SYBASE®
An SAP® Company

Developer Guide: Migrating to Sybase Mobile SDK 2.2 SP02

Sybase Unwired Platform 2.2 SP02

DOCUMENT ID: DC01857-01-0222-01
LAST REVISED: January 2013

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

Contents

Migrate Your Artifacts

(Audience: application developers) Migrate your applications to the latest version of Sybase® Unwired Platform 2.2 to take advantage of new features.

The upgrade to Sybase Unwired Platform 2.2 is performed in place, which means you can continue to run 2.1 ESD #3 applications without migrating them. However, you might need to perform some migration tasks to take advantage of new features and system improvements, or if you update the application as part of its life cycle.

Note: References to 2.2 include support packages; specific support packages are identified only if there is a change significant to a particular support package. Sybase recommends you always install the latest support package available.

After you install and upgrade your Unwired Server instances, migrate your mobile business objects (MBOs), projects, and applications as needed. These instructions are for migrating from Unwired Platform 2.1 ESD #3 to 2.2.

If you upgraded from a version earlier than 2.1 ESD #3 (including 2.1, 2.1 ESD #1, and 2.1 ESD #2), refer to *Release Bulletin 2.1 ESD #3*, and its updates, for additional application migration information: *http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00835.0213/doc/html/title.html*.

For supporting information, see:

- *New Features*
- *Supported Hardware and Software*

Migrate Mobile Business Objects

No steps are required to migrate 2.1 ESD #3 mobile business objects (MBOs) to version 2.2 SP02; however, you may need to perform some migration steps to take advantage of new features.

If you are migrating from a version earlier than 2.1 ESD #3, see *Release Bulletin 2.1 ESD #3* on Product Documentation, the *Mobile Business Object* section: *http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00835.0213/doc/html/jwo1333684770287.html*.

Migrate Mobile Business Objects to 2.2 SP02

- In versions earler than Unwired Platform version 2.2 SP02, Sybase Unwired WorkSpace allowed mapping of operations with multiple MBO arguments ('Filled from Attribute',

client parameter, and personalization key) at the same time, even though it might not work properly on the device application during runtime.

With version 2.2 SP02, when adding a mapping of an operation argument, Sybase Unwired WorkSpace now allows only one of the three sources (MBO attribute, client parameter, personalization key) to map into the operation argument at one time; that is, the argument value sources are mutually exclusive.

However, when migrating the Mobile Application project from earlier versions, Sybase Unwired WorkSpace preserves the original MBO operation argument value assignment choices the developer made, to retain backward compatibility with the project in the earlier version. Sybase Unwired WorkSpace does not remove any mappings when migrating a project.

In a migrated project, if an operation argument is mapped to a client parameter as well as an attribute or personalization key, this warning appears:

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

The developer should adjust the MBO model so that an operation argument maps to only one source.

Note: The developer can provide a default value for the operation argument, regardless of how the argument is mapped.

Migrate Object API Applications

No steps are required to migrate 2.1 ESD #3 applications to version 2.2 SP02; however, you may need to perform some migration steps to take advantage of new features.

If you are migrating from a version earlier than 2.1 ESD #3, see *Release Bulletin 2.1 ESD #3* on Product Documentation, the *Migrating Object API Applications* section for your platform: *http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00835.0213/doc/html/jwo1335475866858.html*.

Native Client Version Compatibility Matrix

Compatibility between versions of the client object API and Unwired Server.

Native Client Object API and Unwired Server Version Compatibility

	Unwired Server 2.1	Unwired Server 2.1 ESD #1	Unwired Server 2.1 ESD #2	Unwired Server 2.1 ESD #3	Unwired Server 2.2 SP02
Native Client Object API 2.1	Yes	Yes	Yes	Yes	Yes
Native Client Object API 2.1 ESD #1	No	Yes	Yes	Yes	Yes
Native Client Object API 2.1 ESD #2	No	No	Yes	Yes	Yes
Native Client Object API 2.1 ESD #3	No	No	No	Yes	Yes
Native Client Object API 2.2 SP02	No	No	No	No	Yes

Android

No migration changes are required for Android applications; however you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate Object API from 2.1 ESD #3 to 2.2

Afaria library changes require you to modify and recompile your applications.

1. Access the Android Afaria client library and JAR files that are available in: `SUP_HOME\MobileSDK22\ObjectAPI\Android`

   Note: Alternatively, navigate to the Mobile Enterprise Technical Support website at *http://frontline.sybase.com/support/downloads.aspx* (registration required).

Download the appropriate Android Afaria client (see *Supported Hardware and Software*).

2. Import the Android Afaria client using information in *Developer Guide: Android Object API Applications*. See *Importing Libraries and Code* (in either the *Development Task Flow for Object API Applications* section, or the *Development Task Flow for DOE-based Object API Applications* section as appropriate).

BlackBerry

No migration changes are required for BlackBerry Object API applications; however, you may need to perform some migration steps to take advantage of new features.

Migrate Object API to 2.2

- **Client library changes –** for BlackBerry:
  - The `sup_client2.jar` client is now shipped as a library, with no separate `sup_client2.cod` and `sup_client2.alx` files. This requires a change to how you develop BlackBerry projects:
    - **Eclipse projects –** export `sup_client2.jar` into the build path configuration.
    - **BlackBerry JDE projects –** create a library project including `sup_client2.jar`.
  - Several client files have been deleted in version 2.2 SP02: `CommonClientLib`, `MessagingClientLib`, `MocaClientLib files`, and `MCL.jar` substitutes. However, `MCL.jar` packages and classes are shipped into `sup_client2.jar`, so change your application to reference `sup_client2.jar` and `UltraliteJ12.jar`

  For information and examples for migrating existing BlackBerry applications to 2.2 SP02 implementing these changes, see *Migrating BlackBerry Applications (Eclipse Project)* on page 5 and *Migrating BlackBerry Applications (JDE Project)* on page 5.
- **API changes –** a new `setApplicationIdentifier(String value, String signerId)` API is available to replace the old signing implementation. It is based on BlackBerry Password Based Code Signing Authority.

  To learn more about the BlackBerry Password Based Code Signing Authority on which the API is based, and about the parameter `signerId`: *http://supportforums.blackberry.com/t5/Java-Development/Protect-persistent-objects-from-access-by-unauthorized/ta-p/524282*.

  To download the BlackBerry signing tool used with this new API: *https://swdownloads.blackberry.com/Downloads/entry.do?code=D82118376DF344B0010F53909B961DB3*.

For information and examples for migrating existing BlackBerry applications to 2.2 SP02 implementing this change, see *Migrating BlackBerry Applications (Eclipse Project)* on page 5 and *Migrating BlackBerry Applications (JDE Project)* on page 5.

Note: With this change, the `setApplicationIdentifier(String value)` API is deprecated and will be removed in a future release.

Migrating BlackBerry Applications (Eclipse Project)

Migrate BlackBerry Object API applications from 2.1 ESD #3 to version 2.2 using an Eclipse project.

