile business object (MBO) operations gives you more control of SAP Mobile Server interactions with the enterprise information system (EIS) to which the MBO is bound, and SAP Mobile Server cache (CDB) updates. Fine-tuning these interactions and updates improves both SAP Mobile Server and device application performance.

MBO operations perform specific functions based on their definition:

- Read operation (MBO attributes, load arguments, and synchronization parameters) – the EIS operation used to define and initially populate the CDB (from the EIS) for the MBO. Also called a load operation.
- Create, Update, Delete (CUD operations) – modify EIS data depending on the definition of the operation. SAP Mobile Server maintains a cache (CDB) of back-end EIS data to provide differential synchronization and to minimize EIS interaction.

  While this type of bulk-fetch and CDB caching are effective in reducing the number of interactions required with the EIS, and work well in some other cases (where MBO data is occasionally updated in the EIS), performance suffers if changes are initiated from SAP Mobile Server (by way of MBO operations), or if changes are frequent.

  When an operation is submitted from a device application to the EIS, the cache must be refreshed for those changes to be available to a client the next time the client synchronizes.
- Entity Read operations – refresh a single instance of an MBO composite graph (MBOs in a composite relationship) in the SAP Mobile Server cache (CDB). This type of operation can only be invoked for Create and Update operations that use an "Immediately update the cache" operation cache policy.

Operation cache policies determine how the CDB is updated after an operation executes. The operation cache policy and option combinations from which to choose to associate with MBO CUD operations include:

- Immediately update the cache with Apply merge of operation input/output
- (Legacy) Apply results to the cache
- Immediately update the cache with Apply output of ENTITY READ operation
- Invalidate the cache
- None

When an MBO operation is called, its cache policy determines how operation results are applied to the CDB.

Other mechanisms used to update the CDB that are external to MBO operations, and not associated with operation cache policies include:

1. EIS-initiated DCN – an HTTP request to SAP Mobile Server, in which the DCN request contains information about the changed data, or the changed data itself.
2. Scheduled data refresh – SAP Mobile 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 policy efficiency. Examples include:
   - A poorly designed MBO might have an operation with a cache 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 policy.
   - This same MBO properly designed might have a cache group that refreshes the MBO nightly, increasing SAP Mobile Server performance by deferring load from peak usage hours.

Setting an Operation Cache Policy

Set an operation's cache 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 Policy** tab.
3. Select the cache policy, and option. Cache-affecting operation behavior includes:

   - **Immediately update the cache with Apply merge of operation –** the attributes passed in from the client are combined with the results from the EIS, and merged into the cache using a graph-based merge algorithm. All MBOs in the composite relationship must be mapped to a result.

     The legacy "Apply results to the cache" mode is used if only one result is mapped; otherwise, all MBOs in the composite relationship must be mapped to a result.

Operations with this cache policy and option cannot be combined with the "Invalidate the cache" policy if more than one result is mapped.

- **Immediately update the cache with Apply output of Entity Read operation –** the results from the EIS operation determine the primary key of the entity to be read from the EIS. You can also use client attributes to specify the primary key when the primary key is not contained in the EIS result. The entity is read in its entirety using the specified Entity Read operation. The result is merged into the cache using a graph-based merge algorithm. All MBOs in the composite relationship must be mapped to an entity read result, otherwise, the MBO model is invalid. You must create an Entity Read operation before you can use this cache policy.

  Operations with this cache policy and option cannot be combined with the "Invalidate the cache" policy.
- **Legacy Immediately update the cache (Apply results to the cache) –** if an operation uses the "Immediately update the cache" policy with the "Apply merge of operation input/output" option, and only one result is mapped, the legacy "Apply operations results" behavior is used, including the ability to upsert many MBO instances into the cache if the associated EIS operation returns more than one MBO instance.

  SAP Mobile WorkSpace does not indicate that legacy mode is used, but the API infers this from the fact that only one result is mapped. Legacy "Apply results to the cache" simply upserts rows into the cache. It does not perform a graph-based merge and consequently does not detect deletions.
- **None –** if you unselect all check boxes for Create or Update operations, the operation has no explicit effect on the CDB. Data refresh depends on other mechanisms to update the CDB, for example, the cache group to which the MBO belongs or data change notification (DCN). These are the behaviors for operations which do not have an explicitly modeled cache effect:
  - **None - Create –** an implicit cache effect: a row is created in the cache and marked as Create pending such that it is invisible during client synchronization until the next refresh operation updates the pending row. This behavior ensures that the surrogate key-to-primary key affinity is maintained when the primary key is specified on the client device. If the primary key is not specified on the client device, a logically deleted row with the provided surrogate key is created in the cache.
  - **None - Update –** have no effect on the cache.
  - **None - Delete –** You cannot define a Delete operation in SAP Mobile WorkSpace without a cache effect.
- **Delete Operation –** apply results to the cache using the "Immediately update the cache" cache policy.
- **Invalidate the cache –** restricts invalidation to only those cache partitions affected by the Create or Update operation. Unless otherwise specified, each previously described cache-affecting operation behavior can optionally invalidate the cache. An operation that uses the invalidate cache policy:

- Performs the Create or Update operation, for example, inserts a new record or updates an existing record in the enterprise information system (EIS).
- Unless the client is using asynchronous operation replay, the cache invalidates and refreshes during the same synchronization session which executed the invalidation operation for those MBO partitions affected by the operation (not the entire MBO instance). That is, for operations where both "Apply merge of operation input/output" and "Invalidate the cache" are selected, when the device synchronizes (for example, an update operation) which also requests a download/synchronization, the:
  - Cache is updated
  - Affected partition refreshes

  For operations where both "Apply merge of operation input/output" and "Invalidate the cache" are selected, and a download/synchronization is not requested, the:
  - Cache is updated
  - Affected partition refreshes on the next download/synchronization request from the client

Operation Cache Policy Requirements and Limitations

Common limitations, guidelines, and restrictions for the various cache policies and options.