These steps use an example that demonstrates the new BlackBerry signing API method.

To learn more about the BlackBerry Password Based Code Signing Authority on which the API is based, and about the parameter `Signer Id`: *http://supportforums.blackberry.com/t5/Java-Development/Protect-persistent-objects-from-access-by-unauthorized/ta-p/524282*

1. Download the BlackBerry signer tool, and install it in your development environment: *https://swdownloads.blackberry.com/Downloads/entry.do?code=D82118376DF344B0010F53909B961DB3*.
2. After installing the signer tool, generate a new key file (for example: `suptest.key`).
3. Create the BlackBerry project in Eclipse:
   a) Navigate to **Configure Build Path > Libraries** tab, and reference:
      - `sup_client2.jar`
      - `UltraliteJ12.jar`
   b) Navigate to the **Order and Export** tab, and check to make sure the `sup_client2.jar` file is included in your application JAR file.
4. Copy the generated key file (for example, `suptest.key`) to the project `src` folder.
5. In your application source code, set the new key file (`suptest` in this example):

   ```
   com.sybase.mobile.Application.getInstance().setApplication
   Identifier(end2end.test.Const.ApplicationIdentifier,
   "suptest");
   ```

6. Build your project, and run the application on a simulator to test it.
7. When you are ready to run the application on a real device, sign the .cod files using the signature tool (**BlackBerry > Sign**). After you sign the .cod files with the BlackBerry signature tool, use the File Signer that you installed in step 1 to sign the .cod file again.
8. Install the cod files on the device using provisioning procedures, and run the application.

Migrating BlackBerry Applications (JDE Project)

Migrate BlackBerry Object API applications from 2.1 ESD #3 to version 2.2 using a BlackBerry JDE project.

These steps use an example that demonstrates the new BlackBerry signing API method.

To learn more about the BlackBerry Password Based Code Signing Authority on which the API is based, and about the parameter `Signer Id`: *http://supportforums.blackberry.com/t5/Java-Development/Protect-persistent-objects-from-access-by-unauthorized/ta-p/524282*

1. Download the BlackBerry signer tool, and install it in your development environment: *https://swdownloads.blackberry.com/Downloads/entry.do?code=D82118376DF344B0010F53909B961DB3*.
2. After installing the signer tool, generate a new key file (for example: `suptest.key`).
3. Create a BlackBerry library project in the IDE, add `sup_client2.jar` to the project, and then build it.
4. Create an empty BlackBerry project in the IDE:
   a) Navigate to **Configure Build Path**, and import JAR files:
      - `UltraliteJ12.jar`
      - `ULjDatabaseTransfer.jar`
   b) Navigate to the **Project Dependencies** tab, and check the library project.
5. Copy the generated key file (for example, `suptest.key` to the project root folder.
6. In your application source code, set the new key file (`suptest` in this example):
   ```
   com.sybase.mobile.Application.getInstance().setApplication
   Identifier(end2end.test.Const.ApplicationIdentifier,
   "suptest");
   ```
7. Build your project, and run the application on a simulator to test it.
8. When you are ready to run the application on a real device, sign the .cod files using the signature tool. After you sign the .cod files with the BlackBerry signature tool, use the File Signer that you installed in step 1 to sign the .cod file again.
9. Install the cod files on the device using provisioning procedures, and run the application.

iOS

No migration changes are required for iOS Object API applications.

Windows and Windows Mobile

No migration changes are required for Windows and Windows Mobile Object API applications; however, you may need to perform some migration steps to take advantage of new features.

Migrate Object API to 2.2

A client library name change requires you to modify and recompile your Windows Mobile and Win32 applications. The version number is appended to the file name: `CMessagingClient.dll` has been renamed to `CMessagingClient2.2.2.dll`.

Object API Changes in SDK Version 2.2 SP02

There are several changes in the Object API for SDK 2.2 SP02.

Write to the Database During Synchronization

Connection API changes enable you to write to the database during synchronization.

Table 1. New Connection Method

Method	Description
`allowConcurrentWrite`	Sets the property to true to allow multiple concurrent writer threads.

Documented in: *Developer Guide: <Device Platform> Object API Applications*, see *Setting Up the Connection Profile*

Synchronization Parameter Enhancements

Subscription API changes let you more easily read or update synchronization parameter names or values for all active synchronization parameter sets; and add synchronization parameters before performing an initial synchronization.

Table 2. New Subscription Methods

Method	Description
`GenericList([mbo]Subscription > GetSubscriptions()`	Returns all subscription information. For list type synchronization parameters, the MBO class should use this method to get all default subscriptions.
`AddSubscription([mbo]Subscrip-tion subscription)`	Adds a subscription.
`RemoveSubscription([mbo]Sub-scription subscription)`	Removes a subscription.

Documented in: *Developer Guide: <Device Platform> Object API Applications*, see *Managing Synchronization Parameters*

Native Push Notification

Use the new `addPushNotificationListener` API to register the push notification listener object, and implement a new PushNotificationListener protocol definition to receive push notifications. When a native push notification is received, the listener's `onPushNotification` method is invoked.

Table 3. New Push Notification Method

Method	Description
`addPushNotificationListener`	Enables push notification.

Documented in: *Developer Guide: <Device Platform> Object API Applications*, see:

- *Native Notification APIs*
- *Callback and Listener APIs*

KPI Tracking

A new Object API client interface enables iOS, Android, and BlackBerry to access performance libraries for tracing or collecting key performance indicators (KPIs).

Table 4. New PerformanceAgentService Methods

Method	Description
`startInteraction()`	The service starts collecting metrics.
`stopInteraction()`	Metrics collection stops and a summary is sent to a reporting target.

Documented in: *Developer Guide: <Device Platform> Object API Applications* (Android, BlackBerry and iOS), see *Tracking KPI*

End to End Trace

New API enables you to provide code in client applications so end users can enable trace from the device to the enterprise information system (EIS).

Table 5. New End to End Trace Classes

Class	Platform
• `com.sybase.mobile.util.e2etrace.E2ETraceService` • `com.sybase.mobile.util.e2etrace.E2ETraceLevel` • `com.sybase.mobile.util.e2etrace.impl.E2ETraceServiceImpl` • `com.sybase.mobile.util.e2etrace.impl.E2ETraceMessage`	Android
• `SUPE2ETraceService` • `SUPE2ETraceLevel` • `SUPE2ETraceServiceImpl` • `SUPE2ETraceMessage`	iOS

Documented in: *Developer Guide: <Device Platform> Object API Applications* (Android and iOS), see *End to End Trace*

Customization Resource Bundles

A new Application API method is available for customization resource bundles.

Table 6. New Application API Method

Method	Platform
`beginDownloadCustomizationBundle()`	Android, BlackBerry, iOS, Windows, Windows Mobile

Documented in: *Developer Guide: <Device Platform> Object API Applications*, see *beginDownloadCustomizationBundle()*

New BlackBerry Application Signing API

A new BlackBerry application signing API is available for `com.sybase.mobile.Application.getInstance().setApplicationIdentifier(String value)`, which is based on the BlackBerry Password Based Code Signing Authority.

Table 7. New BlackBerry Application Signing API

Method	Description
`setApplicationIdentifier(String value, signerId)`	Identifies the signed key file used by the application. The key file is used with the BlackBerry Password Based Code Signing Authority.

Table 8. Deprecated BlackBerry API

Method	Description
`setApplicationIdentifier(String value)`	This method set will be removed in a future release.

Documented in: *Developer Guide: BlackBerry Object API Applications*, see *Signing*

Migrate Hybrid Web Container Projects

No steps are required to migrate 2.1 ESD #3 Hybrid Web Container projects to version 2.2; however, you may need to perform some migration steps to take advantage of new features.

If you are migrating from a version earlier than 2.1 ESD #3, see *Release Bulletin 2.1 ESD #3* on Product Documentation, the *Migrating Mobile Workflow Projects* section for your platform: *http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc00835.0213/doc/html/vhu1317418586580.html*.

Note: Prior to 2.2, Hybrid Web Container was known as Mobile Workflow.

Hybrid Web Container Version Compatibility Matrix

Compatibility between versions of the Hybrid Web Container and Unwired Server, and Hybrid Web Container and Hybrid App applications.

Hybrid Web Container and Unwired Server/ Compatibility

Client/ Hybrid Web Container	Unwired Server 2.1	Unwired Server 2.1 ESD #2	Unwired Server 2.1 ESD #3	Unwired Server 2.2 SP02
Hybrid Web Container 2.1	Yes	Yes	Yes	Yes

Client/ Hybrid Web Container	Unwired Server 2.1	Unwired Server 2.1 ESD #2	Unwired Server 2.1 ESD #3	Unwired Server 2.2 SP02
Hybrid Web Container 2.1 ESD #2	No	Yes	Yes	Yes
Hybrid Web Container 2.1 ESD #3	No	Yes	Yes	Yes
Hybrid Web Container 2.2 SP02	No	Yes	Yes	Yes

Note: There was no 2.1 ESD #1 Hybrid Web Container. 2.1 ESD #1 shipped with 2.1 Mobile Workflow clients.

Hybrid Web Container and Hybrid App Compatibility

Client/ Hybrid Web Container	Hybrid App 2.1	Hybrid App 2.1 ESD #2	Hybrid App 2.1 ESD #3	Hybrid App 2.2 SP02
Hybrid Web Container 2.1	Yes	No	No	No
Hybrid Web Container 2.1 ESD #2	Yes	Yes	No	No
Hybrid Web Container 2.1 ESD #3	Yes	Yes	Yes	No
Hybrid Web Container 2.2 SP02	Yes	Yes	Yes	Yes

Note: * There was no 2.1 ESD #1 Hybrid Web Container. 2.1 ESD #1 shipped with 2.1 Mobile Workflow clients.

Migrate Hybrid Web Containers

Migration changes may be required to take advantage of new Hybrid Web Container features.

Migrate Hybrid Web Container from 2.1 ESD #3 to 2.2

- Sybase Unwired Platform 2.2 Hybrid Web Container embeds the latest Cordova 2.0 library (previously known as PhoneGap). Any 2.1 ESD #3 Hybrid App that uses PhoneGap 1.4.1 is incompatible with 2.2 Hybrid Web Container unless you upgrade the Hybrid App.

Note: Sybase Unwired Platform 2.1 ESD #3 shipped the Android and iOS Hybrid Web Container with the PhoneGap 1.4.1 library embedded. After PhoneGap 1.4.1, backward compatibility was broken, and the name changed to Cordova.

- Update any native PhoneGap plug-in to use the new Cordova interface names.
- BlackBerry Hybrid Web Container has been changed to a standalone application. The 2.2 BlackBerry Hybrid Web Container can coexist only with the 2.1 ESD #3 version. Coexist means the application is not upgraded; instead, multiple versions of the applications exist.
- Windows Mobile Hybrid Web Container has been changed to a standalone application. The 2.2 Windows Mobile Hybrid Web Container can coexist only with the 2.1 ESD #3 version (no upgrade).

Migrate Hybrid Apps

Migration changes may be required to take advantage of new Hybrid App features.

Migrate Hybrid Apps from 2.1 ESD #3 to 2.2

A Hybrid App developed in an earlier release (such as 2.0, 2.1, 2.1 ESD #2, and 2.1 ESD #3) should be compatible in a 2.2 Hybrid Web Container unless it uses PhoneGap 1.4.1 functionality (PhoneGap broke backward compatibility with its release of the new and renamed Cordova 2.0 library).

To use new 2.2 Hybrid Web Container functionality, manually reconstruct the Hybrid App to use the new JavaScript API files shipped in the folder: *SUP_HOME* `\MobileSDK22\HybridApp\API\Container`.

The Designer in Sybase Unwired Platform 2.2 SP02 generates code using the new JavaScript API. Earlier versions did not use the new JavaScript API to generate code, and generated 2.1 ESD #3 compatible Hybrid Apps instead.

Additional notes for in-place upgrades:

- The 2.2 Android Hybrid Web Container is an in-place upgrade from 2.1 ESD #3, if you built it from a template source keeping the same “Application id.” In other words, Hybrid Apps that are already deployed remain intact on the device after the upgrade, and the old binaries are replaced with the 2.2 Android Hybrid Web Container binaries. See the Android matrix in *Hybrid Web Container Migration Paths for Android* on page 14.
- The 2.2 BlackBerry Hybrid Web Container is not upgraded but coexists with other versions. See the BlackBerry matrix in *Hybrid Web Container Migration Paths for BlackBerry* on page 15.
- The 2.2 iOS Hybrid Web Container is an in-place upgrade from any earlier version, if you built it from template source keeping the same “bundle id.”. In other words, Hybrid Apps that are already deployed remain intact on the device after the upgrade, and the old binaries are replaced with the 2.2 iOS Hybrid Web Container binaries. See the iOS matrix in *Hybrid*

Web Container Migration Paths for iOS on page 16 (the Applications Build from Source Code section).

- The 2.1 ESD #3 to 2.2 iOS Hybrid Web Container from the Apple App Store is an in-place upgrade. See the iOS matrix in *Hybrid Web Container Migration Paths for iOS* on page 16 (Applications Downloaded from Apple's App Store).
- The 2.2 Windows Mobile Hybrid Web Container coexists. See the Windows Mobile matrix in *Hybrid Web Container Migration Paths for Windows Mobile* on page 17.

`HttpAuthDCNServlet` no longer uses the "admin" security profile to authenticate users when an application user name does not include a security configuration name. Instead, if the user name includes a security configuration, `HttpAuthDCNServlet` uses it. For Workflow DCN, where the user name may not include a security profile, `HttpAuthDCNServlet` uses the value of the requested "security parameter" to authenticate the user. This should not affect your Hybrid Apps, but is useful to know if you are developing Hybrid Apps that take advantage of DCN updates.