Cache Policy	Limitation, Guideline, Restriction
Immediately update the cache	• (With Entity Read) Create and Update operations support shared data and updates an object graph within the cache. • (Entity Read and Operation Merge) If non-SAP Mobile Platform applications update the EIS, those changes are not reflected in the cache using this cache policy and the MBO developer instead should use another refresh mechanism to propagate those changes to the cache; for example, a cache group that uses either an On demand, Scheduled, or DCN policy that eventually brings the cache into consistency with the EIS. • (Entity Read and Operation Merge) Supports only one root MBO for a given MBO graph.

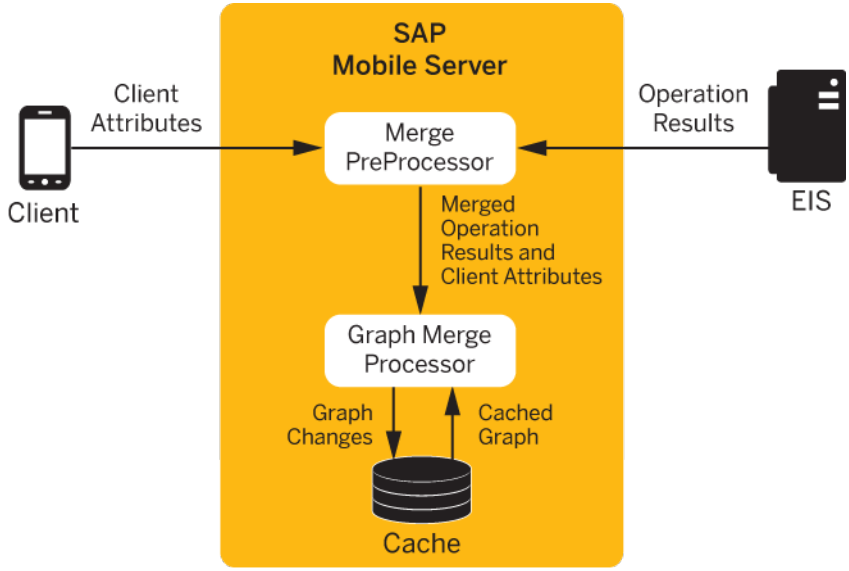
Cache Policy	Limitation, Guideline, Restriction
Legacy "Apply results to the cache"	Modifies the cache based on the returned result set of the called MBO operation: • By default, Create and Update operations use this policy, which can be modified. • Delete operations use this policy, which is always set and cannot be modified. • The MBO must be bound to a data source that has one or more primary-key attributes set. • All columns contain all key attributes. Validation for this cache policy for Web service and SAP MBOs includes: • A warning message if you model a Create operation that does not have a primary key attribute. • A warning message if you model a Create operation if the output record does not contain the primary-key columns. For database MBOs: • SAP Mobile WorkSpace must execute a database MBO operation to retrieve the output records, which can affect the business data on the server to which the MBO is bound if it is an Update, Delete, or Insert operation. • For JDBC data sources, SAP Mobile WorkSpace does not validate whether an operation with this operation cache policy returns primary key fields as part of the output record. If the MBO developer uses this policy option, you must know whether the definition of the operation can return the primary key column or not.

Cache Policy	Limitation, Guideline, Restriction

Invalidate the cache	• The entire cache table is invalidated only if the partitions affected by the operation cannot be determined from the MBO attributes. If the affected partitions can be determined, only the affected partitions are invalidated.
	• For shared data, this policy decouples the client Update operation from the cache refresh. There is a potential granularity mismatch if the Update operation is at a primary key level and the cache refresh is at a partition level; this may result in the refresh of more data than was actually modified.
	• An "On Demand" cache group policy with a zero coherence window has the same outcome as this policy. If there are other non-SAP Mobile Platform applications updating EIS data, the partition refresh pulls those data changes into the cache.
	• Do not use this policy for an operation when the MBO is in a DCN (data change notification) cache group, since the policy has no effect.
	• To achieve optimum performance when using cache invalidating operations, propagate result-affecting load arguments into the MBO attributes, which are defined in the **Attributes > Load Arguments** tab.
	• A partition key identifies the cache partitions affected by the operation. Partition key definition guidelines require that:
	  • 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 mapped only to personalization keys are excluded as partition keys, since personalization keys are available only within a client context and not suitable for this cache policy. 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.
	  • If an MBO belongs to a cache group that uses a DCN policy, the operations of that MBO do not support an "Invalidate the cache" policy.

Merging Operation Input/Output Into the Cache

The MBO developer can configure input mappings so that client provided parameters and personalization keys are passed with Create or Update operations, which use the "Apply merge of operation input/output" cache policy option, to merge them with the operation results and apply them to the SAP Mobile Server cache to update a composite object graph from a single operation.