See also

- *Hybrid Web Container Migration Paths for Android* on page 14
- *Hybrid Web Container Migration Paths for BlackBerry* on page 15
- *Hybrid Web Container Migration Paths for iOS* on page 16
- *Hybrid Web Container Migration Paths for Windows Mobile* on page 17

Android

No migration changes are required for Android Hybrid Apps; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate Hybrid Web Container from 2.1 ESD #3 to 2.2

Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.

Hybrid Web Container Migration Paths for Android

Supported Hybrid Web Container (HWC) migration paths on Android.

Table 9. Android Migration Paths

	2.1 HWC	2.1 ESD #2 HWC	2.1 ESD #3 HWC	2.2 SP02 HWC
2.1 HWC	N/A	In-place upgrade	Coexist	Coexist
2.1 ESD #2 HWC	N/A	N/A	Coexist	Coexist
2.1 ESD #3 HWC	N/A	N/A	N/A	In-place upgrade
2.2 SP02 HWC	N/A	N/A	N/A	N/A

Note: *There was no 2.0 or 2.1 ESD #1 Android Hybrid Web Container.

- N/A – not applicable.
- Coexist – the application is not upgraded; multiple versions of the application can coexist.
- In-place upgrade – the application is upgraded to the new version (you must modify the application to add new features).

See also

- *Migrate Hybrid Apps* on page 12

BlackBerry

No migration changes are required for BlackBerry Hybrid Apps; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate Hybrid Web Container from 2.1 ESD #3 to 2.2

Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.

Hybrid Web Container Migration Paths for BlackBerry

Supported Hybrid Web Container (HWC) migration paths on BlackBerry.

Table 10. BlackBerry Migration Paths

	2.1 HWC	2.1 ESD #2 HWC	2.1 ESD #3 HWC	2.2 SP02 HWC
2.1 HWC	N/A	In-place upgrade	In-place upgrade	Coexist
2.1 ESD #2 HWC	N/A	N/A	In-place upgrade	Coexist
2.1 ESD #3 HWC	N/A	N/A	N/A	Coexist
2.2 SP02 HWC	N/A	N/A	N/A	N/A

Note: There was no 2.0 ESD #1 or 2.1 ESD #1 for BlackBerry Hybrid Web Container.

- N/A – not applicable.
- Coexist – the application is not upgraded; multiple versions of the application can coexist.
- In-place upgrade – the application is upgraded to the new version (you must modify the application to add new features).

See also

- *Migrate Hybrid Apps* on page 12

iOS

No migration changes are required for iOS Hybrid Apps; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate Hybrid Web Container from 2.1 ESD #3 to 2.2

Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.

Hybrid Web Container Migration Paths for iOS

Supported Hybrid Web Container migration paths on iOS, including paths for applications downloaded from the Apple App Store and those built from source code.

iOS Migration Paths (Applications Downloaded from Apple App Store)
This matrix identifies the supported Hybrid Web Container migration or the iOS container downloaded from Apple App store.

	2.1 HWC	2.1 ESD #2 HWC	2.1 ESD #3 HWC	2.2 SP02 HWC
2.1 HWC	N/A	Coexist	Coexist	Coexist
2.1 ESD #2 HWC	N/A	N/A	In-place upgrade	In-place upgrade
2.1 ESD #3 HWC	N/A	N/A	N/A	In-place upgrade
2.2 SP02 HWC	N/A	N/A	N/A	N/A

Note: There was no 2.1 ESD #1 Hybrid Web Container.

- N/A – not applicable.
- Coexist – the application is not upgraded; multiple versions of the application can coexist.
- In-place upgrade – the application is upgraded to the new version (you must modify the application to add new features).

iOS Migration Paths (Applications Built from Source Code)
This matrix identifies the supported Hybrid Web Container migration for the iOS container that one builds from the supplied source code while keeping the same "bundle ID" between versions.

	2.1 HWC	2.1 ESD #2 HWC	2.1 ESD #3 HWC	2.2 SP02 HWC
2.1 HWC	N/A	In-place upgrade	In-place upgrade	In-place upgrade
2.1 ESD2 HWC	N/A	N/A	In-place upgrade	In-place upgrade
2.1 ESD3 HWC	N/A	N/A	N/A	In-place upgrade
2.2 SP02 HWC	N/A	N/A	N/A	N/A

Note: There was no 2.1 ESD #1 Hybrid Web Container.

See also

- *Migrate Hybrid Apps* on page 12

Windows Mobile

No migration changes are required for Windows Mobile Hybrid Apps; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate Hybrid Web Container from 2.1 ESD #3 to 2.2

1. Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.
2. If you plan to run two different versions of the Windows Mobile Hybrid Web Container on the same device (such as version 2.1 ESD #3 and version 2.2), you must assign a unique App ID with custom code. See *Using Multiple Hybrid Web Containers on the Same Windows Mobile Device* on page 18.

Hybrid Web Container Migration Paths for Windows Mobile

Supported Hybrid Web Container (HWC) migration paths on Windows Mobile.

Table 11. Windows Mobile Migration Paths

	2.1 HWC	2.1 ESD #2 HWC	2.2 SP02 HWC
2.1 HWC	N/A	In-place upgrade	Coexist
2.1 ESD #2 HWC	N/A	N/A	Coexist
2.2 SP02 HWC	N/A	N/A	N/A

Note: There was no new 2.1 ESD #1 or 2.1 ESD #3 for Windows Mobile Hybrid Web Container; 2.1 ESD #3 shipped with 2.1 ESD #2 Windows Mobile clients.

- N/A – not applicable.
- Coexist – the application is not upgraded; multiple versions of the application can coexist.
- In-place upgrade – the application is upgraded to the new version (you must modify the application to add new features).

See also

- *Migrate Hybrid Apps* on page 12

Using Multiple Hybrid Web Containers on the Same Windows Mobile Device

You can configure two or more Hybrid Web Containers on a Windows Mobile device.

Each container can be installed separately on the same device, can connect to a different server, and can be used independently.

1. Create a Visual Studio project for each container.
2. For each container, edit the project's `config.properties` file and specify a unique AppID property for your container.
   For example: `AppID="HWC1"`.

   Note: Do not change the AppID property at runtime.
3. Rebuild the project, as described in *Building the Windows Mobile Hybrid Web Container Using the Provided Source Code*.
4. Configure the container's CAB build. In each project, edit the `OneBridge_ppc.inf` file and customize these properties:

   AppName – provide a unique name for each container.

   InstallDir – enter the path where the container is to be installed on the device. Each container must have a different path.

   Shortcuts – declare a shortcut that launches the container application. Users can change shortcut names. Shortcut names do not have to be unique.

   Here are sample customized lines in `OneBridge_ppc.inf`:

   ```
   [CEStrings]
   AppName = "HWC"
   InstallDir=%CE1%\Sybase\%AppName%
   ...
   [Shortcuts.All]
   Hybrid Web Container,0,HWCA.exe,%CE11%
   ```
5. Build the CAB file for each container, as described in *Packaging a CAB File*.

Hybrid Web Container API Changes in Version 2.2

Changes in the Hybrid Web Container API include refactored JavaScript API names and new JavaScript APIs.

JavaScript APIs Available in Designer

JavaScript APIs are now available from the Hybrid App Designer. Hybrid App Designer now generates Hybrid Apps that use the new JavaScript APIs from Hybrid Web Container.

Note: This means version 2.1 ESD #3 APIs are deprecated; use version 2.2 JavaScript APIs.

You can access generated API documents from *Developer Guide: Hybrid Apps*, via a link in *Hybrid Web Container and Hybrid App JavaScript API Reference*.

- **Hybrid Web Container –** the files where the Hybrid Web Container JavaScript APIs are defined are located in *SUP_HOME*\UnwiredPlatform\MobileSDK<version>\HybridApp\API\Container.
- **Hybrid App –** the files where the Hybrid App JavaScript APIs are defined are located in *SUP_HOME*\UnwiredPlatform\MobileSDK<version>\HybridApp\API\AppFramework.

Hybrid Web Container JavaScript Name Changes

Some JavaScript API names have been refactored. The JavaScript APIs are now in the "hwc" namespace and the word "Workflow" has been replaced with "HybridApp". The updated API can be found in the installation directory: *SUP_HOME*\MobileSDK22\HybridApp\API.

Hybrid App Framework

This release provides an optional application framework that you can use on top of the core Hybrid Web Container APIs to manage data and user interface exchange. JavaScript APIs are now provided in the installation directory.

Table 12. Refactored Hybrid App JavaScript Classes

Classes	Description
API.js	Functions that users typically call from within a function in Custom.js (or in a function contained in a different .js file called by a function in Custom.js). Roughly categorized into utility functions, message data functions, user interface functions, native functions, and validation functions.
Custom.js	Entry point where user-supplied code is added. Typical examples of methods include customBeforeMenuItemClick, customAfterDataReceived, and customBeforeShowScreen.
Utils.js	Functions included in the application framework that the Hybrid App invokes directly.

Classes	Description
`WorkflowMessage.js`	JavaScript objects and methods that provide an in-memory object hierarchy representation of messages being sent between the Hybrid App and the Unwired Server.
`Resource.js`	Access localized string resources.

Table 13. New JavaScript Methods

Methods	Description
`hwc.CustomIcon`	Indexes custom icons; defined in `hwc-api.js`.
• `doOnlineRequest` • `asynchronous` • `cachePolicy`	Makes online request calls.

Documented in generated developer documentation, and available from *Developer Guide: Hybrid Apps*, *Hybrid Web Container and Hybrid App JavaScript APIs*.

Access OData Sources

JavaScript APIs that enable a Hybrid App to access OData enterprise information system (EIS) data sources.

Table 14. New OData SDK Access

Classes	Description
`Data.js`	Open source API shipped with Unwired Platform

Documented in:

- *Developer Guide: Hybrid Apps*, see *Develop OData-based Hybrid Apps*

Migrate OData Applications

No steps are required to migrate 2.1 ESD #3 applications to version 2.2; however, you may need to perform some migration steps to take advantage of new features.

Before using Sybase Mobile SDK 2.2, explicitly migrate OData applications built on 2.1 ESD # 3 or earlier releases to 2.2 using the migration guidelines for Android, BlackBerry, and iOS

applications. The OData SDK APIs in Unwired Platform 2.2 have been refined and enhanced for better usage. When you add the Unwired Platform 2.2 OData SDK into your existing application, the application will encounter compilation errors, that should be resolved.

OData Client Compatibility Matrix

Compatibility between versions of OData clients and Unwired Server.

OData Client and Unwired Server Version Compatibility

OData SDK Client	Unwired Server 2.1	Unwired Server 2.1 ESD #1	Unwired Server 2.1 ESD #2	Unwired Server 2.1 ESD #3	Unwired Server 2.2 SP02
OData SDK Client 2.1	Yes	Yes	Yes	Yes	Yes
OData SDK Client 2.1 ESD #1	No	Yes	Yes	Yes	Yes
OData SDK Client 2.1 ESD #2	No	Yes	Yes	Yes	Yes
OData SDK Client 2.1 ESD #3	No	Yes	Yes	Yes	Yes
OData SDK Client 2.2 SP02	No	Yes	Yes	Yes	Yes

Android

No migration changes are required for OData Android applications; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate OData SDK Version 2.1 ESD #3 to 2.2

- For information and examples for migrating existing 2.1 ESD #3 Android applications to 2.2, see *Migrating Android Applications* on page 22.
- Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully

support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.

- Any device applications that implement Afaria libraries need to be recompiled to use standalone Afaria libraries, and reprovisioned.
  1. Download the latest Afaria libraries from: *http://frontline.sybase.com/support/downloads.aspx* (registration required).
  2. Copy the libraries to a new location.
  3. Relink the libraries in your development environment. See *Developer Guide: OData SDK*, *Downloading the Latest Libraries* (Android section).

Migrating Android Applications

Migrate Android Online Data Proxy applications from 2.1 ESD #3 (and earlier) to 2.2.

These steps use an example scenario that includes before and after code snippets, removed code snippets, and additional information.

1. Modify user registration code, and all APIs related to registration. Error handling that used exceptions in 2.1 ESD #3 and earlier uses standard error objects in 2.2.

   Before:

```
try {
     LiteUserManager.initInstance(getContext(), helper.APP_NAME);
      LiteUserManager lum = LiteUserManager.getInstance();
      lum.clearServerVerificationKey();
      LiteUserManager.enableHTTPS(true);

      LiteMessagingClient.setODPHTTPAuthChallengeListener(this);

      lum.setConnectionProfile(Helper.SEVERIP,
                               Helper.SERVERPORT, helper.COMPANYID);
      lum.registerUser(Helper.USERNAME, Helper.ACT_CODE);

    }  catch (MessagingClientException oe)
    {      // Exception handling
    }
```

   After:

```
try {
        ODPUserManager.initInstance(getContext(),
Helper.APP_NAME);
        ODPUserManager oum = null;
        oum = ODPUserManager.getInstance();
        ODPClientConnection.clearServerVerificationKey();
        ODPUserManager.enableHTTPS(true);

       ODPClientConnection.setODPHTTPAuthChallengeListener(this);
        oum.setConnectionProfile(Helper.SEVERIP,
        Helper.SERVERPORT,  Helper.COMPANYID);
        lum.registerUser(Helper.USERNAME, Helper.ACT_CODE, true);
    } catch (ODPException oe)
```

```
    {     // Exception Handling
    }
```

Removed: all asynchronous method calls, and the method call that takes in the vault API.