- **Merge PreProcessor –** accepts client attributes and EIS operation results as input and merges them into a single result according to these rules:
  - Client attributes always contain the surrogate key of the composite root instance, since it is used to identify the graph instance.
  - Client attributes must contain a fully specified (no null attributes) primary key for the pre-processor to merge them with the operation results.
  - If an MBO instance is present in the client attributes but missing from the EIS results, the client-provided instance is used during merge processing. This allows the cache to be populated based solely on client-provided attributes.
  - If an MBO instance or attribute is present in the EIS result, but not in the client attributes, the EIS result is used.

  Essentially the preprocessor treats client attributes as default values that take effect only when the corresponding MBO attribute or MBO instance does not exist in the EIS result. The client attributes contain the client's version of the composite graph nodes influenced by the operation. When a new composite graph is inserted, the set of client attributes contains the complete graph. During an update to a composite graph, only the input attributes associated with the modified nodes and their ancestors are present in the client attributes.

  Note: There is a distinction between a null attribute and a missing attribute, since the preprocessor considers null attribute values to be valid non-primary-key attribute values. If the EIS result contains a null attribute value, the preprocessor does not attempt to find a corresponding default client attribute.
- **Graph Merge Processor –** takes the results from the preprocessing phase (graphId and premerged results) and compares them to the current cache contents. The graph merge

processor processes the composite graph from parent to child comparing cached graph parts to the corresponding EIS graph parts. The graph merge processor compares the cached graph part (the portion of the graph associated with a particular MBO type) with the corresponding premerged results and updates the cache as follows:

- If an MBO instance held in the cache differs from the one held in the pre-merged result, the cache is updated with the version held in the pre-merged results.
- If an MBO instance is contained in the pre-merged result, but not in the cache, the MBO instance is inserted into the cache using the surrogate key contained in the client attributes if a suitable match is found during pre-processing. If a suitable match is not found during pre-processing, a new surrogate key is generated and assigned to the MBO instance before it is inserted into the cache.
- If the MBO instance is contained the cache, but not in the pre-merged results, the row is logically deleted from the cache.
- If an MBO result is missing from the pre-merged results, the graph-part associated with the missing result does not change in the cache.

Note: Apply merge of operation input/output processing applies only to a single composite MBO graph instance. Neither the client attributes nor the EIS results can contain multiple root MBO instances. If a set of client attributes is missing a primary key and cannot be matched with an EIS result during the graph merge processing, the surrogate key is discarded and a warning message is generated. A logically deleted row is not created in the cache.

Object Queries

Object queries are SQL statements associated with a mobile business object (MBO), against the persistent store on the device that returns a subset of a result set. For example, an object query is used to filter data already downloaded from the CDB 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 SAP Mobile Server or a local business object.

Table 44. Object Query Usage

MBO type	Usage
Local business object	The requested data must be available on the mobile device. If not, MBO operations must be called to insert the requested data. The query can then continue to return data from the client's local database.
Bound to a datasource	**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	MBOs that use an Online cache group policy generate a single read-only object query named findByParameter, which is automatically generated by SAP Mobile WorkSpace. findByParameter query parameter(s) are generated for every load argument that has a **Propagate to attribute**. The findByPrimaryKey object query and any other user defined object queries are removed for MBOs that use an Online cache policy. By default, the return type is **Return multiple objects** and **Create an index** is false, and these are the only values you can modify. If you modify a **Propagate To attribute** of a load argument belonging to an MBO using an **Online** cache group policy, the object query is automatically updated.

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. SAP Mobile 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.
- You can remove the findByPrimaryKey query if the native device application (ObjectAPI) does not use it, which can potentially improve device application performance. You can also choose not to create the index even if using the findByPrimaryKey object query, as all index generation is optional.

See *Object Queries* in *Mobile Data Models: Using Mobile Business Objects* for additional information.

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 –** SAP Mobile 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**.

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**. SAP Mobile WorkSpace generates an index based on the query definition.

Packaging and Deploying Mobile Business Objects

Create and deploy a deployment package to the SAP Mobile 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 Mobile Servers, and so on.

See also

- *Developing a Mobile Business Object* on page 69
- *Working with Mobile Business Objects* on page 117
- *Mobile Business Object Overview* on page 70

Deploying a Mobile Application Project

Deploy a Mobile Application project directly to an SAP Mobile 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 SAP Mobile Server portion of the wizard, by dragging a Mobile Application project folder from Workspace Navigator and dropping it on the SAP Mobile 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. Select a deployment mode (**Update**, **Replace**, or **Verify**), target version, Package name, and click **Next**.
3. Select the MBOs from each Synchronization Group to be deployed and click **Next.**

   Note: If any selected MBOs contain errors, the **Next** and **Finish** buttons are disabled.
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, click **Connect**, and select a Domain and Security Configuration for the deployment package and click **Next**. (Optional) If no SAP Mobile Server connection exists, click **Create** and define a connection profile for one to which you can connect and deploy the deployment package.
6. Deploy applications to SAP Mobile Server – select the applications to deploy to SAP Mobile Server. A unique Application ID identifies the application and uses the project name by default.

   SAP Mobile WorkSpace lists not only local applications defined through the mobile application project's context menu **Properties > Mobile Application Project Configuration**, but all applications already assigned to the selected domain of the target server (available applications), whether those existing applications contain this current mobile application project or not. SAP Mobile WorkSpace validates the projects for:
   - If the local and server applications are the same, but the display name or description differ, they display in the target applications list, but a validation error appears because the assigned application ID must be unique.
   - When deploying the project/package with "Replace" mode, if the project/package already exists in an available application that exists on the server, but that application is not selected as the target application, a warning indicates that the server will remove the project/package from the existing application.
   - If a local application is added to the target applications list, and a server application with the same ID but different display name/description is not assigned, a warning indicates that you can modify the display name/description of the existing sever application with that of the local application.
7. Map connection profile to server connections – you must map design-time connection profiles to server-side (runtime) 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, or select **<New Server Connection...>** to create a new server connection.