2. Modify data fetch-related code.

   Before:

```
ISDMConnectivitiyParameters params = new
SDMConnectivityParameters();
  params.setLanguage("en");
  params.setUserName(backendUsername);
  params.setUserPassword(backendPassword);
  params.setBaseUrl(url);

SDMPreferences preference = new SDMPreferences();
  requestManager  = new SDMRequestManager(
              new SDMLogger(preference), preference, params, 2);

  final ISDMRequest request = new SDMBaseRequest();
  Hashtable headers = new Hashtable();
  headers.put("X-CSRF-Token", "fetch");
  request.setHeaders(headers);
  request.setPriority(ISDMRequest.PRIORITY_HIGH);
  request.setRequestMethod(ISDMRequest.REQUEST_METHOD_GET);
  request.setRequestUrl(url);
  request.setListener(listener);

requestManager.makeRequest(request);
```

   After:

```
ISDMConnectivitiyParameters params = new
SDMConnectivityParameters();
  params.enableXsrf(true);
  params.setLanguage("en");
  params.setUserName(backendUsername);
  params.setUserPassword(backendPassword);
  params.setBaseUrl(url);

SDMPreferences preference = new SDMPreferences();
  requestManager  = new SDMRequestManager(
              new SDMLogger(preference), preference, params, 2);
  final ISDMRequest request = new SDMBaseRequest();

  request.setPriority(ISDMRequest.PRIORITY_HIGH);
  request.setRequestMethod(ISDMRequest.REQUEST_METHOD_GET);
  request.setRequestUrl(url);
  request.setListener(listener);

requestManager.makeRequest(request);
```

   Removed: no method calls have been removed from the request interface.

3. Modify user deletion code, and related API code.

   Before:

```
LiteUserManager lum = LiteUserManager.getInstance();
lum.deleteUser();
```

After:

```
ODPUserManager oum = ODPUserManager.getInstance();
oum.deleteUser();
```

Removed: nothing has been removed from the interface.

4. Modify native and online push notification. For Android, 2.1 ESD #3 and earlier supported online push only via the messaging channel, and did not support native notifications. In 2.2, a new message call has been added for registering native notifications.

   Before:

   ```
   UserManager.setPushListener(ISDMNetListener listener);
   ```

   After:

   ```
   ODPClientConnection. registerForPayloadPush(ISDMNetListener
   listener);
   ODPClientConnection.
   registerForNativePush(IODPPushNotificationListenerlistener);
   ```

   Removed: nothing has been removed in the interface.

5. Modify certificate management API code sections.

   Before:

   ```
   Vector v = CertificateStore.listAvailableCertificatesFromStore();
   String certificate getSignedCertificateFromStore(v.elementAt(0));
   ```

   After:

   ```
   Vector v =
   ODPCertificateManager.listAvailableCertificatesFromStore();
   String certificate getSignedCertificateFromStore(v.elementAt(0));
   ```

   Removed: the `getSignedCertificateFromStore()` API has been removed along with the Afaria method calls.

6. Make modifications to implement additional changes and new features, then recompile your code if required.

   Since Afaria is a standalone, separately consumable library in 2.2, no methods related to Afaria are exposed as a part of the ODP interface. Application developers must consume the Afaria JARs directly.

BlackBerry

No migration changes are required for OData BlackBerry applications; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate OData SDK Version 2.1 ESD #3 to 2.2

- For information and examples for migrating existing 2.1 ESD #3 BlackBerry applications to 2.2, see *Migrating BlackBerry Applications* on page 25.
- Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.
- Afaria does not support BlackBerry, so no project or client changes are required for BlackBerry migration from 2.1 ESD #3 to 2.2.

Migrating BlackBerry Applications

Migrate BlackBerry Online Data Proxy applications from 2.1 ESD #3 to 2.2.

These steps use an example scenario that includes before and after code snippets, removed code snippets, and additional information.

1. Modify user registration code, and all APIs related to registration. Error handling has been changed from exceptions in 2.1 ESD #3 and earlier to standard error objects in 2.2.

   Before:

   ```
   try {
            UserManager. clearServerVerificationKey();
            UserManager.initialize(HomeScreen.appID);

   UserManager.setConnectionProfile(serverIP,serverPort,farmID);
            UserManager.registerUser(userName, activationCode);
            UserManager.addUserRegistrationListener(this);
            UserManager. setODPHTTPAuthChallengeListener(this);
            UserManager. setODPHttpErrorListener(this);

       }  catch (MessagingClientException oe) {
       }
   ```

   After:

   ```
   try {
            ODPClientConnection.clearServerVerificationKey();
            ODPUserManager.initInstance(HomeScreen.appID);
            oum = ODPUserManager.getInstance();
   ```

```
        oum.setConnectionProfile(serverIP, serverPort, farmID);
        oum.registerUser(userName,activationCode,true);
        oum. setUserRegistrationListener(this);
        ODPClientConnection. setODPHTTPAuthChallengeListener
(this);
        ODPClientConnection. setODPHttpErrorListener(this);

    }  catch (ODPException oe) {
    }
```

Removed: nothing has been removed from the interface.

2. Modify data fetch-related code.

   Before:

```
ISDMConnectivitiyParameters params = new
SDMConnectivityParameters();
    params.setLanguage("en");
    params.setUserName(backendUsername);
    params.setUserPassword(backendPassword);
    params.setBaseUrl(url);
  SDMPreferences preference = new SDMPreferences();
    requestManager  = new SDMRequestManager(
      new SDMLogger(preference), preference, params, 2);
    final ISDMRequest request = new SDMBaseRequest();
    Hashtable headers = new Hashtable();
  headers.put("X-CSRF-Token", "fetch");
    request.setHeaders(headers);
    request.setPriority(ISDMRequest.PRIORITY_HIGH);
    request.setRequestMethod(ISDMRequest.REQUEST_METHOD_GET);
    request.setRequestUrl(url);
    request.setListener(listener);
  requestManager.makeRequest(request);
```

   After:

```
ISDMConnectivitiyParameters params = new
SDMConnectivityParameters();
    params.enableXSRF(true);
    params.setLanguage("en");
    params.setUserName(backendUsername);
    params.setUserPassword(backendPassword);
    params.setBaseUrl(url);
  SDMPreferences preference = new SDMPreferences();
    requestManager  = new SDMRequestManager(
      new SDMLogger(preference), preference, params, 2);
    final ISDMRequest request = new SDMBaseRequest();

    request.setPriority(ISDMRequest.PRIORITY_HIGH);
    request.setRequestMethod(ISDMRequest.REQUEST_METHOD_GET);
    request.setRequestUrl(url);
    request.setListener(listener);
  requestManager.makeRequest(request);
```

   Removed: nothing has been removed from the interface.

3. Modify user deletion code, and related API.

Before:

```
UserManager. deleteUser();
```

After:

```
ODPUserManager.initInstance(HomeScreen.appID);
oum = ODPUserManager.getInstance();
oum.deleteUser();
```

Removed: nothing has been removed from the interface.