Contact the system administrator in cases where your development environment permits access to systems that the SAP Mobile Server prohibits.

Note: You can also modify server connection properties (Web service connections only).

8. 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**.
9. (Optional) Specify the name and location for the new deployment profile. This is useful for troubleshooting MBO and deployment errors.
   - 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.
10. Click **Finish** to deploy the project to the SAP Mobile 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 SAP Mobile 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 SAP Mobile Server, and an optional description and click **Next**.

   The value of the Package name field entered here is not the package name on the SAP Mobile 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 SAP Mobile Server. You must deploy your custom classes to the server to use them.
6. Select and assign available applications to be deployed. Each application has an Application ID that uniquely identifies the application and is required for certain application types (for example: Hybrid App, and OData).
7. Review the summary and click **Finish**.

   The mobile deployment package (`fileName.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.
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 45. Mobile Deployment Package Configuration

Screen	Description
General Information	• Package name – identifies the package to be deployed to the SAP Mobile Server. • Description – (optional) a description of the package.

Screen	Description
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 SAP Mobile Server.
Application and Application ID	Each application has an Application ID that uniquely identifies the application and is required for certain application types (for example: Hybrid App and OData).
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 SAP Mobile 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.

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), and click **Next**.
3. Select a target SAP Mobile Server for the package, connect to it, select the domain and security configuration (if not using the default), and click **Next**.
4. Select and assign available applications to be deployed. Each application includes an Application ID that uniquely identifies the application and is required for certain application types (for example: Hybrid App and OData).
5. (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 SAP Mobile 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.
6. Map logical roles to physical roles. Click **Next**.
7. (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.

8. Click **Finish** to deploy the mobile deployment package to the SAP Mobile Server 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 SAP Mobile 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 SAP Mobile 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 46. 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 Mobile 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 SAP Mobile 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 SAP Mobile Servers to which mobile deployment packages are deployed. A deployment profile can contain multiple mobile deployment packages, each with multiple target SAP Mobile Servers as their destination.

Adding a Target Server to a Deployment Package

You can select multiple target SAP Mobile 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 SAP Mobile 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 SAP Mobile 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 SAP Mobile 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 SAP Mobile 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 SAP Mobile Server.

Table 47. Deployment Modes and Target Version

Option	Description
Update	(Default) Updates the target package with an updated version. Use update where existing deployments and client applications need to continue to work. After a successful (or failed) Update deployment, existing client applications previously synchronized with SAP Mobile Server continue to function: existing surrogate keys do not change, all operations are fully functional, and so on. No data loss on the devices occur, and the existing client can synchronize and upload operations as if the Update deployment never occurred.
Replace	Replaces any of the target objects with those in the package. Replace deletes an existing package immediately and replaces it with the newly deployed package with same name and version. Active client connections or EIS connections are terminated. SAP Mobile Server disables the package automatically and re-enables it once it becomes fully functional again. Replace and Update options should not be used for active packages, which may require the administrator to manually disable 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 name	The location in which the deployed unit resides. The default value is the Mobile Application project name.

Supported Changes for Update Deployment

Understand package changes supported by an Update deployment.

Allowed package updates include:

- Adding MBOs to a new synchronization group
- Adding new operations to an MBO
- Changing the Cache Policy
- Adding or removing Result Set Filters
- Adding or removing Result Checkers
- Adding or removing a logical role
- Adding synchronization parameters (excluding list type)
- Adding personalization keys

During deployment, multiple messages may be returned via the ProgressMonitor, which display in the Error Log view in SAP Mobile WorkSpace. Verification does not halt if an error is detected, it continues to log all relevant error messages. When deploying a package in

Update mode where an existing version of that package is deployed, deployment fails verification and returns an error if it "breaks" the existing application.

Detailed messages returned from SAP Mobile Server for an Update deployment indicating what the update consists of includes:

- `virtualTable {MBOName} :removed MBO`
- `MBO: {MBOName} Column {name} :new attribute`
- `MBO: {MBOName} Column {name} :removed attribute`
- `Columns Conflict in MBO {MBOName} different nodes`
- `MBO: {MBOName} Implementations {MBOName} :new Implementation`
- `MBO: {MBOName} Implementations {MBOName} :removed Implementation`
- `Implementations Conflict in MBO {MBOName} different nodes`
- `Service {name} :removed Service`
- `Cache {name} :removed Cache`
- `cachedTable {virtualTableName} :removed cachedTable from {name}`
- `loadGroup {name}:removed loadGroup`

Schema validation messages returned from SAP Mobile Server if an Update deployment fails include:

- `New deployment document fails schema validation.`
- `Virtual table {name} does not contain referenced column {columnRef}.`
- `Virtual table {name} contains an empty or invalid primary key definition.`
- `Verify that the {name} MBO contains attributes suitable for inclusion in a primary key (i.e. non-blob non-clob columns).`
- `Cache-referenced virtual table {referencedTable} does not exist.`
- `queryReference virtual table {referencedTable} does not exist.`
- `Virtual table {virtualTableName} referenced by cache {cache} does not contain column {cachedColumnName}.`
- `The virtual table {tableName} is referenced by more than one cache definition.`
- `The data source {dataSourceName} is not a resource defined on the Mobile server`
- `The connection factory {dataSourceName} is not a resource defined on the Mobile server`
- `The endpoint factory {endpointName} is not a resource defined on the Mobile server`
- `Deploys intersect, not allowed with mode UPDATE`
- `Cannot Merge Documents. Name or version does not match`

Target Server Properties

Select SAP Mobile Server specific domain properties when deploying a project to the server to which you are connected.