4. Modify native notification (push) code, and related API.

   Before:

   ```
   UserManager.setPushListener(ISDMNetListener listener);
   ```

   After:

   ```
   ODPClientConnection. registerForPayloadPush(ISDMNetListener
   listener);
   ODPClientConnection.
   registerForNativePush(IODPPushNotificationListenerlistener);
   ```

   Removed: nothing has been removed from the interface.

5. Make modifications to implement additional changes and new features, then recompile your code if required.

iOS

No migration changes are required for OData iOS applications; however, you might need to make some changes to take advantage of new features, or when you modify an application.

Migrate OData SDK Version 2.1 ESD #3 to 2.2

- For information and examples for migrating existing 2.1 ESD #3 iOS applications to 2.2, see *Migrating iOS Applications* on page 28.
- Scale-out nodes take requests only from messaging clients (OData SDK, Hybrid Web Container) and HTTP clients (REST APIs). For these clients to connect to the scale-out node, clients must be built with Unwired Platform version 2.2. Only 2.2 clients can fully support HTTP cookies. You must migrate existing clients to version 2.2 if you want to connect to scale-out nodes. For details about cookie support, see the corresponding *Developer Guide* for your client type.
- Any device applications that implement Afaria libraries need to be recompiled to use standalone Afaria libraries, and reprovisioned.
  1. Download the latest Afaria libraries.
  2. Copy the libraries to a new location.
  3. Relink the libraries in your development environment. See *Developer Guide: OData SDK*, *Downloading the Latest Libraries* (iOS section).

Migrating iOS Applications

Migrate iOS Online Data Proxy applications from 2.1 ESD #3 (or earlier) to 2.2.

These steps use an example scenario that includes before and after code snippets, removed code snippets, and additional information.

1. Modify user registration code, and all APIs related to registration.

   In the following before and after code examples, which show an automatic registration scenario:
   - The user manager class is renamed to comply with Sybase Unwired Platform naming standards.
   - Error handling has been changed from exceptions in 2.1 ESD #3 or earlier to standard error objects in 2.2.
   - With 2.2, asynchronous user registration follows the delegation design pattern of iOS, that is, the application developer implements an `ODPUserManagerDelegate` to receive failure or success notifications, compared to the behavior in earlier versions, where application-defined selectors were assigned for success and failure callbacks.
   - The asynchronous registration call has been merged with the normal registration call, and a simple Boolean determines whether the call is synchronous.

   Before:

```
@try
{
    if ([LiteSUPAppSettings isSUPKeyProvisioned]) {
        [LiteSUPUserManager clearServerVerificationKey];
    }
    LiteSUPUserManager* userManager = [LiteSUPUserManager
getInstance:@"NewFlight"];

    [ODPClientListeners
setCertificateChallengeListenerDelegate:self];
    [ODPClientListeners
setHTTPAuthChallengeListenerDelegate:self];
    [ODPClientListeners setHTTPErrorListenerDelegate:self];

    [userManager setDelegate:self];
    [userManager
setDidFailToRegisterUser:@selector(registrationSuccessful:)];
    [userManager
setDidSuccessfulUserRegistration:@selector(registrationFailed:)];

    [userManager setConnectionProfile:@"10.53.138.119"
withSupPort:5001 withServerFarmID:@"0"];
    [userManager registerUser:@"supuser"
withSecurityConfig:@"HttpAuth" withPassword:@"s3puser"];
}
@catch (NSException *exception)
{
```

```
    NSLog(@"%@", [exception reason]);
}
```

After:

```
    if ([ODPAppSettings isServerKeyProvisioned]) {
        [ODPClientConnection clearServerVerificationKey];
    }
ODPUserManager* userManager = [ODPUserManager
getInstance:@"com.sap.NewFlight"];
[ODPClientListeners
setCertificateChallengeListenerDelegate:self];
[ODPClientListeners setHTTPAuthChallengeListenerDelegate:self];
[ODPClientListeners setHTTPErrorListenerDelegate:self];

[userManager setDelegate:self];

[userManager setConnectionProfileWithHost:@"10.53.138.119" port:
5001 farm:@"0" error:nil];
NSError* regError = nil;
[userManager registerUser:@"supuser" securityConfig:@"SSO"
password:@"s3puser" error:&regError isSyncFlag:NO];

if (regError) {
    NSLog(@"%@", regError);
}
```

Removed:

```
[userManager registerUser:@"user" withSecurityConfig:@"sec"
withPassword:@"pwd" withVaultPassword:@"vaultpwd"];
[userManager registerUserAsynchronousWithUserName:@"user"
activationCode:@"code"];
[userManager registerUserAsynchronousWithUserName:@"user"
securityConfig:@"sec" password:@"pwd"];
[userManager registerUserAsynchronousWithUserName:@"user"
securityConfig:@"sec" password:@"pwd" vaultPassword:@"vaultpwd"];
[userManager setConnectionProfileFromAfaria:url
appUrlScheme:urlScheme];
NSMutableDictionary* settings = [userManager
getSettingsFromAfariaWithUrl:url UrlScheme:urlScheme];
```

2. Modify data fetch-related code.

   There are no major differences in using the `SDMRequesting` interface with Sybase Unwired Platform 2.2. All basic API code remains the same. The renaming of the ODP request call is abstracted by the `SDMRequestBuilder` and does not affect the application. In the following before and after example for data fetch code:

   - One major difference is XCSRF handling for the client. With 2.2, clients are responsible for token persistence during the session. Enable XCSRF handling by calls to a simple method in the `SDMRequesting` interface. In 2.1 ESD #3 and earlier, fetching the token and passing it in subsequent update calls was handled by the application itself.

- The new get Endpoint call method returns an error in case of failure when fetching the endpoint. You can write your application to receive the error or not. This holds true for the push Endpoint method as well.

Before:

```
id<SDMRequesting> request = [SDMRequestBuilder requestWithURL:
[NSURL URLWithString: [LiteSUPAppSettings
getApplicationEndPoint]];
[request setUsername:@"user"];
[request setPassword:@"pwd"];
[request setDelegate:self];
[request setRequestMethod:@"GET"];
[request addRequestHeader:@"X-CSRF-Token" value:@"Fetch"];
[request setDidFailSelector:@selector(requestFailed:)];
[request setDidFinishSelector:@selector(requestFinished:)];
[request startAsynchronous];

NSString* xCsrfToken = [[request responseHeaders]
objectForKey:@"X-CSRF-TOKEN"];
```

After:

```
[SDMRequestBuilder enableXCSRF:YES];
id<SDMRequesting> request = [SDMRequestBuilder requestWithURL:
[NSURL URLWithString: [ODPAppSettings
getApplicationEndpointWithError:nil]]];
[request setUsername:@"user"];
[request setPassword:@"pwd"];
[request setDelegate:self];
[request setRequestMethod:@"GET"];
[request setDidFailSelector:@selector(requestFailed:)];
[request setDidFinishSelector:@selector(requestFinished:)];
[request startAsynchronous];
```

Removed: nothing has been removed from the `SDMRequesting` interface.