Table 48. Target Server Properties

Property	Description
Domain	The domain to which deployment occurs. Separating projects into domains allows you to share various SAP Mobile Server resources between customers while keeping their data separate.
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 SAP Mobile 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 SAP Mobile Server.

When developing mobile applications, you bind the mobile business objects to the data sources available to the SAP Mobile WorkSpace and available from the Enterprise Explorer. When you deploy Mobile Application projects or Mobile Deployment packages to the SAP Mobile Server, you must change data sources from connection profiles to server connections available to the SAP Mobile 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 SAP Mobile 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.

   Note:

   - If you modify an existing design-time connection profile for which there is an existing server connection mapping, you must create a new server connection mapping for the changed design-time connection profile. For example, if you use a connection profile with a port number of 1112 and create a corresponding server connection mapping

during deployment, then modify the connection profile port to 1113, when you redeploy the project you cannot use the existing server connection mapping and instead must create a new one.
- When you deploy a project to SAP Mobile Server and create a server connection, any defined project connection properties that are duplicates of those defined in the server connection properties, are saved into the server connection profile properties.

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 SAP Mobile Server, you can map these logical roles to physical roles that are valid on SAP Mobile 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 SAP Mobile 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 49. 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 SAP Mobile 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 SAP Mobile Server to which you are deploying.
Assigned physical roles	Displays the physical roles that are assigned to the package that you are deploying.
Physical role availability	Modify physical role availability: • 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 SAP Mobile Server available for mapping. • Remove All – makes all physical roles located on the SAP Mobile Server unavailable for mapping. **Note:** Removing physical roles only removes their availability to be mapped to logical roles. It does not remove them from the SAP Mobile Server.

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 SAP Mobile Server.

1. From the Enterprise Explorer, expand the SAP Mobile Servers folder, SAP Mobile 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 SAP Mobile 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 SAP Mobile Server.

1. From the Enterprise Explorer, expand the Mobile 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 SAP Mobile 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 SAP Mobile Server.

1. From the Enterprise Explorer, expand the Mobile 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 SAP Mobile Server. Instead you must either delete the package and redeploy an updated package, or deploy a new package.

Automated Deployment of SAP Mobile WorkSpace Projects

Save a mobile application project's deployment unit to a model file, which can be deployed to SAP Mobile Server using command line utilities.

A model file allows you to deploy a deployment unit from batch files or other processes where deployment outside of SAP Mobile WorkSpace is desired.

Saving the Deployment Unit to a Model File

Save a mobile application project's deployment unit to a model file from SAP Mobile WorkSpace.

Prerequisites

Ensure all referenced data sources are running and that SAP Mobile WorkSpace has access to them.

Task

1. From WorkSpace Navigator, right-click the project and select **Save Deployment Unit to Model File**.

The deployment unit for the project is saved to the model file of the project, named `sup.model`.

2. If there are any changes to the project, rerun **Save Deployment Unit to Model File**, so that the deployment unit is saved to the model file and the project and model file remain in sync.

Extracting the Deployment Unit

Use the AFX_EXTRACTOR utility to extract the deployment unit from the model file.

Syntax

```
java -classpath "AFX_EXTRACTOR_PATH;LOG4J_PATH"
com.sybase.uep.tooling.utilities.AFXExtractor
"MODEL_FILE_PATH" "DIRECTORY_STORED_AFX"
```

Parameters

- **AFX_EXTRACTOR_PATH –** represents the path of the JAR file including the AFXExtractor class, which is installed in the *SMP_HOME*\MobileSDK<Version>\MobileWorkSpace\mobile\eclipse\plugins directory.
- **LOG4J_PATH –** represents the path of the log4j JAR file, which is installed in the *SMP_HOME*\MobileSDK<Version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.tooling.api_<version number>\lib directory.
- **MODEL_FILE_PATH –** represents the path of the model file.
- **DIRECTORY_STORED_AFX –** represents the directory path which contains the deployment unit file.

Examples

- – `java -classpath "C:\<SAP Mobile Platform_InstallDir>\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.tooling_utilities_1.7.0.201104081506.jar;C:\<SAP Mobile Platform_InstallDir>\MobileSDK<version>\MobileWorkSpace\mobile\eclipse\plugins\com.sybase.uep.tooling.api_1.5.5.201012021142\lib\log4j-1.2.15.jar " com.sybase.uep.tooling.utilities.AFXExtractor "c:\sup.model" "c:\afx\result"`

  Errors are logged to the `afx_extractor.log` file located in the directory from which you run the AFX_EXTRACTOR utility. For example, if you run the utility from the default command-line directory, `C:\Documents and Settings\<USERNAME>`,

then the log file is `C:\Documents and Settings\<USERNAME>\afx_extractor.log`.

The Deployment Descriptor File

A deployment configuration can be defined in the deployment descriptor file, including deployment mode, synchronization mode, package name, domain name, named security, endpoint mapping information, and role mapping information.

You can deploy with this descriptor file using the option **-dcf descriptorFile** option. Most of the configuration can be specified in the deployment command. For endpoint mapping, map an existing endpoint for an entity or operation.

Schema

The schema for the descriptior file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:dd="http://www.sybase.com/sup/supadmin"
    xmlns:afx="http://www.sybase.com/sup/afx"
    targetNamespace="http://www.sybase.com/sup/supadmin"
    elementFormDefault="qualified"
attributeFormDefault="unqualified">

    <xs:import namespace="http://www.sybase.com/sup/afx"
               schemaLocation="afx.xsd"/>

    <xs:element name="deployment">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="dd:deploy_mode" minOccurs="1"
                                                      maxOccurs="1"/>
                <xs:element ref="dd:sync_mode" minOccurs="1"
                                                      maxOccurs="1"/>
                <xs:element name="package_name" type="xs:string"
                                                       minOccurs="1"
maxOccurs="1"/>
                <xs:element name="domain_name" type="xs:string"
                                                       minOccurs="1"
maxOccurs="1"/>
                <xs:element name="named_security" type="xs:string"
                                                       minOccurs="1"
maxOccurs="1"/>
               <xs:element ref="dd:endpoint_mapping" minOccurs="0"

maxOccurs="unbounded"/>
                <xs:element ref="dd:role_mapping" minOccurs="0"

maxOccurs="unbounded"></xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="deploy_mode" type="dd:deploy_mode_type"/>
```

```
    <xs:simpleType name="deploy_mode_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="REPLACE"/>
            <xs:enumeration value="UPDATE"/>
            <xs:enumeration value="VERIFY"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="sync_mode" type="dd:sync_mode_type"/>
    <xs:simpleType name="sync_mode_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="REPLICATION"/>
            <xs:enumeration value="MESSAGE"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="endpoint_mapping"
type="dd:endpoint_mapping_type"/>
    <xs:complexType name="endpoint_mapping_type">
        <xs:choice>
            <xs:element ref="dd:entity"/>
            <xs:element ref="dd:operation"/>
        </xs:choice>
    </xs:complexType>

    <!-- named security configuration need support -->
    <xs:element name="role_mapping" type="dd:role_mapping_type"/>
    <xs:complexType name="role_mapping_type">
        <xs:sequence>
            <xs:element name="physical_role" type="xs:string"
                                minOccurs="0" maxOccurs="unbounded"/
>
        </xs:sequence>
        <xs:attribute name="logic_role" type="xs:string"
                          use="required"></xs:attribute>
        <xs:attribute name="mapped_type" type="dd:mapped_type_type"
                          use="required"></xs:attribute>
                </xs:complexType>

    <xs:element name="entity" type="dd:entity_type"/>
    <xs:complexType name="entity_type">
        <xs:sequence>
            <xs:element name="ds_data" type="dd:ds_data_type"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>

    <xs:element name="operation" type="dd:operation_type"/>
    <xs:complexType name="operation_type">
        <xs:sequence>
            <xs:element name="ds_data" type="dd:ds_data_type"/>
        </xs:sequence>
        <xs:attribute name="entity_id" type="xs:string"/>
        <xs:attribute name="entity_name" type="xs:string"/>
        <xs:attribute name="name" type="xs:string"/>
```

```
    </xs:complexType>

    <xs:complexType name="ds_data_type">
        <xs:sequence>
            <xs:element name="data_source_name" type="xs:string" />
            <xs:element name="data_source_type"
                                    type="dd:data_source_type"/>
            <xs:choice minOccurs="0" maxOccurs="1">
                <xs:element ref="afx:connection-profile"
minOccurs="1"
                                                    maxOccurs="1"/>
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <!-- currently, 'connection_property' only
available
                                                                when
'data_source_type' is 'WSSOAP' -->
                    <xs:element ref="dd:connection_property"
minOccurs="1"
                                                maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="data_source_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="JDBC" />
            <xs:enumeration value="DOE" />
            <xs:enumeration value="JCA" />
            <xs:enumeration value="SAP" />
            <xs:enumeration value="WSSOAP" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="mapped_type_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="MAPPED" />
            <xs:enumeration value="NONE" />
            <xs:enumeration value="AUTO" />
        </xs:restriction>
    </xs:simpleType>