3. Modify user deletion code, and related API code.

   The difference here is the restructuring of classes. The message to stop the client has been renamed and grouped under a different class, and exception handling has been replaced with error handling using the standard error object.

   Before:

```
@try {
    LiteSUPUserManager* userManager = [LiteSUPUserManager
getInstance:@"NewFlight"];
    [userManager shutDown];
    [userManager deleteUser];
}
@catch (NSException *exception) {
    NSLog(@"%@", [exception reason]);
}
```

   After:

```
ODPUserManager* userManager = [ODPUserManager
getInstance:@"com.sap.NewFlight"];
ODPClientConnection* clientConnection = [ODPClientConnection
getInstance:@"com.sap.NewFlight"];
[clientConnection stopClient];
NSError* error = nil;
[userManager deleteUserWithError:&error];
```

Removed: nothing has been removed from the interface.

4. Modify APNS code, and related API code.

   This section discusses the client-side API, which gets the device token and passes it to Unwired Server. The only major change is renaming the class that holds these methods.

   Before:

   ```
   [LiteSUPMessagingClient setupForPush:app];
   [LiteSUPMessagingClient deviceTokenForPush:app
   deviceToken:token];
   [LiteSUPMessagingClient pushNotification:app
   notifyData:dataDict];
   [LiteSUPMessagingClient pushRegistrationFailed:app
   errorInfo:error];
   ```

   In version 2.1 ESD #3, online push with payload was achieved with `setDelegate` calls. Also, a new delegate `SDMSUPPushDelegate` has been adapted, and its method `pushNotificationReceived:` implemented.

   ```
    [SUPUtilities setDelegate:self];
   ```

   The delegates for these methods have remained the same; the information has not been repeated here.

   After:

   ```
   [ODPClientConnection setupForPush:app];
   [ODPClientConnection deviceTokenForPush:app deviceToken:token];
   [ODPClientConnection pushNotification:app notifyData:dataDict];
   [ODPClientConnection pushRegistrationFailed:app errorInfo:error];
   ```

   In version 2.2, the online push with payload calls were adapted to the delegate `ODPPushDelegate`, and its method `pushNotificationReceived:` implemented.

   ```
   [ODPClientConnection registerForPayloadPush:self];
   ```

   Removed: nothing has been removed from the interface.

5. Modify certificate management API code sections.

   Before:

   ```
   LiteSUPCertificateStore* store = [LiteSUPCertificateStore
   getInstance];
   NSString* base64string = [store
   getSignedCertificateFromFile:filePath
   withCertificatePassword:password];
   ```

After:

```
NSString* base64string = [ODPCertificateManager
getSignedCertificateFromFile:filePath
withCertificatePassword:password];
```

Removed:

```
LiteSUPCertificateStore* store = [LiteSUPCertificateStore
getInstance];
[store getSignedCertificate:cert
withCertificatePassword:password];
[store getSignedCertificateFromAfariaForURL:url
withUsername:username withPassword:password];
[store getSignedCertificateFromAfariaForURLScheme:urlScheme
withUsername:username withPassword:password];
[store getSignedCertificateFromServer:server
withPassword:password withCertificatePassword:passCert];
```

Note: These methods were deprecated in 2.1 ESD #3, and have been removed in 2.2.

6. Make modifications needed to implement additional changes and new features, then recompile your code if required.

   Since Afaria is a standalone, separately consumable library in version 2.2, no methods related to Afaria are exposed as a part of the Online Data Proxy interface. Application developers must consume the Afaria library directly.

OData SDK API Changes in Version 2.2

Changes in the OData SDK API include Afaria, DataVault, refactored ODP class names, and ODPException.

Afaria APIs

Afaria APIs are no longer packaged with the OData SDK API, but are now available directly from the standalone Afaria library: *http://frontline.sybase.com/support/downloads.aspx* (registration required).

Table 15. Afaria Methods Removed from OData SDK

Methods	Platform
• `setConnectionProfileFromAfaria()` • `getSettingsFromAfaria()` • `getSignedCertificateFromAfaria()`	Android

Methods	Platform
• `setConnectionProfileFromAfaria()` • `getSettingsFromAfaria()` • `getSignedCertificateFromAfaria()`	BlackBerry
• `setConnectionProfileFromAfariaForUrl` • `getSettingsFromAfariaForUrl` • `getSignedCertificateFromAfaria`	iOS

Related topics or references have been removed from *Developer Guide: OData SDK*.

DataVault API

DataVault APIs are no longer packaged with the OData SDK API, but are now available from the standalone DataVault library that is packaged with the Sybase Mobile SDK.

Table 16. Deleted DataVault Methods

Methods	Platform
`LiteDataVault`	Android
`SUPDataVault`	BlackBerry
`LiteSUPDataVault`	iOS

Related topics or references have been removed from *Developer Guide: OData SDK*.

ODP Class Name Changes

Some API class names have been refactored to keep naming conventions across all platforms. APIs are logically grouped in the corresponding refactored classes. For consistency, some new classes have been added.

Table 17. Changed (Refactored) Class Names

Methods	Platform
• `ODPAppSettings` – refactored from `LiteAppSettings`. • `ODPCertificateManager` – refactored from `LiteCertificateStore`. • `ODPUserManager` – refactored from `LiteUserManager`. • `ODPException` – new class. • `ODPClientConnection` – refactored from `LiteMessagingClient`.	Android
• `ODPAppSettings` – refactored from `AppSettings`. • `ODPCertificateManager` – refactored from `CertificateStore` • `ODPUserManager` – refactored from `UserManager`. • `ODPException` – new class. • `ODPClientConnection` – APIs moved into this class for logical grouping.	BlackBerry
• `ODPAppSettings` – refactored from `AppSettings`. • `ODPCertificateManager` – refactored from `CertificateStore`. • `ODPUserManager` – refactored from `UserManager`.	iOS

See *Developer Guide: OData SDK*, the *ODP SDK API Usage* topic (iOS, Android, and BlackBerry sections).

ODPException API

Error codes thrown by ODP APIs are defined in the `ODPException` class.

See *ODPException class* (Android and BlackBerry) documentation in Javadoc.

Migrate REST API Applications

No migration changes are required for REST API applications.

Migrate JCo Connections

Understand default behavior when configuring Unwired Server to SAP® EIS JCo connections.

By default, Sybase Unwired Platform version 2.2 JCo connections follow these rules:

1. Each distinct connection of credentials and connection parameters creates a new destination name.
2. Each destination name may pool one connection.
3. Each attempted invocation on a destination opens two socket-based RFC connections to the remote system: one for the user's JCo destination, and one for the associated JCo repository's automatically generated destination.
4. By default, each destination may cache up to one idle connection for up to five minutes. Evict these connections earlier if the global connection pool maximum size is reached.
5. Once the idle limit is reached, pooled connections are closed.

Index

U

V