    <xs:element name="connection_property">
        <xs:complexType>
            <xs:attribute name="name" type="xs:string"
                                    use="required"></xs:attribute>
            <xs:attribute name="value" type="xs:string"
                                    use="required"></xs:attribute>
            <xs:attribute name="encrypted" type="xs:boolean"
                                    use="optional"></xs:attribute>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Example

```
<sup:deployment xmlns:sup="http://www.sybase.com/sup/supadmin">
<sup:deploy_mode>UPDATE</sup:deploy_mode>
<sup:sync_mode>MESSAGE</sup:sync_mode>
<sup:package_name>test:1.0</sup:package_name>
<sup:domain_name>default</sup:domain_name>
<sup:named_security>admin</sup:named_security>
<!—This endpoint mapping set ws as the endpoint for entity
ConversionRate-->
<sup:endpoint_mapping>
<sup:entity id="_uq_hYGAlEeCo7Mo8NbZ9vw" name="ConversionRate">
<sup:ds_data>
<sup:data_source_name>ws</sup:data_source_name>
<sup:data_source_type>WSSOAP</sup:data_source_type>
</sup:ds_data>
        </sup:entity>
</sup:endpoint_mapping>
<!—This endpoint mapping set sampledb as the endpoint for entity
Customer-->
<sup:endpoint_mapping>
        <sup:entity id="_lLL_EGAlEeCo7Mo8NbZ9vw" name="Customer">
<sup:ds_data>
<sup:data_source_name>sampledb</sup:data_source_name>
<sup:data_source_type>JDBC</sup:data_source_type>
</sup:ds_data>
       </sup:entity>
</sup:endpoint_mapping>
<!—Following endpoint mapping set sampledb as the endpoint for the
operations-->
<sup:endpoint_mapping>
         <sup:operation entity_id="_lLL_EGAlEeCo7Mo8NbZ9vw"
entity_name="Customer" name="create">
             <sup:ds_data>
                 <sup:data_source_name>sampledb</
sup:data_source_name>
                 <sup:data_source_type>JDBC</sup:data_source_type>
             </sup:ds_data>
         </sup:operation>
</sup:endpoint_mapping>
<sup:endpoint_mapping>
       <sup:operation entity_id="_lLL_EGAlEeCo7Mo8NbZ9vw"
entity_name="Customer" name="update">
           <sup:ds_data>
               <sup:data_source_name>sampledb</sup:data_source_name>
                 <sup:data_source_type>JDBC</sup:data_source_type>
           </sup:ds_data>
        </sup:operation>
</sup:endpoint_mapping>
<sup:endpoint_mapping>
      <sup:operation entity_id="_lLL_EGAlEeCo7Mo8NbZ9vw"
entity_name="Customer" name="delete">
          <sup:ds_data>
               <sup:data_source_name>sampledb</sup:data_source_name>
                <sup:data_source_type>JDBC</sup:data_source_type>
```

```
          </sup:ds_data>
      </sup:operation>
</sup:endpoint_mapping>
<sup:role_mapping logic_role="Role3" mapped_type="MAPPED">
<sup:physical_role>SUP Administrator</sup:physical_role>
</sup:role_mapping>
<sup:role_mapping logic_role="Role1" mapped_type="NONE">
<sup:physical_role>None</sup:physical_role>
</sup:role_mapping>
</sup:deployment>
```

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 and Android 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 Guide: <Device Platform> Object API Applications* for your platform (BlackBerry, Windows Mobile, or iOS) for more information.
- Use the Hybrid App Forms Editor in Eclipse to generate the code for a Hybrid App package.

  See *Developer Guide: Hybrid Apps* 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. You must have an active connection to the data sources to which the MBOs are bound.
- Ensure the enterprise information system (EIS) data sources referenced by the MBOs are running and available or code generation fails.

- Install either the `net_rim_api.jar` file (BlackBerry) or the `android.jar` file (Android) before generating code for these platforms.

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*. The most recent configuration is used if the code generation wizard is invoked from the Basic developer profile.

Option	Description
Code generation configuration	A table lists all existing named configurations plus the most recently used configuration. You can select any of these, click **Next**, and proceed. Additionally, you can: • Create new configuration – click **Add** and enter the **Name** and optional **Description** of the new configuration and click **OK** to save the configuration for future sessions. You can also select **Copy from** to copy an existing configuration which can then be modified. • Most recent configuration – if you click **Next** the first time you generate code without creating a configuration, the configuration is saved and displays as the chosen configuration the next time you invoke the code generation wizard. If the most recent configuration used is a named configuration, it is saved as the first item in the configuration table, and also "Most recent configuration", even though it is still listed as the original named 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.

SAP Mobile WorkSpace automatically computes the default 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.

5. Click **Next**.
6. Enter the information for these code generation options:

Option	Description
Language	Choose the language used for developing the client applications: • 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.
	• Java
	  • Java ME for BlackBerry
	  • Android
	• C#
	  • .NET Framework for Windows
	  • .NET Compact Framework 3.5 for Windows Mobile
	• Objective C
	  • iOS

SAP Mobile Server	Specify a default SAP Mobile Server connection profile to which the generated code connects at runtime.
Server domain	Choose the domain to which the generated code will connect. If you specified an SAP Mobile Server to which you previously connected successfully, the first domain in the list is chosen by default. You can enter a different domain manually. **Note:** This field is only enabled when an SAP Mobile Server is selected.

Option	Description
Page size	(Optional) Select the page size for the generated client code. The page size field is filled either with the recommended values if this is the very first invocation (based on the selected MBO's attributes), or the value saved from the previous code generation invocation. In cases where the page size is smaller then the recommended size, the code generation wizard shows the warning: `Page size is smaller than the maximum business object size, which may cause synchronization failure` and you must chose a larger page size to avoid a runtime errror. 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. **Note:** This field is only enabled when an SAP Mobile Server is selected.
Package, Namespace, or Name Prefix	• Package – enter a package name for Java. The package name must follow Java naming conventions for packages. For example, no leading or trailing spaces and no special characters such as §&/, except that the first letter may be upper-case. • Namespace – enter a namespace for C#. • Name Prefix – enter a name prefix for Objective C.

Option	Description
Destination	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. **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.
Third-party jar file	Enter or browse to the location of the third party jar file. For example, `net_rim_api.jar` for BlackBerry, or `android.jar` for Android. If you select Java as the language, and if the BlackBerry or Android third-party JAR file has not been added, the warning `The dependent third-party class 'net.rim.device.api.system.ApplicationDescriptor' cannot be found` or `The dependent third-party class 'android.content.Context' cannot be found` displays.

7. Select **Generate metadata classes** to generate metadata for the attributes and operations of each generated client object.
8. Select **Including 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 **Including object manager classes** option is enabled only for BlackBerry and C# if you select **Generate metadata classes**. 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.

Note: When generating code for Android or iOS, "Generate metadata classes" is automatically selected and cannot be unselected. The "Including object manager classes" option is unavailable and unsupported.

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 or android.jar File

Depending on your Java development environment, install either the `net_rim_api.jar` file (BlackBerry) or the `android.jar` file (Android) to avoid code generation errors when generating Java code for native application development.

After generating Java code for BlackBerry or Android, 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 `<MobileApplicationProjectName>DB.java` file, and appear because of dependencies on the `net_rim_api.jar` or `android.jar` file if not in the project build path. You can avoid these errors by following this procedure for any projects for which you generate Java code:

1. Install the BlackBerry or Android development environment.
2. During code generation, set **Third-party jar file** to the appropriate JAR file. For example, for BlackBerry locate `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`.

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/*.

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.

See also

- *Opening a Perspective* on page 239
- *Opening a View* on page 243
- *Introduction to SAP Help* on page 253

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 240
- *Rearranging Views in a Perspective* on page 241
- *Moving the Perspective Shortcut Bar* on page 242
- *Resetting an Active Perspective to its Default Appearance* on page 242
- *Resetting an Inactive Perspective to its Default Appearance* on page 243

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 240
- *Rearranging Views in a Perspective* on page 241
- *Moving the Perspective Shortcut Bar* on page 242
- *Resetting an Active Perspective to its Default Appearance* on page 242
- *Resetting an Inactive Perspective to its Default Appearance* on page 243

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 50. 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.

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

- *Perspectives* on page 240
- *Perspective Shortcut Bar* on page 240
- *Moving the Perspective Shortcut Bar* on page 242

- *Resetting an Active Perspective to its Default Appearance* on page 242
- *Resetting an Inactive Perspective to its Default Appearance* on page 243

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.

1. Right-click in the Perspective shortcut bar to open its context menu.
2. 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.
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 240
- *Perspective Shortcut Bar* on page 240
- *Rearranging Views in a Perspective* on page 241
- *Resetting an Active Perspective to its Default Appearance* on page 242
- *Resetting an Inactive Perspective to its Default Appearance* on page 243

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 240
- *Perspective Shortcut Bar* on page 240
- *Rearranging Views in a Perspective* on page 241
- *Moving the Perspective Shortcut Bar* on page 242
- *Resetting an Inactive Perspective to its Default Appearance* on page 243

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 240
- *Perspective Shortcut Bar* on page 240
- *Rearranging Views in a Perspective* on page 241
- *Moving the Perspective Shortcut Bar* on page 242
- *Resetting an Active Perspective to its Default Appearance* on page 242

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 245
- *Opening a Perspective* on page 239
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249

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 244
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249
- *Editors* on page 249
- *Resources* on page 250

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 243
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249

- *Editors* on page 249
- *Resources* on page 250

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 243
- *Detaching a View* on page 244
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249
- *Editors* on page 249
- *Resources* on page 250

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 243
- *Detaching a View* on page 244
- *Floating a View* on page 245
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249
- *Editors* on page 249
- *Resources* on page 250
- *Opening a Perspective* on page 239

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 243
- *Detaching a View* on page 244
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *Enterprise Explorer* on page 249
- *Editors* on page 249
- *Resources* on page 250
- *Opening a Perspective* on page 239

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 > editorName**.
   - 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 243
- *Detaching a View* on page 244
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247
- *Editors* on page 249
- *Resources* on page 250
- *Opening a Perspective* on page 239

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 243
- *Detaching a View* on page 244
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247
- *Enterprise Explorer* on page 249
- *Resources* on page 250

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 243
- *Detaching a View* on page 244
- *Floating a View* on page 245
- *Creating a Fast View* on page 245
- *WorkSpace Navigator* on page 247

- *Enterprise Explorer* on page 249
- *Editors* on page 249

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, models, schemas, or zip files.

When you import a resource, a copy of the resource is made and brought into SAP Mobile 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 51. File System

Field	Description
From directory	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 SAP 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

- *Projects* on page 239
- *Opening a Perspective* on page 239
- *Opening a View* on page 243

Help Features

Familiarize yourself with online help features.

Table 52. Online Help Features

Feature	Description
Preferences	Using the Preferences dialog box, specify how you want help to display: • 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.

Feature	Description
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.
Printing a single help topic	Print selected topics by clicking the **Print Page** icon in the Help Contents toolbar (right pane). You can also select the **Print Topics** icon in the Table of Contents toolbar (left pane) and select **Print Selected Topic**.
Printing a group or collection of topics	Print a group or collection or topics by clicking the **Print Topics** icon in the table of Contents toolbar (left pane) and select **Print Selected Topic** and **All Subtopics**. If you have PDF capabilities, you can choose to print the group of topics to the PDF-designated printer to create a PDF.
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.

Feature	Description
Breadcrumbs	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. To return to your original topic, click the **Back** arrow in the tool bar.
Tutorials, Samples, and Quick Start Videos	You can access information by: • 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. **Note:** Tutorials, Samples, or Quick Start videos may not be available.

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:

- SAP 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 53. 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 SAP Mobile Platform Performance Issues* for information about troubleshooting issues with the Eclipse - based user interface or other SAP Mobile Platform components.

Index

O

P

R

S

T

W

X